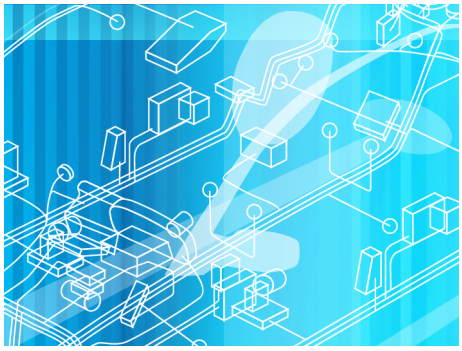
Fraunhofer Institut Software- und Systemtechnik

Grobentwurf des VEIA-Referenzprozesses

Martin Große-Rhode, Simon Euringer,
Ekkart Kleinod, Stefan Mann



ISST-Bericht 80/07
Januar 2007

Herausgeber: Fraunhofer-Gesellschaft e. V.
Institut für Software- und Systemtechnik
Leitung: Prof. Dr. Jakob Rehof
Institutsteil Berlin: Mollstraße 1
10178 Berlin
Institutsteil Dortmund: Joseph-von-Fraunhofer-Straße 20
44227 Dortmund

ISSN 0943-1624

Das Projekt VEIA

Das Projekt »Verteilte Entwicklung und Integration von Automotive-Produktlinien« wird vom Bundesministerium für Bildung und Forschung im Rahmen der Forschungsoffensive »Software Engineering 2006« unter dem Förderkennzeichen »01ISF15A« gefördert.

Am Projekt sind vier Partner beteiligt:

- BMW Group
- Fraunhofer ISST
- PROSTEP IMP GmbH
- Technische Universität München

Autoren

Dieses Dokument wurde geschrieben von

- Martin Große-Rhode (Fraunhofer ISST)
- Simon Euringer (BMW Group)
- Ekkart Kleinod (Fraunhofer ISST)
- Stefan Mann (Fraunhofer ISST)

Fraunhofer Institut Software- und Systemtechnik

Inhalt

1	**Einleitung**	3
1.1	Einordnung des Referenzprozesses in VEIA	3
1.2	Der Referenzprozess	3
1.3	Struktur des Dokuments	4
2	**Artefakte im Engineering-Prozess**	5
2.1	Übersicht über die Artefakte	6
2.2	Produktlinienbeschreibung	8
2.3	Dienstlandkarte und Dienste	10
2.4	Funktionsnetz	13
2.5	Softwarearchitektur	14
2.6	Technische Architektur	15
2.7	Artefaktbeziehungen	16
2.8	Erstellung, Bewertung und Pflege der Artefakte	19
2.9	Modellierungstechniken	24
3	**Prozessphasen**	25
3.1	Phase 1: Strategische Ziele sammeln	28
3.2	Phase 2: Produktlinie und Produkte definieren	29
3.3	Phase 3: Systemarchitekturen entwickeln	32
3.4	Phase 4: Subsysteme entwickeln	34
3.5	Phase 5: Implementierungen	37
3.6	Phase 6: Integration und Test	37
4	**Fallbeispiel »Condition Based Service« (CBS)**	39
4.1	Strategische Ziele / Rahmenbedingungen	41
4.2	Produktlinienbeschreibung	42
4.3	Dienstlandkarte und Beschreibung der Dienste	43
4.4	Funktionsnetz / Funktionsbeschreibung CBS	50
4.5	Technische Architektur	53
4.6	Verteilung des Funktionsnetzes auf die technische Architektur	54
4.7	Software	55
4.8	Varianz, Verteilung und Metriken	56
5	**Zusammenfassung und Ausblick**	63

Notation 65

Akronyme 67

Literatur 69

1 Einleitung

1.1 Einordnung des Referenzprozesses in VEIA

Das Projekt VEIA[1] hat sich das Ziel gesetzt, auf der Grundlage der Konzepte der Produktlinientechnik eine Methode für die verteilte Entwicklung und Integration von Automotive-Systemen zu erarbeiten, die sich an den konkreten Anforderungen industrieller Entwicklungsprozesse orientiert und praktisch anwendbar ist.

In der ersten sechsmonatigen Projektphase wurde die Grobstruktur eines Referenzprozesses festgelegt, der in erster Linie dazu dient, die Anforderungen an die vier Teilprojekte zu konkretisieren und die Aktivitäten der Projektpartner aufeinander abzustimmen.

Das Ergebnis des Projekts werden Methoden, Notationen und Werkzeuge sein, mit denen die Aktivitäten des Referenzprozesses durchgeführt und die Artefakte erstellt werden können. Projektextern dient der Vergleich des Referenzprozesses mit den Anforderungen der Anwendungsdomäne auch zur Validierung und Bewertung der Projektergebnisse.

1.2 Der Referenzprozess

Um das Projektziel zu erreichen, ist eine in Bezug auf die zu unterstützenden Prozesse realistische und pragmatische Herangehensweise erforderlich. Dazu ist mit dem Referenzprozess insbesondere ein Bindeglied zwischen den aus der Forschung bekannten Software- und Systementwicklungsprozessen auf der einen Seite und den in der Anwendungsdomäne bestehenden Produktentstehungsprozessen für E/E-Systeme auf der anderen Seite zu schaffen.

Mit Hilfe eines Prozessmodells werden die Artefakte, die im Engineering-Prozess

[1] Das Akronym VEIA steht für das vom BMBF geförderte Verbundprojekt »Verteilte Entwicklung und Integration von Automotive-Produktlinien« von Fraunhofer ISST, Technische Universität München, BMW AG und PROSTEP IMP GmbH.
Webseite: http://veia.isst.fraunhofer.de/

sukzessive zu entwickeln sind, identifiziert sowie deren kausale Abhängigkeiten beschrieben. Zusammen mit den entsprechenden Aktivitäten und Rollen – Erstellen, Bewerten und Pflegen der Artefakte – beschreiben sie den Engineering-Prozess. Begleitende und unterstützende Prozesse werden im VEIA-Prozessmodell nicht definiert.

Das vorliegende Dokument skizziert den Grobentwurf des VEIA-Referenzprozesses. Es werden folgende Prozessaspekte betrachtet:

- die Artefakte im Automotive-E/E-Engineering-Prozess und deren kausale Beziehungen,
- Modelle zur Beschreibung der Artefakte und
- die darauf aufbauenden Prozessphasen unter Berücksichtigung methodischer und organisatorischer Rahmenbedingungen.

1.3 Struktur des Dokuments

Das folgende Kapitel 2 stellt die Artefakte des Referenzprozesses vor. Dabei werden die Artefakte motiviert und in ihren Beziehungen zu den anderen Artefakten dargestellt.

Im daran anschließenden Kapitel 3 wird der Prozess als zeitliche Abfolge von Phasen beschrieben. Dabei werden die Artefakte aufgegriffen und je nach Entstehung bzw. Verwendung in eine zeitliche Reihenfolge gebracht.

Kapitel 4 greift noch einmal die Artefaktsicht auf und illustriert diese am Beispiel Condition Based Service (CBS).

Kapitel 5 fasst die Inhalte des Dokuments noch einmal kurz zusammen und gibt einen kleinen Ausblick auf die anschließenden Aktivitäten in VEIA.

Den Schluss des Dokuments bilden die Kapitel »Notation« und »Akronyme«, die bei Verständnisschwierigkeiten helfen sollen, Abbildungen oder Begriffe zu klären.

2 Artefakte im Engineering-Prozess

Die Anwendungsdomäne Automotive-E/E-Systeme legt einen systemorientierten Ansatz nahe. Hardware und Software werden simultan entwickelt. Beide Entwicklungen sind in den längerfristigen Entstehungsprozess der gesamten Fahrzeugreihe integriert. Mögliche Wechselwirkungen zwischen diesen Entwicklungsprozessen müssen zu jedem Zeitpunkt berücksichtigt werden können. Lösungsvarianten müssen erfasst und analysiert werden können, um frühzeitig im Prozess Entscheidungen treffen und nachvollziehen zu können.

Neben den Entwicklungsaktivitäten müssen explizit auch die notwendigen Analyseaktivitäten beschrieben werden. Darunter fallen verschiedene Arten der Architekturbewertung, die deutlich machen, welche Arten von Architekturbeschreibungen benötigt werden, zu welchem Zeitpunkt sie benötigt werden, und in welchem Maße sie die Projektentwicklung optimieren können.

Der Grobentwurf des VEIA-Referenzprozesses definiert die Anforderungen an die Artefakte, die als Zwischenergebnisse im Prozess erstellt werden. Sie dienen der Abstimmung zwischen den beteiligten Entwicklungspartnern und Stakeholdern sowie zur Qualitätssicherung. Es wird beschrieben, welche Entwicklungs- und Analyseaktivitäten vorzusehen sind, welche Informationen dazu jeweils benötigt und welche geliefert werden. Dieser Zusammenhang wird durch kausale Abhängigkeiten zwischen den Artefakten erfasst. In welchem Zeitraster diese Aktivitäten auszuführen sind, wird durch Prozessphasen definiert (siehe Kapitel 3).

Abschnitt 2.1 gibt einen Überblick über die in diesem Kapitel diskutierten Entwicklungsartefakte. In den nachfolgenden Abschnitten 2.2 bis 2.6 werden die einzelnen Entwicklungsartefakte mit ihren gegenseitigen Beziehungen detailliert diskutiert. Die Festlegung auf diese Artefakte dient als Grundlage für die weitere Projektarbeit in Bezug auf die Unterstützung der Produktlinienentwicklung durch Methoden, Notationen, Werkzeuge und Analyseverfahren. In Abschnitt 2.7 werden zusammenfassend die Artefaktbeziehungen systematisiert, worauf aufbauend in Abschnitt 2.8 Aktivitäten zur Erstellung der Artefakte mit ihrer prinzipiellen Reihenfolge beschrieben werden. Modellierungstechniken werden zusammenfassend in Abschnitt 2.9 diskutiert.

2.1 Übersicht über die Artefakte

Artefakte repräsentieren Zwischenergebnisse im Engineering-Prozess, die zur Diskussion und Abstimmung verwendet werden: Ziele, die mit den zu entwickelnden Systeme verfolgt werden, ihre Konkretisierung in Form von Anforderungen, darauf aufbauende Lösungen mit den zugrundeliegenden Entwurfsentscheidungen.

Die Entwicklungsartefakte lassen sich folgenden Bereichen zuordnen:

Produkte zur Beschreibung, welche Systeme mit welchen Eigenschaften (Merkmalen) entwickelt werden

Der Bereich umfasst die Definition des Produktlinienumfangs in Form von Produktmerkmalen und Produktbenennungen. Üblicherweise werden in diesem Bereich die Fahrzeugproduktlinien definiert.

Funktionen zur Beschreibung der zu implementierenden Funktionalität der Systeme

Der Bereich wird über die Artefakte grob in die Abstraktionsebenen Anforderungen (Beschreibung der Dienste), logischer Entwurf (Vernetzung von Funktionen) und Implementierungsentwurf (Softwarearchitektur und Beschreibung der Softwarekomponenten) unterteilt. Er entspricht damit der Struktur bekannter Softwareentwicklungsprozesse.

Infrastruktur zur Beschreibung der technischen Ressourcen, die zur Umsetzung der Funktionalität verwendet werden.

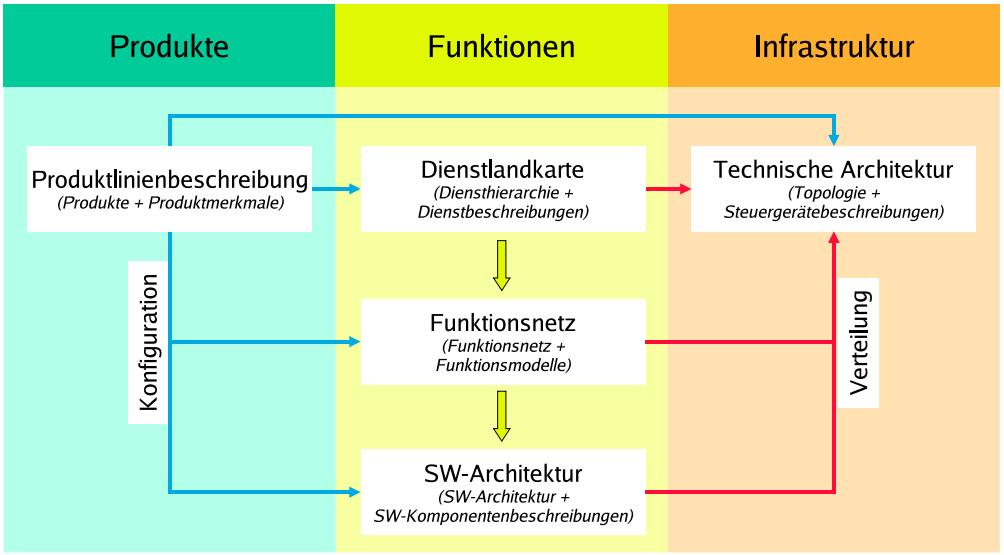
Der Bereich umfasst die Beschreibung der technischen Architektur des Gesamtsystems und die Spezifikation der einzelnen Steuergeräte.

Die konkrete Ausprägung der Artefakte orientiert sich an methodischen Anleitungen, um schrittweise und zielgerichtet vom »Was soll entwickelt werden?« zum »Wie wird dies am besten entwickelt und umgesetzt?« zu kommen. Zum Beispiel umfassen die Artefakte im Bereich »Funktionen« funktionale Anforderungssammlungen (in Form von Diensten) zur Erfassung der Funktionalität aus Nutzungssicht (Black-Box-Sicht auf das System), Entwurfsentscheidungen in Form von Funktionsnetzen zur Festlegung der zu erfassenden und zu verarbeitenden Informationen sowie die Festlegung von Softwarearchitekturen als informationstechnische Umsetzung der Funktionalität.

Folgende Entwicklungsartefakte werden im VEIA-Referenzprozess für Automotive-E/E-Systeme berücksichtigt (siehe Abbildung 1):

– Produktlinienbeschreibung (Abschnitt 2.2),

Abbildung 1 Artefakte und ihre Beziehungen im VEIA-Referenzprozess.



- Dienstlandkarte (Abschnitt 2.3),
- Funktionsnetz (Abschnitt 2.4),
- Softwarearchitektur (Abschnitt 2.5),
- Technische Architektur (Abschnitt 2.6),
- Konfigurationsmodelle (Abschnitt 2.7.2) und
- Verteilungsmodelle (Abschnitt 2.7.3).

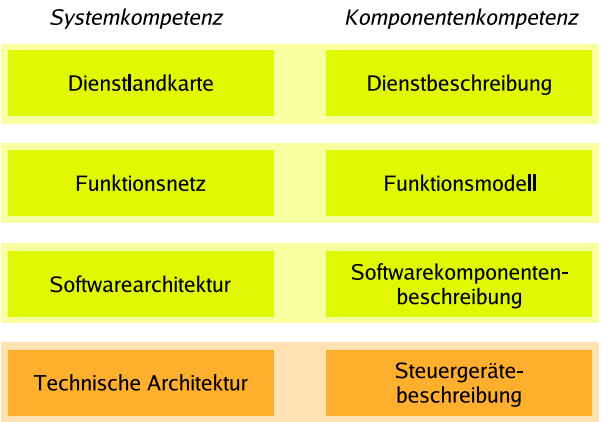
Wie in Abbildung 1 dargestellt ist, sind diese Artefakte den zuvor skizzierten Bereichen zugeordnet sowie miteinander in Beziehung gesetzt, da die Artefakte entweder aufeinander aufbauen oder aber miteinander konsistent sein müssen.

Im Unterschied zu Systementwicklungsprozessen, in denen der Entwurf von Lösungen, insbesondere von Software und Infrastrukturen, immer erst nach der Analyse der – funktionalen und nicht-funktionalen – Anforderungen erfolgt, enthält der VEIA-Referenzprozess mehrere Einstiegspunkte.

In den ersten Prozessschritten können neben der Erfassung der Anforderungen an die Funktionen des Systems (*Dienste*) bereits Anforderungsanalyse und Entwurf der Infrastruktur erfolgen. Dies ist dadurch begründet, dass technische Architekturen (Steuergerätetopologie, Kommunikationstechnologien etc.) oft projekt- und produktlinienübergreifend entwickelt werden. Der Bereich »Produkt« repräsentiert die Erweiterung des Entwicklungsprozesses von Einzelsystemen zu Systemfamilien.

Im Sinne der Produktlinientechnik wird dadurch der Umfang der Produktlinie definiert.

Abbildung 2 Artefakte: System- vs. Komponentendimension.



Orthogonal zu den Abstraktionsebenen, die durch die Artefaktgruppen repräsentiert werden, ist die Unterscheidung in System- und Komponentenebene wichtig und im Referenzprozess zu berücksichtigen (siehe Abbildung 2). Auf Komponentenebene sind spezielle Kompetenzen (Verantwortlichkeiten, Fachlichkeiten etc.) für die Erstellung von entsprechenden Artefakten (einzelne Dienstbeschreibungen, Funktionsmodelle, Softwarekomponentenbeschreibungen, Steuergerätebeschreibungen) erforderlich. Sie reflektieren zudem eine entsprechende Aufgabenverteilung im Engineering-Prozess.

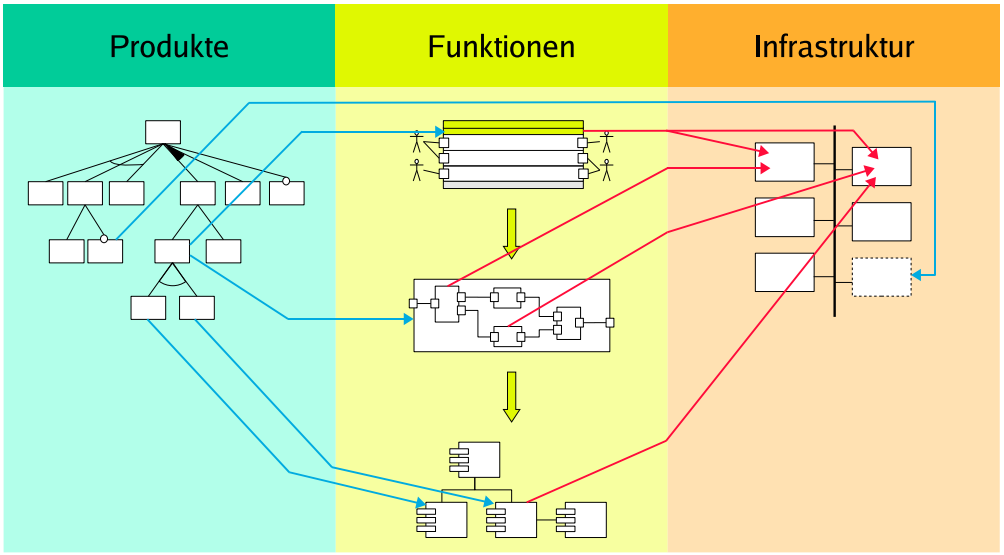
System- und Komponentenebene wird in allen Artefaktgruppen unterschieden. Die Aufteilungen sind jedoch nicht kongruent, da sich im allgemeinen keine 1:1-Beziehungen zwischen Diensten, Funktionen, Softwarekomponenten und Steuergeräten herstellen lassen.

In Abbildung 3 sind die im Folgenden diskutierten Entwicklungsartefakte beispielhaft illustriert.

2.2 Produktlinienbeschreibung

In der Produktlinienbeschreibung werden die Produktmerkmale erfasst, die innerhalb der Produktlinie insgesamt angeboten werden. Damit werden die von außen – aus Kunden- bzw. Marketingsicht – sichtbaren Merkmale an die zu entwickelnde

Abbildung 3 Artefakte und ihre Beziehungen im VEIA-Referenzprozess (schematische Illustration).



Systeme dargestellt. Merkmale können sich sowohl auf Dienste und Funktionen als auch auf Infrastrukturen beziehen (vgl. auch Abschnitt 2.7.2).

Die Produktlinienbeschreibung im Bereich »Produkte« ist im Allgemeinen eine Fahrzeugproduktlinienbeschreibung. Ihre Varianz und ihre Merkmale legen den Umfang der Fahrzeugproduktlinie fest. Damit einhergehend ist die Varianz für die anderen Artefakte (Dienste, Funktionen, Soft- und Hardware) vorgegeben und wird dort größtenteils übernommen. Jedes Artefakt besitzt aber eigene Varianz, die nicht aus der Fahrzeugproduktlinie stammt. Damit können in der Softwarearchitektur eine eigene Softwareproduktlinie und in der technischen Architektur eine eigene Steuergeräteproduktlinie entstehen, die jedoch innerhalb der Modelle verwaltet werden.

Zur einer Produktlinienbeschreibung gehören zum einen die Identifikation der Produkte der Produktlinie (z. B. Benennung der Fahrzeugtypen, Baureihen etc.), zum anderen die Erfassung der Merkmale der Produktlinie und ihre Zuordnung zu den Produkten. Dabei werden die Merkmale dahingehend strukturiert, ob es bzgl. der Produktlinie invariante, variierende oder optionale Merkmale sind. Invariante Merkmale gelten für alle Produkte der Produktlinie. Variierende und optionale Merkmale beschreiben die Abweichungen, die zwischen den Produkten der Produktlinie bestehen. Weitere Beziehungen wie Abhängigkeiten oder gegenseitiger Ausschluss müssen ebenfalls erfasst werden.

Die Zuordnung von Merkmalen zu Produkttypen und -gruppen über Ausstattungspakete sowie die Unterscheidung zwischen Serien- und Sonderausstattungen basieren auf der Analyse der Beziehungen zwischen den Produktlinienmerkmalen.

Die Produktlinienbeschreibung dient als Basis zur Konfiguration der Entwicklungsartefakte der Bereiche »Funktionen« und »Infrastruktur«, um in deren Modellen die einzelnen Produkte identifizieren und ableiten zu können.

2.3 Dienstlandkarte und Dienste

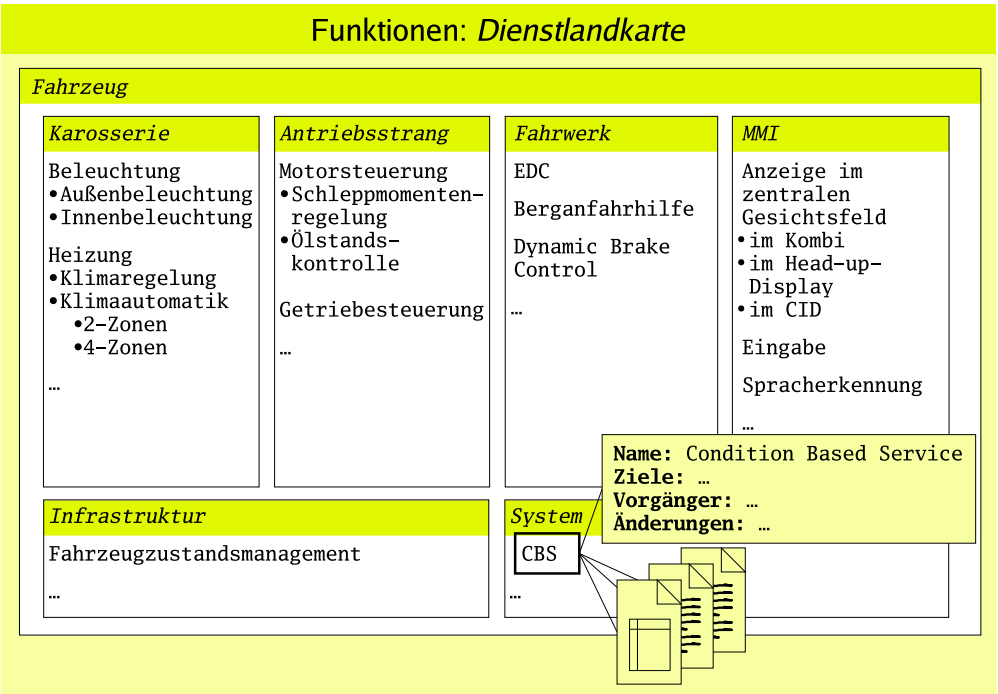
Aus methodischer Sicht ist es notwendig, die (anderweitig) erfassten (textuellen) funktionalen Anforderungen zu analysieren und zu konsolidieren. Dazu dienen *Dienste*, die die von »außen« sichtbare Funktionalität der zu entwickelnden E/E-Systeme beschreiben. Dabei wird zunächst nur die (idealisierte) Kernfunktionalität beschrieben. Die Vervollständigung um weitere Aspekte wie Fehlerbehandlung und Diagnosefähigkeit wird davon (im Sinne der Aspektorientierung) getrennt.

Die Erfassung der Dienste erfolgt in einer vorgegebenen, inhaltlichen und organisatorischen Struktur, wie sie heutzutage bereits gesetzt ist, zum Beispiel durch KIFA, AUTOSAR-Bereiche oder BMW-Funktionsordnungsstruktur. Die übergeordnete Struktur entspricht der Systemebene; auf der »Komponentenebene« erfolgt die Anforderungserfassung für die bereichsspezifischen Dienste durch die Stakeholder dieses Bereichs. Siehe zur Illustration das Schema zur Dienstlandkarte in Abbildung 4.

Dienste werden zu Beginn informell, in geeigneten, nicht eingeschränkten Formaten beschrieben, zum Beispiel mit freien Texten oder textuellen Beschreibungen in DOORS, mit Hilfe von Anwendungsfällen (Use Cases), durch Regelungskreise oder graphischen Skizzen. Die Heterogenität der Beschreibungen in der Gesamtstruktur ist der Grund für die Bezeichnung »Dienstlandkarte«. Die heterogenen Informationen werden konzeptionell an den entsprechenden Knoten in der hierarchisch aufgebauten Dienstlandkarte angehängt.

Ein Dienst ist eine Funktion eines Fahrzeugs, die dem Nutzer angeboten wird, zum Beispiel `CBS`, `Klimaautomatik` in den Varianten `2-Zonen-` und `4-Zonen-Klimaautomatik`, `Motorsteuerung` mit den Teildiensten `Schleppmomentenregelung` und `Ölstandskontrolle`, `Elektronische Dämpfercontrol (EDC)` und verschiedene Möglichkeiten für eine `Anzeige im zentralen Gesichtsfeld` des Fahrers. Unter einem Nutzer werden dabei Zielgruppen verstanden, die die Funktionalität nutzen oder davon in irgendeiner Form betroffen sind, zum Beispiel `Fahrer`, `Beifahrer` und `Service & Werkstätten`.

Abbildung 4 Dienstlandkarte (Schema).



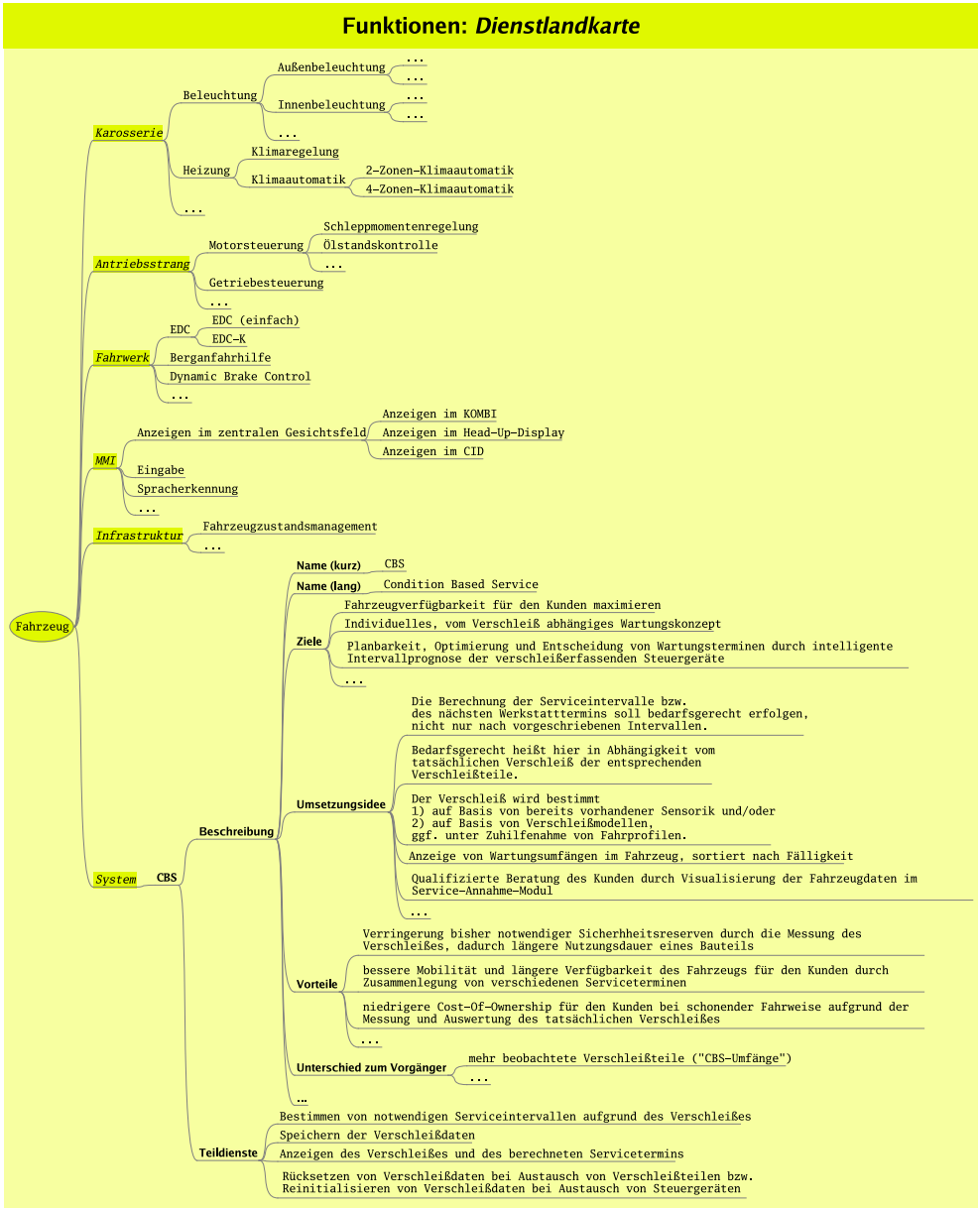
Dienste werden hierarchisch angeordnet, wobei unterschiedliche Hierarchiebeziehungen zu unterscheiden sind, zum Beispiel:

- Funktionale Verfeinerung von Diensten in Teildienste, zum Beispiel: `Beleuchtung` in die Teildienste `Außenbeleuchtung` und `Innenbeleuchtung`,
- Alternative Dienste, zum Beispiel gibt es für den Dienst `Klimaautomatik` die Alternativen `2-Zonen` und `4-Zonen`.

Aufgabe im weiteren Projektverlauf wird es sein, diese und weitere mögliche und sinnvolle Hierarchiebeziehungen in der Dienstlandkarte zu identifizieren und methodisch zu unterstützen. In Abbildung 5 sind beispielhaft hierarchisch angeordnete Dienste als Mindmap dargestellt, ohne jedoch eine Unterscheidung der Hierarchiebeziehungen explizit in dieser Illustration vorzunehmen.

Ein Dienst (d. h. jeder Knoten in der Diensthierarchie) muss genauer beschrieben werden. Hierzu eignet sich als erstes ein Beschreibungsschema, das auch informelle Anteile erfassen und verwalten lässt. Im weiteren Verlauf werden diese Anteile durch formalere Beschreibungen (gemäß Dienstmodell, vgl. [GHHJ06], siehe auch [BS01, SS03, KSTW04, Bro05, KMM05]) ergänzt oder auch unter Umständen

Abbildung 5 Dienstlandkarte (Illustration als Mindmap).



ersetzt. Beispielsweise gehören zu einem Beschreibungsschema die üblichen Kurz- und Langnamen eines Dienstes, die Begründung für den Dienst (das Ziel, das mit der Realisierung dieses Dienstes verfolgt wird) sowie eine kurze (textuelle) Beschreibung. Weitere Angaben sind in jeglicher Form möglich.

Ziel der Beschreibung von Diensten in der Dienstlandkarte ist deren formalisierte Beschreibung als (partielle) Funktionen, d. h. Funktionen, die noch nicht unbedingt für alle möglichen Eingaben spezifiziert sind. Vorteil einer solchen formaleren Beschreibung ist die bessere (werkzeuggestützte) Analysemöglichkeit zur Bewertung der Anforderungen, zum Beispiel, ob und wie Dienste sich gegenseitig beeinflussen, ob Widersprüche in den Dienstbeschreibungen existieren, ob der volle Dienstumfang sich kostengünstig und mit der vorgesehenen Infrastruktur realisieren lässt.

Die formale Dienstbeschreibung umfasst die Spezifikation aller Ein- und Ausgänge des Dienstes sowie die (partielle) Beschreibung, wie aus den Eingangswerten (Sensorik) die Ausgangswerte (Anweisungen für die Aktorik) erzeugt werden. Die Dienstbeschreibung kann, wie bereits gesagt, mittels Unterdienste verfeinert werden, deren Ein- und Ausgänge, im Unterschied zu den Funktionen im Funktionsnetz, jedoch immer noch auf Systemebene liegen. In diesem Sinne ist die Dienstbeschreibung eine formalisierte Form von Anwendungsfallbeschreibungen (gemäß UML-Use Cases). Der Begriff der Dienste und deren Formalisierung wird detailliert in [GHHJ06] beschrieben.

Die Dienstlandkarte dient der Definition von Produkten, d. h. der Erfassung von funktionalen Anforderungen, und nicht der Spezifikation ihrer Realisierung, d. h. Implementierungsdetails, die über die allgemeine Dienstbeschreibung hinausgehen, sollten hier nicht erfasst werden. Die Dienstlandkarte ist »vollständig«, wenn »alle« Dienste durch eine Dienstbeschreibung definiert wurden und alle relevanten Informationen an die Strukturknoten bzw. an die Dienste annotiert wurden. Konflikte sollten gelöst sein, Widersprüche nicht mehr vorkommen.

2.4 Funktionsnetz

Zur Realisierung der Dienste werden Funktionen entworfen, aus denen die Dienste zusammengesetzt werden können. Mit dem Entwurf des Funktionsnetzes werden gemeinsame Bestandteile von Diensten identifiziert und Beziehungen zwischen den Diensten realisiert.

Funktionen beschreiben Informationsflüsse und Verhalten, aber keine software- oder hardwaretechnischen Realisierungen. Im Gegensatz zu Diensten repräsen-

tieren sie jedoch auch systeminterne Funktionen und Informationsflüsse. Das Funktionsnetz auf der Systemebene beschreibt die logischen Schnittstellen der Funktionen und die Informationsflüsse zwischen den Funktionen. Es dient damit der virtuellen Integration, d. h. der Absicherung der Integrationsfähigkeit der zu implementierenden Komponenten sowie der systemweiten Optimierung.

Einzelbereiche bzw. Einzelfunktionen im Funktionsnetz werden durch Funktionsmodelle repräsentiert, die von den jeweiligen Fachstellen mit der entsprechenden Fachkompetenz entwickelt werden. Die Abstimmung der durch das Funktionsnetz vorgegebenen Funktionsschnittstellen und der sich aus der Funktionsmodellierung ergebenden Funktionsschnittstellen erfolgt iterativ.

Ein Funktionsmodell beschreibt eine Funktion in Struktur und Verhalten, jedes Funktionsmodell kann wiederum hierarchisch aus weiteren Funktionen aufgebaut sein, wie es bereits in MOSES [Fra03a, Fra05a, Kle06] beschrieben wurde.

Für jede Funktion werden mit Hilfe von Eingangs- und Ausgangsports die Schnittstellen in Form von Signalen (bzw. Operationssignaturen) angegeben. Mit Hilfe der Verhaltensmodellierung lässt sich die Ein-/Ausgaberelation zwischen Funktionseingang und Funktionsausgang spezifizieren und (idealerweise) auch simulieren. Die Simulation kann stufenweise durchgeführt werden: sind »alle« Funktionsmodelle simulierbar, so kann auch das Funktionsnetz simuliert werden. Darauf aufbauend können zur Bewertung der Funktionsmodelle bzw. des Funktionsnetzes beispielsweise Signalflussanalysen oder Fehlerbetrachtungen vorgenommen werden.

2.5 Softwarearchitektur

Mit der Softwarearchitektur wird die Umsetzung des Funktionsnetzes mit seinen Funktionen durch Softwarekomponenten beschrieben. Auf oberster Ebene wird die systemweite Softwarearchitektur definiert. Die systemweite Softwarearchitektur ist das Modell aller Softwarekomponenten und ihrer Beziehungen. Architekturstile, die hier Anwendung finden, sind beispielweise unter anderem »Master-Slave«- und »Client-Server«-Beziehungen zwischen Softwarekomponenten.

Aus Funktionsmodellen werden die Beschreibungen der Softwarekomponenten abgeleitet und in einem Softwarekomponentenmodell festgehalten. Dazu werden für die Implementierung notwendige Details zu Struktur und Verhalten ergänzt. In AUTOSAR entspricht dies der Befüllung des *Software Component Description Template*. Die Gruppierung von Funktionen zu Softwarekomponenten erfolgt im Allgemeinen gemäß anderer Kriterien als bei der funktionalen Dekomposition

in Funktionsmodellen, so dass die hierarchische Strukturierung von Softwarekomponenten nicht kongruent mit der im Funktionsnetz sein muss.

Die Beziehungen zwischen Softwarekomponenten (Verwendungsbeziehungen, zum Beispiel Prozeduraufrufe, gemeinsame Nutzung von globalen Variablen etc.) wird mit Hilfe von Softwarearchitekturbeschreibungen festgehalten. Die Softwarearchitektur dient damit – ähnlich wie das Funktionsnetz für Funktionen – der Integration, d. h. Softwarekomponentenspezifikationen oder ihre Implementierungen können dahingehend überprüft werden, ob sie zu den Vorgaben der Softwarearchitektur kompatibel sind.

Die Softwarearchitektur beschreibt die Struktur der Applikationssoftware. Die Basissoftware gehört zu einem Steuergerät und wird damit in der technischen Architektur berücksichtigt. Ziel ist es, die Softwarearchitektur der Applikationssoftware unabhängig von der technischen Architektur zu entwerfen. In manchen Fällen wird das aufgrund spezieller Hardwareanforderungen nicht möglich sein.

Eine Softwarekomponente ist die Bezeichnung für die kleinste Einheit, in der Software ausgeliefert (*deployed*) wird. Auf einem Steuergerät können mehrere Softwarekomponenten installiert sein. Eine Softwarekomponente kann aber nicht über mehrere Steuergeräte verteilt installiert werden. Jedoch kann eine Softwarekomponente mehrmals (gleichzeitig auf einem oder mehreren Steuergeräten) installiert sein. Die Allokation einer Softwarekomponente auf mehrere Steuergeräte ist damit als mehrfache Instanziierung dieser Komponente zu verstehen.

2.6 Technische Architektur

Mit Hilfe von technischen Architekturmodellen (Hardwaretopologiemodellen und Steuergerätemodellen) werden die Technologieentscheidungen für die Kommunikationsbusse und andere Signalträger inklusive der Zuordnung der Steuergeräte sowie Entscheidungen zur strukturellen Vernetzung der Kommunikationsnetze und Gatewaytechnologien spezifiziert.

Ein Steuergerätemodell beschreibt dabei ein Steuergerät, wobei es sich auch um ein virtuelles Steuergerät handeln kann, d. h. ein Steuergerätemodell kapselt ein Subsystem. Für die Beschreibung wird grundsätzlich auf die Möglichkeiten des AUTOSAR *ECU Resource Templates* zurückgegriffen, es werden also die technischen Ressourcen des Steuergeräte, wie potentielle Rechenleistung, Speicherplatz oder Peripherieanbindung beschrieben.

Die Steuergerätemodelle müssen den Anforderungen der technischen Systemar-

chitektur genügen, das heißt für die Entwicklung der Steuergeräte werden deren Schnittstellen und innere Struktur soweit spezifiziert, wie es für die Integration in den Gesamtsystementwurf notwendig ist. Wie beim Entwurf von Funktionsnetzen und Funktionen erfolgt die Schnittstellenbeschreibung der Steuergeräte iterativ mit der Spezifikation der technischen Architektur (Festlegung von Busschnittstellen etc.). Die Modellierung der inneren Struktur (z. B. Festlegung von Prozessoren, Speicher etc.) dient sowohl der Abschätzung von Leistung und Kosten etc. als auch der Softwareplanung.

Im AUTOSAR-Kontext gehört die Basissoftware zu den Ressourcen; sie wird gemäß der Anwendungsanforderungen und der Verteilung der Anwendungssoftwarekomponenten generiert. Insofern kann die Beschreibung (Benennung und Konfiguration) der Basissoftware der Steuergerätebeschreibung zugeordnet werden. Das Ziel von AUTOSAR ist es, die Basissoftware jeweils generieren zu können.

Der VEIA-Referenzprozess unterstützt den Entwurf der Hardwaretopologie bereits in der Anfangsphase der System- bzw. Produktlinienentwicklung und entkoppelt ihn damit teilweise von der Funktionsentwicklung. Architekturbewertungen, die unabhängig von der tatsächlichen Ausnutzung der Infrastruktur sind, wie statistische Durchlaufzeiten von Signalen etc., können damit bereits frühzeitig durchgeführt werden. Für funktionsabhängige Bewertungen – wie Ausführungszeiten bspw. mit HW/SW-Co-Simulation oder Kostenbetrachtungen – werden Verteilungsmodelle herangezogen, die in unterschiedlichen Abstraktionsstufen entwickelt werden.

Die Verteilung der Funktionen auf die Hardwareressourcen wird durch die entsprechenden Verteilungsmodelle repräsentiert (Abschnitt 2.7.3). Prinzipielle Modellierungskonzepte für technische Architekturen sowie Verteilungsmodelle wurden bereits in MOSES [Fra03b, Fra04a, Fra04b, Fra05b, Fra05a, Kle06] betrachtet.

2.7 Artefaktbeziehungen

Wie bereits diskutiert, bestehen zwischen den Entwicklungsartefakten des VEIA-Referenzprozesses kausale Abhängigkeiten. Diese lassen sich wie folgt unterteilen:

- Verfeinerungs- bzw. Realisierungsbeziehungen,
- Konfigurationsbeziehungen und
- Verteilungsbeziehungen.

Sie ergeben sich aus dem Zweck, den einzelne Artefakte im Entwicklungsprozess zu erfüllen haben. Durch Formalisierung und Ausnutzung solcher Beziehungen

kann geprüft werden, inwieweit Inkonsistenzen zwischen den einzelnen Artefakten bestehen oder ob zu betrachtende Systemaspekte bislang »vergessen« wurden. Die Ausnutzung dieser Beziehungen stellt ein wichtiges Hilfsmittel in der Sicherstellung der (virtuellen) Integrationsfähigkeit dar und dient zur frühen Absicherung von Test- und Integrationsaktivitäten im rechten Ast des V-Modells.

Verfeinerungs- bzw. Realisierungsbeziehungen stellen eine Form von vertikalen Relationen dar, da sie Artefakte unterschiedlicher Abstraktionsebenen bzw. unterschiedlicher Entwicklungsstufen miteinander in Beziehung setzen. Dagegen sind Konfigurations- und Verteilungsbeziehungen horizontale Relationen, hier werden Artefakte unterschiedlicher Systemsichten auf potentiell gleicher Abstraktionsebene oder Entwicklungsstufe verbunden.

In Abbildung 1 sind die Beziehungen aufgeführt: die Verfeinerungsbeziehungen zwischen den Modellen des Funktionsbereichs; die Konfigurationsbeziehungen zwischen der Produktlinienbeschreibung und den anderen Modellen und schließlich die Verteilungsbeziehungen zwischen den Funktions-Artefakten und der technischen Architektur.

2.7.1 Verfeinerungs- und Realisierungsbeziehungen

Zwischen Dienstlandkarte und Funktionsnetz sowie zwischen Funktionsnetz und Softwarearchitektur bestehen potentiell n:m-Relationen. Diese Relationen verweisen darauf, welche Dienste durch welche Funktionen »realisiert« werden. Analog dazu sind die Relationen zwischen Funktionen und Softwarekomponenten zu sehen. Das Verhältnis »n:m« hat seine Ursache darin, dass die Strukturierung und Dekomposition von Diensten, von Funktionen und von Softwarekomponenten unterschiedlichen Kriterien genügen muss. Diese Kriterien sowie ihre Beziehungen werden in den weiteren VEIA-Arbeitspaketen untersucht und ihre Ausnutzung methodisch unterstützt.

2.7.2 Konfigurationsbeziehungen

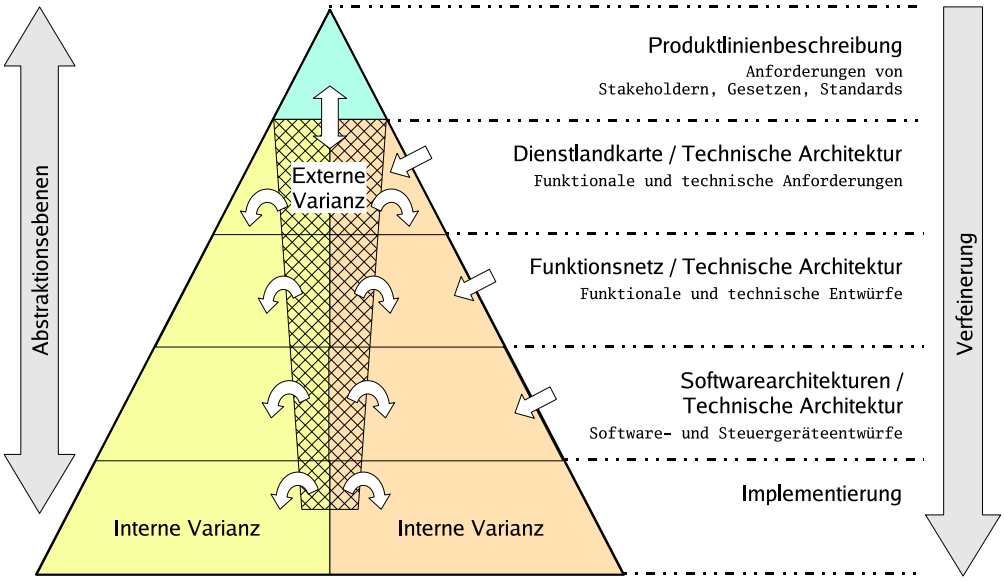
Neben der Erfassung und Beschreibung von Varianz in Produktlinienmodellen ist es wichtig, Varianz, die inhärent in den jeweiligen Artefakten repräsentiert ist, wieder »aufzulösen«: um einerseits anzugeben, wann und in welchen Situationen welche Variante gewählt wird, und um andererseits (darauf aufbauend) die einzelnen Produkte (bzw. Produktmodelle) ableiten zu können. Die Konfiguration (Annotation)

variierender Artefakte dient diesem Zweck.

Die äußere (externe) Varianz wird in der Produktlinienbeschreibung erfasst. Die Beziehungen zwischen der Produktlinienbeschreibung und den anderen Entwicklungsartefakten sind durch entsprechende Konfigurationsmodelle zu erfassen. So muss z. B. geklärt sein, welche der Merkmalsvarianten in der Produktlinienbeschreibung durch welche Funktionsvarianten im Funktionsnetz »realisiert« bzw. reflektiert wird.

Konfigurationsbeziehungen müssen keine 1:1-Beziehungen sein, sondern können auch komplexer aufgebaut sein. So können auch Merkmalskombinationen für eine Funktionsvariante verantwortlich sein. Ein erstes Konfigurationsmodell wurde bereits in MOSES angegeben, vgl. [Fra04b, Fra05b, Fra05a, Kle06].

Abbildung 6 Variabilitätspyramide nach [PBL05].



In Abbildung 6 werden die prinzipiellen Beziehungen zwischen Varianz auf den unterschiedlichen Abstraktionsebenen dargestellt. Die Abbildung ist den Arbeiten von [PBL05, S. 71f.] entlehnt und wurde hier in Bezug auf die in diesem Dokument vorgeschlagenen Entwicklungsartefakte und Abstraktionsebenen angepasst. Die Abbildung zeigt, dass von höheren zu niedrigeren Abstraktionsebenen die Komplexität von Varianz zunimmt, weil einerseits der Verfeinerungs- und Detaillierungsgrad zunimmt (eine untere Abstraktionsebene »reflektiert« die in der höheren Abstraktionsebene eingeführte Varianz) und weil andererseits auch auf jeder Abstraktionsebene zusätzliche »interne« Varianz hinzukommt (hereinkom-

mende äußere Pfeile), da auf der jeweiligen Abstraktionsebene weitere Aspekte zusätzlich mit betrachtet werden müssen.

Die in der Produktlinienbeschreibung definierte, nach außen sichtbar gemachte Varianz (externe Varianz) betrifft im allgemeinen mehrere Abstraktionsebenen: es werden Vorgaben an Funktionalität, deren Softwareumsetzung und auch an die technische Architektur und ihre Realisierung gestellt.

2.7.3 Verteilungsbeziehungen

Funktionen (bzw. deren Umsetzung in Software) müssen auf der Infrastruktur ausgeführt werden können. Die Infrastruktureelemente stellen dabei einschränkende Ressourcen dar: wenn beispielsweise keine Kommunikationsverbindung auf Hardwareebene zwischen Steuergeräten vorgesehen wurde, kann keine Kommunikation stattfinden, auch wenn die Funktionen, die auf die nicht verbundenen Steuergeräte verteilt wurden, miteinander kommunizieren sollen.

Verteilungsbeziehungen bestehen zwischen Dienstlandkarte und technischer Architektur auf der obersten Abstraktionsebene, als Verfeinerungen dazu zwischen Funktionsnetz und technischer Architektur sowie zwischen Software und technischer Architektur. Die Verteilung von Software auf die technische Architektur wird auch Softwareallokation genannt.

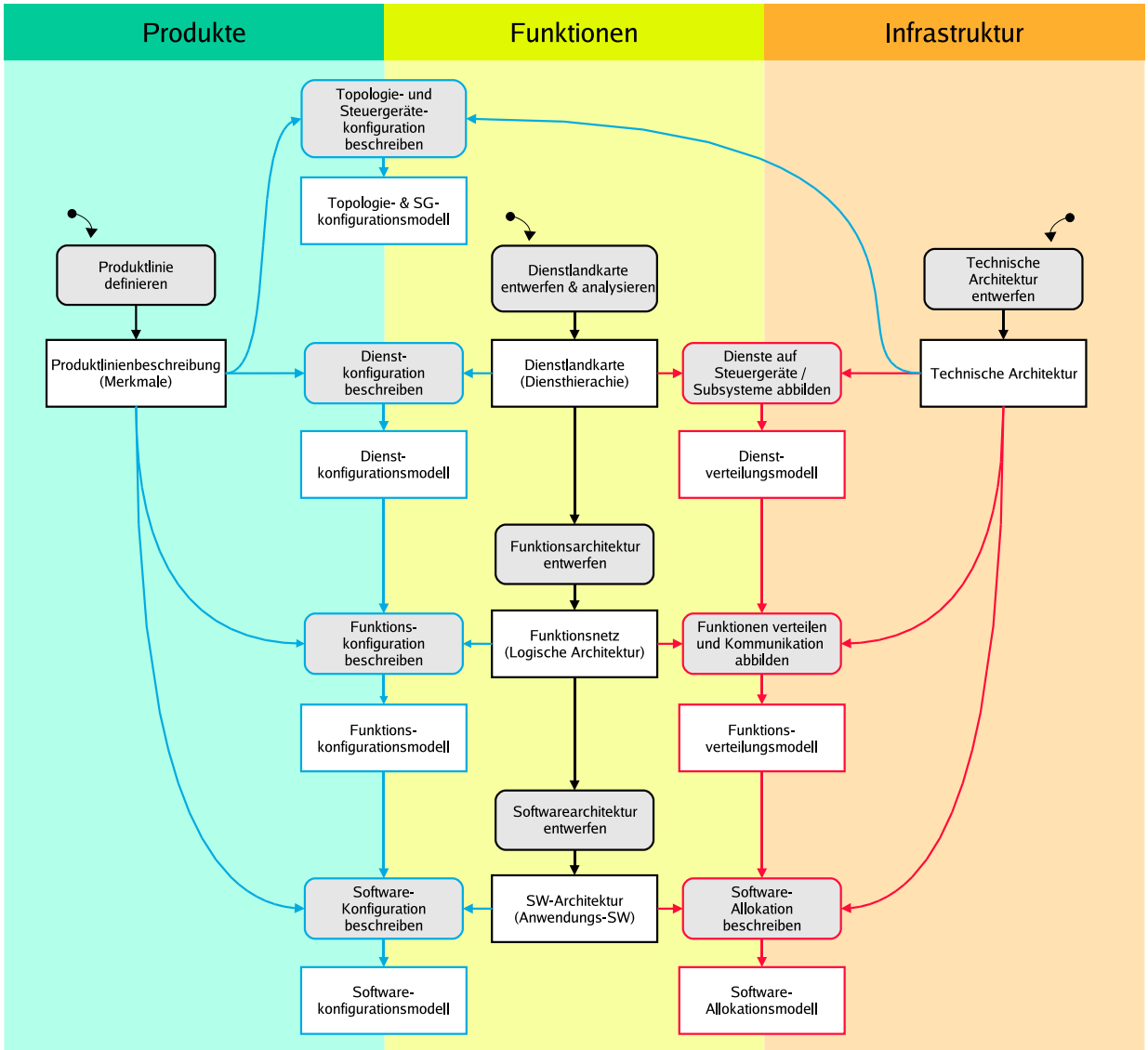
Prinzipielle Beziehungen zwischen Funktionsnetz, Software und technischer Architektur, auch unter Betrachtung von Produktlinienaspekten, wurden bereits in MOSES definiert, vgl. [Fra04b, Fra05b, Fra05a, Kle06], dort unter dem Begriff »Partitionierung«.

2.8 Aktivitäten zur Erstellung, Bewertung und Pflege der Entwicklungsartefakte

Die zuvor beschriebenen Entwicklungsartefakte müssen methodisch unterstützt erstellt und im Sinne der Qualitätssicherstellung im Entwicklungsprozess bewertet und gepflegt werden.

In Abbildung 7 wird das Schema aus Abbildung 1 um entsprechende, grundlegende Aktivitäten zur Erstellung der Artefakte unter Berücksichtigung eines methodischen Vorgehens und der kausalen Abhängigkeiten erweitert. Die Aktivitäten werden durch graue Kästen mit abgerundeten Ecken dargestellt.

Abbildung 7 Artefakte: Kausale Zusammenhänge und Aktivitäten zur Erstellung der Entwicklungsartefakte.



2.8.1 Vorbemerkungen

Es ist zu beachten, dass an dieser Stelle zwar eine kausale Reihenfolge der Artefakte vorgegeben wird, diese aber so zu verstehen ist, dass ein abhängiges Artefakt erst vollständig fertiggestellt werden kann, wenn dessen kausale Vorgänger fertig sind. Ob Zwischenversionen erstellt werden, um den Prozess zu linearisieren oder überhaupt handhabbar zu gestalten, ist Aufgabe der Prozessphasenbeschreibung von Kapitel 3.

Bei der Beschreibung der Artefakte wird nur auf die Artefakte auf Systemebene (vgl. Abbildung 2) eingegangen. Das ist als Leseerleichterung zu verstehen. Es ist klar, dass z. B. die Erstellung der Dienstlandkarte so abläuft, dass Dienstbeschreibungen entworfen werden, in die Landkarte integriert werden, weitere Beschreibungen folgen, die wiederum integriert werden müssen, usw. usf. Dabei werden getroffene Entscheidungen revidiert bzw. angepasst, bis die Dienstlandkarte als Artefakt feststeht. Diese Iterationen werden nicht weiter beschrieben.

Ebenso klar ist, dass Änderungen an abhängigen Artefakten auch ihre kausalen Vorgänger betreffen können. Diese Normalität der Entwicklung wird nicht besonders berücksichtigt, sondern als Aufgabe der Modellverwaltung und des Versionierens der Modelle, also des Änderungsmanagements, angesehen. Der Grund für diese und die vorige Vereinfachung ist, dass sonst zwischen allen Artefakten Rückwirkungen eingezeichnet werden müssten, was die Lesbarkeit verringert und das Verständnis erschwert, ohne wichtige zusätzliche Informationen zu liefern.

Eine weitere Einschränkung ist, dass die Wiederverwendung (*Carry Over*) noch nicht thematisiert wird. Es ist klar, dass alle vorhandenen Artefakte wiederverwendet werden, wenn sich das anbietet. Inwiefern die Artefakte systematisch aufgehoben werden (Domänenansatz von MOSES), wurde in MOSES geklärt oder muss noch präzisiert werden. Diese Beschreibung wäre allerdings damit überfrachtet.

Die Artefakte wurden so gewählt, dass jedes Artefakt andere Informationen als das andere trägt, die Modelle also nicht redundant gehalten werden. Bestimmte Informationen werden innerhalb des Prozesses redundant auftreten, z. B. enthalten *Use-Cases* der Dienstlandkarte gleichartige Informationen wie Dienstbeschreibungen, nur in unterschiedlichem Formalisierungsgrad. Wie mit dieser Redundanz umgegangen wird, ist Aufgabe der (noch zu beschreibenden) Modellverwaltung, also prinzipiell eine Werkzeugfrage.

2.8.2 Prinzipielles Vorgehen

Schritt 1: Produktlinienbeschreibung, Dienstlandkarte, technische Architektur

Der Ausgangspunkt der Modellierung ist die Dienstlandkarte. Sie ist die Referenz für alle folgenden Modelle. Sie wird aus den Anforderungen aller Stakeholder an das Fahrzeug erstellt. Aus diesen Anforderungen werden die Use-Cases eines Dienstes erhoben. Damit können die Schnittstellen des Dienstes sowie sein Verhalten beschrieben werden. Die Formalisierung der Use-Cases ergibt die Dienstbeschreibungen. Die Dienste werden integriert und abgestimmt. Die Dienstlandkarte enthält die Varianz des Fahrzeugprojekts auf Dienstebene.

Während der Erstellung der Dienstlandkarte werden das Fahrzeug, die Fahrzeugproduktlinie oder die Baureihe detailliert spezifiziert. Es wird festgelegt, welche Fahrzeuge gebaut werden, welche Ausstattung angeboten wird etc. Diese Informationen werden in der Dienstlandkarte sowie in der Baureihenbeschreibung bzw. dem Featuremodell hinterlegt. Dienstlandkarte und Featuremodell werden durch das Konfigurationsmodell verbunden. Die Baureihenbeschreibung bzw. das Featuremodell können erst nach der Dienstlandkarte fertiggestellt werden, da die Dienstlandkarte die Referenzbeschreibung des Systems ist.

Ebenfalls parallel zur Dienstlandkarte wird die Topologie entworfen. Es wird festgelegt, wieviel und welche Steuergeräte verbaut werden, danach wird entschieden, wie diese Steuergeräte vernetzt werden. So können sehr früh Anforderungen an Flashzeiten, Buslast, Kommunikationsgeschwindigkeit etc. berücksichtigt werden. Die Dienste der Dienstlandkarte werden auf die Topologie verteilt, so wird festgelegt, welches Steuergerät von welchem Dienst betroffen ist.

Mit der Erstellung von Topologie und Featuremodell wird auch die Beziehung zwischen diesen beiden Modellen im Topologie- und Steuergerätekonfigurationsmodell beschrieben.

Schritt 2: Funktionsnetz

Jetzt kann die Funktionsarchitektur entworfen werden. Dazu werden die Dienste genommen, analysiert und als Funktionen modelliert. Die Funktionen werden dann verfeinert, umstrukturiert und nach logischen Gesichtspunkten zusammengefasst. Die Funktionen werden dabei immer mit dem gesamten Funktionsnetz synchronisiert, so dass ein simulationsfähiges Funktionsnetz entsteht. Die Varianz wird aus den Diensten übernommen und angepasst, unter Umständen wird sie

erweitert oder eingeschränkt, da im Funktionsnetz durchaus Varianz anders als in der Dienstlandkarte bestehen kann.

Die Varianz wird über die Funktionskonfiguration mit dem Featuremodell in Beziehung gesetzt.

Ausgehend von der Dienstverteilung werden auch die Funktionen auf die Topologie verteilt. Die Verteilung der Funktionen darf die Verteilung der zugehörigen Dienste nicht verletzen, also nicht erweitern. Das heißt, eine Funktion, die einen Dienst realisiert, darf nur auf ein Steuergerät verteilt werden, auf das auch der Dienst verteilt wurde.

Schritt 3: Softwarearchitektur

Mit Fertigstellung des Funktionsnetzes kann die Softwarearchitektur fertiggestellt werden. Dazu werden die Funktionsmodelle in Softwarekomponenten umgewandelt. Logische Verbindungen müssen betrachtet werden, Signale bzw. Operationen müssen in Software überführt werden. Dabei ist die Softwarearchitektur so zu entwerfen, dass die Softwarekomponenten nicht zu groß werden, um alloziert werden zu können aber auch nicht zu klein, um den Allokationsaufwand und damit den Flashaufwand unnötig zu erhöhen.

Prinzipiell beschreiben die Softwarekomponenten nur Applikationssoftware unabhängig von der Topologie. Wie bereits oben geschildert wurde, ist das manchmal nicht durchzuhalten, z. B. wenn Nachrichten aufgeteilt werden oder die Basissoftware ein bestimmtes Softwaredesign erfordert. In diesem Fällen ist die Allokation durchzuführen, um die Software entwerfen zu können. Ziel sollte sein, diese Fälle auf ein Minimum zu reduzieren, um die Wiederverwendung zu erhöhen. In der Praxis können Softwarekomponenten durch den Einsatz anpassbarer Code-Generatoren unabhängig von der Basissoftware beschrieben werden.

Die Varianz des Funktionsnetzes ist der Ausgangspunkt für die Varianz der Softwarekomponenten, auch hier sind wieder Modifikationen möglich. Damit muss auch die Beziehung zwischen Softwarekomponenten und Features durch die Softwarekonfiguration festgehalten werden.

Den Abschluss bildet die Allokation der Softwarekomponenten auf die Topologie. Das Allokationsmodell gibt an, welche Softwarekomponente auf welchem Steuergerät läuft. Dabei sind der Nachrichtenkatalog der Topologie sowie die Basissoftware zu berücksichtigen. Auch das Allokationsmodell darf die Grenzen des Funktionsverteilungsmodells nicht überschreiten, es gelten analoge Einschränkungen wie bei der Verteilung von Funktionen. Zusätzlich dürfen Softwarekomponenten nur auf ein Steuergerät alloziert werden.

Mit Erstellung der Softwarearchitektur und des Allokationsmodells kann auch die Topologie fertiggestellt werden, die bis zu diesem Zeitpunkt wegen der Abhängigkeit von der Softwarearchitektur noch offen gehalten werden musste. Im Sinne des HW/SW-Co-Design, das im Hintergrund der Überlegungen steht, beeinflussen Änderungen der Softwarearchitektur auch die zugrundeliegende Hardware.

Damit sind alle Modelle fertiggestellt, die Modellierung ist somit abgeschlossen. Die erstellten Modelle werden nun genutzt, um die Implementierung und Integration zu treiben sowie um Testfälle und Testszenarios zu generieren.

2.9 Modellierungstechniken

Wesentliche Modellierungstechniken, die zur Beschreibung der Artefakte des Referenzprozesses verwendet werden können, wurden bereits im Projekt Modellbasierte Systementwicklung (MOSES) betrachtet [Fra03a, Fra03b, Fra04a, Fra04b, Fra05b, Kle06]. Hier werden prinzipiell Anforderungen, logische Architekturen, Hard- und Softwarearchitekturen sowie die Verteilung der logischen Architektur auf Software- und Hardwarearchitekturen unterschieden. Außerdem werden in MOSES bereits die Modellierung von Produktlinien und Domänenmodellen unterstützt und Basiskonzepte für verteiltes Modellieren sowie für Mehrfach- bzw. Wiederverwendung von Teilmodellen bereitgestellt.

An der TU München (Lehrstuhl für Software & Systems Engineering) wurde das Konzept der Dienste als Formalisierung von Anwendungsfallmodellierung (Use-Case-Modellierung) bzw. der Modellierung von nutzersichtbarer Funktionalität untersucht, vgl. [BS01, SS03, KSTW04, Bro05, KMM05]. Im Rahmen von VEIA wird das Konzept auf die Produktlinienentwicklung von Automotive-E/E-Systemen angepasst werden, siehe [GHHJ06].

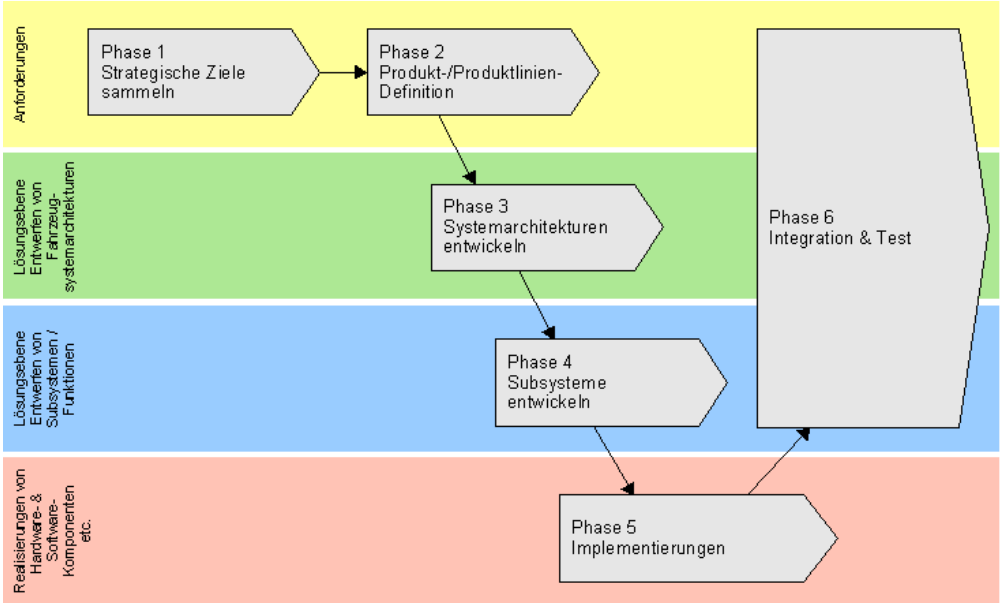
AUTOSAR stellt u. a. Beschreibungstechniken für Softwarearchitekturen, -komponenten sowie technische Architekturen (Hardwaretopologien und Steuergerätebeschreibungen) zur Verfügung. Jedoch wird der Produktlinienansatz nicht explizit unterstützt, so dass hier entsprechende Anpassungen notwendig sein werden.

Die Notationen der UML werden an geeigneter Stelle gemäß der aus den Artefakten abzuleitenden Erfordernisse eingesetzt: Verhaltensmodelle wie Statecharts oder Aktivitätsdiagramme können an entsprechender Stelle leicht eingebettet werden. Klassendiagramme sind geeignete Mittel, um die Realisierung von Softwarekomponenten zu beschreiben. Für Dienst- und Funktionsmodellierung sowie für Architekturbeschreibungen sind sie nicht geeignet.

3 Prozessphasen

Die in diesem Kapitel diskutierten Prozessphasen ordnen die Entwicklungsartefakte des VEIA-Referenzprozesses aus Kapitel 2 unter Berücksichtigung der kausalen Abhängigkeiten und aufgrund methodischer sowie organisatorischer Gesichtspunkte in einer (groben) zeitlichen Abfolge. Ein Überblick über die betrachteten Prozessphasen ist in Abbildung 8 dargestellt.

Abbildung 8 Die Phasen des VEIA-Referenzprozesses.



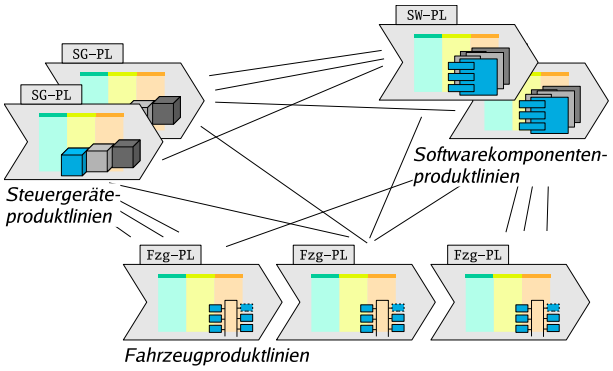
Der Referenzprozess wird zunächst für ein Fahrzeugprojekt / eine Fahrzeugproduktlinie angegeben, im Folgenden kurz *Projekt* genannt. Das Ziel eines solchen Projekts ist die Erstellung von *Produkten* (Fahrzeugen, Fahrzeugtypen) in einer *Produktlinie* (Fahrzeugprojekte).

Produktlinien können generell auf unterschiedlichen Ebenen betrachtet werden. Im Automotive-Bereich sind zumindest folgende Ebenen zu berücksichtigen (Abbildung 9):

– Fahrzeugproduktlinien für die Festlegung, welche Fahrzeugtypen auf eine gemeinsame Basis gestellt werden,

– Steuergeräteproduktlinien (Gleich- oder Ähnlichteile) – vor allem Steuergerätezulieferer arbeiten auf dieser Ebene – und
– Softwarekomponentenproduktlinien.

Abbildung 9 Produktlinien unterschiedlicher Ebenen und ihre Beziehungen.



In Abbildung 9 sind die Beziehungen zwischen diesen Produktlinien dargestellt. Softwarekomponentenproduktlinien können so aufgebaut werden, dass sie für unterschiedliche Fahrzeugproduktlinien verwendet werden können. Die Systemfunktion *CBS* (siehe Kapitel 4) zielt darauf ab, eine generische Lösung, womöglich als Softwarekomponentenproduktlinie, zu entwickeln, die für mehrere Fahrzeugproduktlinien einsetzbar ist. Steuergerätezulieferer haben analog dazu auch ein Interesse, dass sie mehrere Fahrzeugproduktlinien (auch unterschiedlicher OEMs) mit ihren Steuergeräteproduktlinien bedienen können.[2] Umgekehrt hat ein OEM das Interesse, Gleich- und Ähnlichteile zu verbauen, also auch Steuergeräteproduktlinien und Softwarekomponentenproduktlinien in seiner Fahrzeugproduktlinie auszunutzen.

Nachfolgend werden die Prozessphasen zur Entwicklung von Produktlinien im Automotive-E/E-Kontext einzeln erläutert, wobei jede Phase nach folgendem Schema skizziert wird:

– Kurzbeschreibung,
– Ziel,
– Nicht-Ziel,
– Artefakte und Konzepte sowie

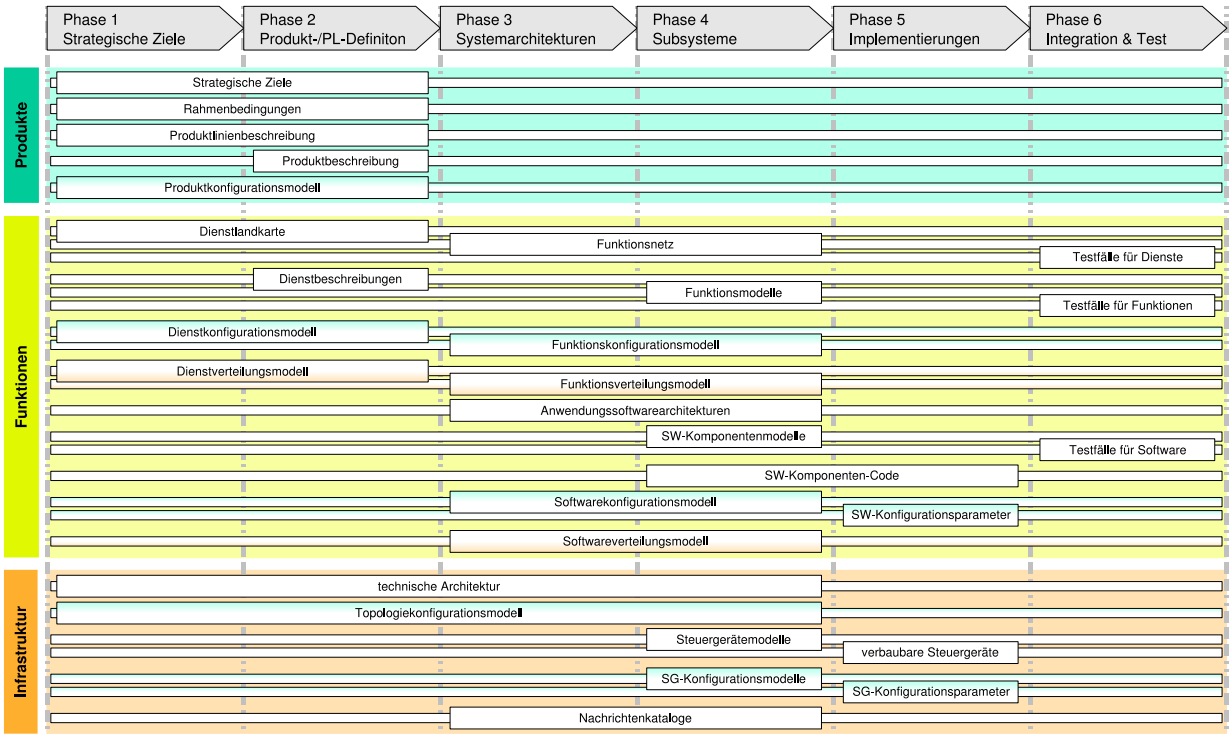
[2] Der Einsatz von Produktlinien bei Zulieferern wird in VEIA nicht untersucht.

– Beispiele.

Je nachdem, auf welcher Ebene eine Produktlinie entwickelt werden soll, stehen einzelne Entwicklungsartefakte und Prozessphasen stärker im Vordergrund als andere. Nichtsdestotrotz müssen Aspekte der zu unterstützenden Produktlinien auf den anderen Ebenen erfasst und berücksichtigt werden.

Abbildung 10 ordnet die Entwicklungsartefakte den im Folgenden beschriebenen Prozessphasen zu. Die Zuordnung gibt an, in welcher Phase primär an einem Artefakt gearbeitet wird, dann ist das Artefakt breiter dargestellt. Die dünnen Balken geben an, bis wann Artefakte ihre Gültigkeit behalten, in VEIA sind alle Artefakte bis zum Ende der Entwicklung gültig und können daher aktualisiert werden. Diese Aktualisierung, die bei Rückwirkungen von Änderungen anderer Artefakte in allen Phasen auftritt, wird, wie bei der Artefaktbeschreibung, aus Übersichtlichkeitsgründen nicht im Bild dargestellt, sie ist jedoch vorhanden und muss über entsprechende Änderungsmanagementprozesse gelebt werden.

Abbildung 10 VEIA-Referenzprozess: Zuordnung der Artefakte zu den Prozessphasen.



Ist die Zuordnung eines Artefakts über mehrere Phasen verteilt, so gibt es mehrere »Entwicklungsstufen« des Artefakts: beispielsweise dient die technische Architek-

tur in Phase 1 als strukturierte Sammlung der Anforderungen für die im Projekt umzusetzende technische Architektur, die in Phase 3 und 4 spezifiziert wird.

3.1 Phase 1: Strategische Ziele sammeln

Die erste Phase ist mit dem Projektstart verbunden. Es werden Vorgaben gesammelt, die die im neuen Projekt zu entwickelnden Produkte erfüllen soll(t)en. Die Vorgaben stammen von den Stakeholdern. Dabei kann auf alte Vorgaben zurückgegriffen werden, die um neue Ideen ergänzt werden.

Diese Phase repräsentiert sozusagen ein *Brainstorming* auf Projektebene. Die Ergebnisse dieser Phase werden in Phase 2 analysiert, auf Realisierbarkeit überprüft, dadurch konsolidiert und die Vorgaben werden festgemacht (*Commitment*).

Ziel

Das Ziel dieser Phase ist es, das Projekt auf einer organisatorisch und technisch groben Stufe zu umreißen. Die Funktionalität, die in die Produkte eingebaut werden soll, muss ebenso wie die organisatorischen und monetären Rahmenbedingungen geplant werden. Insbesondere werden die Produktvarianten skizziert, die als Fahrzeugproduktlinie gebaut werden sollen.

Nicht-Ziel

Es ist nicht Ziel, detaillierte technische Vorgaben zu machen oder ausgefeilte und »gute« Anforderungen (gemäß Anforderungsmodellierungsmethoden) zu erstellen.

Artefakte & Aktivitäten

Artefakte & Aktivitäten für die Phase sind in Tabelle 1 aufgelistet.

Tabelle 1 Übersicht von Artefakten und Aktivitäten in Phase 1 »Strategische Ziele sammeln«

Merkmal	Inhalte
Eingang	– Unternehmensziele – gesetzliche Vorgaben – ...

↪ ...

Merkmal	Inhalte
Verarbeitung	Beschreiben der grundlegenden Eigenschaften des Produkts (Produktvarianten, Funktionalität, technische Architektur) und der zugehörigen Rahmenbedingungen (z. B. Kosten, Werke, Personal)
Ausgang	– Strategische Ziele und Rahmenbedingungen für die Fahrzeugproduktlinie – Produktlinienbeschreibung (Merkmalsbeschreibungen) – Dienstlandkarte als Wiederverwendungsvorgabe bzw. »Wunschliste« der Stakeholder – initiale technische Architektur als Wiederverwendungsvorgabe bzw. »Wunschliste« der Stakeholder
Durchführende	Strategie, Vertrieb
Wiederverwendung	– bekannte und bereits in Vorgängerprodukten realisierte Funktionalität, – bekannte und bereits in Vorgängerprodukten verwendete technische Architekturen, – Erfahrungswerte aus Vorgängerprojekten (bzgl. Umsetzbarkeit, Kosten etc.)
Varianz	– Das Projekt wird soweit definiert, dass zu entwickelnde (Produkt-)Varianten skizziert werden: beispielsweise durch Varianzbetrachtungen in der Ausstattung (Produktliniemodell).

Beispiele

Ein am tatsächlichen Verschleiß ausgerichteter Service unter Ausnutzung vorhandener Sensorik ist zu realisieren. Zum Beispiel soll dem Fahrer angezeigt werden, wann der nächste Bremsen- oder Ölwechsel fällig wird. Damit soll die Nutzungsdauer des Fahrzeugs zwischen zwei Servicezeitpunkten auf das Fahrprofil hin optimiert werden.

Die Integration von handelsüblicher Consumer-Elektronik (iPod, MP3-Spieler, Laptop, Handy) soll im Sinne von »Best in Class« unterstützt werden.

3.2 Phase 2: Produktlinie und Produkte definieren

Die zweite Phase dient hauptsächlich dazu, die im Projekt zu entwickelnden Produkte bzw. die zu entwickelnde Fahrzeugproduktlinie detailliert zu definieren (Definition des *Scope* der Produktlinie, Definition aller Produktvarianten): Die initialen Anforderungen aus Phase 1 sind zu konsolidieren und die Dienstlandkarte ist so zu verfeinern, dass die zu implementierende Funktionalität in Schnittstelle und Verhalten als Black-Box genau beschrieben ist. Mit der technischen Architektur werden bereits die prinzipielle Busstruktur betrachtet und Flash- sowie Kommunikationsanforderungen erhoben.

Ziel

Das Ziel der Phase ist es, die initiale Dienstlandkarte aus Phase 1 zu vervollständigen (Sind alle Dienste erfasst?), Widersprüche zu eliminieren und Mehrdeutigkeiten aufzulösen. Die Dienste (Black-Box-Funktionen) sind vollständig zu beschreiben – in dem Sinne, dass sie so zu konkretisieren sind, dass der Entwurf des Funktionsnetzes (in Phase 3) beginnen kann. Dazu sind die Schnittstelle der Dienste und ihr wesentliches Verhalten (Idealverhalten) zu spezifizieren. Die Schnittstelle eines Dienstes umfasst die Eingänge (Sensorik, Sollwertgeber) und Ausgänge (Aktorik, Anzeigen, weitere Ausgaben) an der Systemgrenze. Eine vollständige Fehlerfallbetrachtung beispielsweise ist in dieser Phase noch nicht notwendig.

Im Bereich der Infrastrukturbeschreibung ist es das Ziel, eine Steuergerätetopologie vorzugeben, mit der die Funktionalität realisiert werden soll. Soweit sinnvoll und möglich, kann eine erste Verteilung der Dienste auf die technische Architektur erfolgen, um erste Abschätzungen durchzuführen, ob (auf Basis von Erfahrungswerten) die angedachte Funktionalität mit der angedachten Infrastruktur prinzipiell zusammenspielen kann.

Die zu entwickelnden Produktvarianten sind festzulegen und in der Produktlinienbeschreibung zu erfassen, d. h. es sind die Produktmerkmale (*Features*, z. B. Ausstattungsmerkmale, kundenwertige Merkmale) und ihre Zuordnung zu den Produkten der Produktlinie zu definieren. An dieser Stelle wird z. B. die Skalierung des Bordnetzes für die Produktvarianten festgelegt, dies führt zu einem frühen Steuergeräteproduktlinienmodell, das in der technischen Architektur abgebildet wird.

Nicht-Ziel

Es ist nicht Ziel, das interne Verhalten der Dienste (interne Funktionsdekomposition) zu modellieren. Das vollständige interne Verhalten steht nicht im Vordergrund, sondern nur neue oder wichtige Verhaltensweisen. Auch der Informationsfluss wird angerissen, aber nicht detailliert.

Ende

Die Phase ist beendet, wenn

- alle Produkte der Produktlinie festgelegt (benannt und in ihren Merkmalen beschrieben) wurden,
- alle Dienste benannt wurden,
- das Black-Box-Verhalten (Idealverhalten) aller Dienste beschrieben ist,

– die Rahmenbedingungen für die einzelnen Dienste festgelegt wurden,
– die Rahmenbedingungen für die technische Architektur definiert wurden.

Artefakte & Aktivitäten

Artefakte & Aktivitäten für die Phase sind in Tabelle 2 aufgelistet.

Tabelle 2 Übersicht von Artefakten und Aktivitäten in Phase 2 »Produktlinie und Produkte definieren«

Merkmal	Inhalte
Eingang	Sämtliche Ergebnisse aus Phase 1.
Verarbeitung	– Analyse der Dienstlandkarte (aus Phase 1) und der Rahmenbedingungen in Bezug auf Machbarkeit, Wirtschaftlichkeit, Konflikte etc. – Überarbeitung, Konsolidierung und Konkretisierung der initialen Anforderungen (Dienstlandkarte mit Rahmenbedingungen). – Modellierung der Dienste im Detail soweit, dass deren Black-Box-Verhalten klar ist. Das Verhalten muss nicht unbedingt vollständig sein (bzgl. Fehlerbehandlung etc.). – Weitere Festlegung der Rahmenbedingungen bzgl. Infrastruktur.
Ausgang	– Fahrzeugproduktlinienmodell (Ausstattungsmerkmale für die Produktlinie und ihre Produktvarianten) – vollständige Dienstlandkarte mit Dienstbeschreibungen inkl. Schnittstellen und Black-Box-Verhalten – verfeinerte, konsolidierte technische Architektur (in Bezug auf Dienstlandkarte) inkl. erster Varianz (Steuergeräteproduktlinienmodell) – verfeinerte und konsolidierte Rahmenbedingungen
Durchführende	»Delegierte« der Fachstellen
Wiederverwendung	Merkmalsmodelle, Dienstlandkarte, Dienstbeschreibungen, technische Architekturen aus Vorgängerprojekten bzw. aus Unternehmensvorgaben
Varianz	Zu beachtende und zu unterstützende Varianz im Projekt (»äußere Varianz«) wird detailliert in Bezug auf Merkmalsmodell, Dienste und technische Architektur analysiert.

Beispiele

Der Dienst CBS dient der Beobachtung des realen Verschleißes von Verschleißteilen. Folgender Verschleiß/Verbrauch ist zu beobachten: Öl, Bremsbeläge, Bremsflüssigkeit, Zündkerzen (bei Ottomotoren), Partikelfilter (bei Dieselmotoren), ... Soweit Sensorik vorhanden ist, ist diese zu nutzen. Ansonsten ist auf Grundlage des Fahrprofils der Verschleiß abzuschätzen. Auf Grundlage der ermittelten Verschleißdaten ist ein optimierter Servicetermin im Kombiinstrument anzuzeigen. Bei »Luxusausstattungen« ist außerdem das Head-Up-Display zu verwenden.

```
Zur Skalierung der Infrastruktur: Die Infrastruktur für die
Standard-Variante ist ohne MOST zu konzipieren. Mit der ersten
Audio-Sonderausstattung wird ein MOST-Bus verbaut.
```

3.3 Phase 3: Systemarchitekturen entwickeln

In der Phase werden die Produkte / die Produktlinie auf Gesamtsystemebene betrachtet und in geeignete Teilsysteme bzgl. Funktionen, Infrastruktur und Software aufgeteilt – mit dem Ziel, den Rahmen für die Fachstellen anzugeben, auf dessen Grundlage die »virtuelle« Integration der Funktionen, Steuergeräte, Softwarekomponenten möglich ist.

Ziel

Das Ziel dieser Phase ist es, die Funktionen der Produkte / der Produktlinie so zu beschreiben, dass deren Verhalten und Schnittstellen so festgelegt sind, dass eine implementierte Funktion lauffähig wäre und mit den anderen Funktionen zusammenspielen würde.

Die Infrastrukturbetrachtungen zielen darauf ab, eine Steuergerätetopologie als Teil der technischen Architektur vorzugeben, auf der die Funktionen ablaufen müssen bzw. können. Eine entsprechende Verteilung der Funktionen auf die Steuergeräte wird vorgenommen.

Die zu implementierenden Einheiten (Subsysteme) werden festgelegt und auf die Fachstellen aufgeteilt, die in Phase 4 die weitere Entwicklung verteilt durchführen.

Nicht-Ziel

Es ist nicht Ziel, die Funktionen detailliert intern zu modellieren und als White-Box zur Verfügung zu stellen. Dies wird in Phase 4 unter Verantwortung der Fachstellen durchgeführt (vgl. Diskussion System- vs. Komponentenebenen, Kapitel 2).

Ende

Die Phase ist beendet, wenn

- alle Funktionen soweit modelliert sind, dass ihr Zusammenspiel gesichert ist, der Datenaustausch beschrieben ist und das Verhalten beschrieben ist, d. h. ihre

Schnittstelle festgelegt ist; Anforderungen an das Verhalten aus Systemsicht festgelegt sind und der Informationsfluss auf Funktionsnetzebene definiert ist (Werden alle benötigten Signale bereitgestellt? Gibt es bereitgestellte Signale, die nicht verwendet werden?),
- die technische Architektur mit den Schnittstellen und Bussen der Steuergeräte festgelegt ist,
- die technischen Signale (»Nachrichtenkataloge«) festgelegt wurden,
- die Rahmenbedingungen für die einzelnen Funktionen festgelegt wurden (z. B. Verteilung, Konfiguration).

Artefakte & Aktivitäten

Artefakte & Aktivitäten für die Phase sind in Tabelle 3 aufgelistet.

Tabelle 3 Übersicht von Artefakten und Aktivitäten in Phase 3 »Systemarchitekturen entwickeln«

Merkmal	**Inhalte**
Eingang	Sämtliche Ergebnisse der Phase 2.
Verarbeitung	– Die Phase dient zum Architekturentwurf der Steuergerätetopologie und des Funktionsnetzes. – Dazu werden die Dienste der Dienstlandkarte durch Funktionen »realisiert«, indem Dienste in Funktionen aufgeteilt werden, abhängige Dienste zu Funktionen zusammengefasst werden und in ein Funktionsnetz integriert werden. – Gleichzeitig wird die Hardwaretopologie verfeinert und das entstehende Funktionsnetz auf die technische Architektur verteilt. – Der Schritt ist beendet, wenn die Funktionen fein genug beschrieben sind, um auf die Topologie verteilt werden zu können und für die detaillierte, interne Modellierung an die verantwortlichen Fachstellen übergeben werden kann. – Es werden Anforderungen an die technische Architektur in Bezug auf eventuelle Auswirkungen auf die Softwarearchitektur gestellt, z. B. ob AUTOSAR verwendet wird, welche Basis-SW eingesetzt wird etc.
Ausgang	– Funktionsnetz mit Funktionsmodellen (Schnittstellenfestlegung) – Technische Architektur: Hardwaretopologie mit Steuergerätevorgaben – Verteilung des Funktionsnetzes auf die technische Architektur – Nachrichtenkataloge
Durchführende	»Delegierte« der Fachstellen, Systemstelle
Wiederverwendung	– Spezifikationen von funktionalen Rahmenarchitekturen und Funktionsbausteinen aus Vorgängerprojekten oder aus projektübergreifenden Domänenmodellen – Rahmenarchitekturen für die Steuergerätetopologie und Hardwarebausteinen aus Vorgängerprojekten oder aus projektübergreifenden Domänenmodellen

↪ ...

Merkmal	Inhalte
Varianz	Klärung auf Systemarchitekturebene, wie mit der identifizierten Varianz (aus der Phase 2) umgegangen werden soll: »Implementierung von Varianz«, Auflösungszeitpunkte (»binding time«), Produktlinienansätze für Subsysteme (z. B. Softwareproduktlinien für einzelne Systemfunktionen, die projektübergreifend Anwendung finden sollen) etc.

Beispiele

Die Konzeption von CBS wird von der Fachstelle *X* verantwortet. Software zum CBS, die zum Beispiel auf dem Kombiinstrument untergebracht werden soll, ist mit der verantwortlichen Fachstelle für das Kombiinstrument abzustimmen.

Das Türsteuergerät soll Zwei- als auch Vier-Türer ansteuern können. Zu den Umsetzungsmöglichkeiten, die in Frage kommen und geklärt werden müssen, gehören folgende Varianten: zwei alternative Funktionen oder eine für alle Fälle passende Funktion vorsehen, zwei alternative Steuergeräte oder ein für alle Fälle passendes Steuergerät bauen lassen. Varianten, die für alle Fälle passen, führen zu konfigurierbaren Funktionen bzw. Steuergeräten, alternative Varianten zu austauschbaren Funktionen bzw. Steuergeräten.

3.4 Phase 4: Subsysteme entwickeln

In der vierten Phase wird jedes in Phase 3 identifizierte Subsystem bzgl. Funktionsmodellierung und technische Architektur weiterentwickelt, soweit bis die Softwareebene erreicht wird und eine entsprechende Softwarearchitektur entworfen wird. Zum Entwurf der Softwarearchitektur gehört u. a. die Entscheidung über Varianz in der Software, es entstehen die Softwareproduktlinien.

Das zu bearbeitende Subsystem kann nur den Funktionsbereich (Funktionsentwicklung, Softwareentwurf), nur den Infrastrukturbereich (Entwicklung der technischen Architektur auf Subsystemebene, Steuergeräteentwicklung), aber auch sämtliche Bereiche umfassen. Weitere Aufteilungen sind möglich. Je nachdem ist mehr oder weniger Absprache / Koordination nötig. Eine verantwortliche Stelle für die Systemsicht ist parallel einzurichten, dies kann eine eigene Fachstelle sein (»Systemstelle«) oder aber durch dezentrale Koordinierung aller Fachstellen erfolgen.

Stellt sich in dieser Phase (oder in weiteren Phasen) heraus, dass die Gesamtsystembeschreibung (aus Phase 3) zu ändern ist, geschieht das in Koordination mit

der »Systemstelle« und / oder anderen Fachstellen (gemäß des Änderungsmanagements).

Ziel

Das Ziel dieser Phase ist es, die Funktionen und Steuergeräte des Produkts so zu beschreiben und zu modellieren, dass sie von einem Zulieferer implementiert werden können oder dass die vorhandenen Funktionen / Softwarekomponenten bzw. Steuergeräte integriert werden können. Dazu ist es u. a. notwendig, die vom Zulieferer zu verwendenden Dokumente vorzubereiten und zu übermitteln. Außerdem sind alle Entscheidungen über Varianz in der Softwarearchitektur, also über die Softwareproduktlinie zu treffen.

Ende

Die Phase ist beendet, wenn

- die Funktionen implementierbar beschrieben sind, d. h. die Softwarearchitektur und Softwarekomponenten spezifiziert worden sind, evtl. Referenzimplementierungen hierzu erstellt wurden,
- die Steuergeräte implementierbar beschrieben sind,
- die Dokumente für den Zulieferer erstellt wurden,
- die Implementierung mit dem Zulieferer abgestimmt ist.

Artefakte & Aktivitäten

Artefakte & Aktivitäten für die Phase sind in Tabelle 4 aufgelistet.

Tabelle 4 Übersicht von Artefakten und Aktivitäten in Phase 4 »Subsysteme entwickeln«

Merkmal	Inhalte
Eingang	Sämtliche Ergebnisse der vorherigen Phase.
	↪ ...

Merkmal	Inhalte
Verarbeitung	– Das zu implementierende Subsystem wird so genau spezifiziert, dass es am Ende einem Zulieferer zur Implementierung übergeben werden kann. Dabei kann ein vollständiger Modellierungsprozess angestoßen werden. Es ist auch möglich, das Subsystem wieder zu unterteilen und diese Teile einzeln zu implementieren. – Detaillierte Funktionsmodellierung (ggf. hardwarespezifische Verfeinerungen der Funktionsmodelle), Spezifikation zugehöriger Hardware, Funktionsverteilung, darauf aufbauend: Softwarearchitekturentwicklung und Spezifikation von Softwarekomponenten. – Berücksichtigung bestehender Funktionen, Softwarekomponenten – Entscheidung über Varianz in der Softwarearchitektur (Softwareproduktlinien) – In dieser Phase ist u. U. der Zulieferer mit seiner Kompentenz einzubinden.
Ausgang	– Hard- und Softwarebeschreibung (Architektur bzw. Komponenten) als Lastenheft (Modelle für Struktur und Verhalten) – Software als Referenzimplementierung
Durchführende	Fachstellen, Systemstelle, unter Umständen in Abstimmung mit den Zulieferern
Wiederverwendung	alle möglichen Artefakte
Varianz	wird weiter verfeinert, bewertet und Umsetzungskonzepte werden erarbeitet (z. B. Auflösungszeitpunkte werden festgelegt, Entscheidung bzgl. 150 %-Ansatz oder Softwarekomponentenproduktlinienansatz werden getroffen etc.)

Beispiele

Detaillierte Spezifikation von CBS bzgl. Produkt-, Funktions- und Infrastrukturbereich.
Funktionale Aspekte:

- Festlegung der Algorithmen zur Ermittlung des Verschleißes der jeweiligen Verschleißteile.
- Konzeption der redundanten Datenhaltung

Softwareaspkete:

- Konzeption von »Master-Slave-Ansatz« vs. »Client-Server-Ansatz«.
- Festlegung, welcher funktionale Umfang auf welchen Steuergeräten laufen soll und wie die Softwarekomponenten dazu aussehen.
- Festlegung der Kommunikationsmechanismen.
- Aufbau einer Softwareproduktlinie für CBS.

Hardwareaspekte:

- Varianz in der technischen Architektur wirkt sich auf die CBS-Umfänge aus ...
- ...

3.5 Phase 5: Implementierungen

Die Phase repräsentiert die Aktivitäten der Zulieferer bei der Umsetzung der bis zu dieser Phase erstellten Spezifikationen, die üblicherweise in Form von Lastenheften übergeben werden. Ergebnisse sind, je nachdem, installierbare Softwarekomponenten (in Form von Binärcode und/oder Software-Quellcode), verbaubare Steuergeräte oder ganze Subsysteme mit Hardware- und Softwareanteilen.

Im Rahmen der Pflichtenhefterstellung und Implementierung können die Zulieferer auch Rückwirkungen auf die modellierten Systeme haben. Diese werden dann entsprechend einvernehmlich geändert.

Diese Phase wird im Rahmen von VEIA nicht weiter betrachtet, daher nachfolgend eine verkürzte Tabelle der Artefakte und Aktivitäten.

Tabelle 5 Übersicht von Artefakten und Aktivitäten in Phase 5 »Implementierungen«

Merkmal	**Inhalte**
Eingang	Sämtliche Ergebnisse der vorherigen Phase.
Verarbeitung	Implementierung der vorgegebenen Modelle
Ausgang	implementierte Steuergeräte mit Softwarekomponenten
Durchführende	hauptsächlich Zulieferer

3.6 Phase 6: Integration und Test

Die sechste Phase repräsentiert den rechten Ast im V-Modell. Der modellbasierte und systemorientierte Ansatz im VEIA-Referenzprozess dient dazu, diese Phase dahingehend zu unterstützen, dass bereits in der Vorgängerphasen jeweilige (virtuelle) Absicherungen durchgeführt und Tests entsprechend vorbereitet werden (Testfälle etc.). Dies führt im Sinne des Produktentstehungsprozess (PEP) zu einer frühen, kostengünstigeren Absicherung, indem insgesamt weniger Hardwareeinsatz (weniger A-Muster) notwendig werden. Der gezielte Einsatz von Produktlinien verspricht außerdem eine bessere Testabdeckung.

Vollständig werden bisherige Integrationsaufgaben und Tests nicht abzulösen sein. Jedoch sollen Fehler und Fehlkonzeptionen so früh wie möglich entdeckt und ausgemerzt werden können.

Tabelle 6 Übersicht von Artefakten und Aktivitäten in Phase 6 »Integration und Test«

Merkmal	**Inhalte**
Eingang	Sämtliche Artefakte aus den Vorgängerphasen, insbesondere Testfälle, sowie die Teilprodukte (Steuergeräte, Softwarekomponenten) der Zulieferer, die zusammengebaut werden.
Verarbeitung	Testen und integrieren der Bauteile
Ausgang	Testergebnisse, Änderungsvorgaben auf Entwicklungsartefakte
Durchführende	Integratoren, Fachstellen
Wiederverwendung	Erfahrungswerte, Testfälle aus Vorgängerprojekten etc., Modelltests
Varianz	muss bei der Integration und bei Tests adäquat berücksichtigt werden

In dieser Phase zeigt sich der Nutzen der Modellierung, der Produktlinienmodelle, des gesamten Referenzprozesses sehr deutlich. Zum einen wird durch die Artefakte und Phasen sichergestellt, dass die entwickelten Modelle die jeweils vorhergehende Spezifikation erfüllen. Damit braucht nach der Implementierung die Algorithmik nicht mehr funktional getestet werden, sondern das Verhalten wurde bereits auf Modellebene abgeprüft.

Außerdem können Test- und Absicherungsstrukturen der Modellierung wiederverwendet werden, wenn die implementierten Systeme getestet werden. Das heißt, den fertigen Systemen können die gleichen Stimulationen wie den Modellen gegeben werden, um zu testen. Damit werden zum einen weniger Fehler in die fertigen Systeme eingebaut, zum anderen braucht man weniger Aufwand, um Fehler zu finden.

4 Fallbeispiel »Condition Based Service« (CBS)

Im Folgenden werden anhand des im Forschungsprojekt VEIA vom Projektpartner BMW zur Verfügung gestellten Fallbeispiels *CBS* die zuvor im Dokument diskutierten Artefakte im VEIA-Referenzprozess illustriert. Dies dient der Veranschaulichung der Konzepte. Es spiegelt nicht notwendigerweise die tatsächliche Entwicklung von CBS bei BMW wider, diesbezüglich werden im Projekt VEIA keine Entscheidungen getroffen. Eine ausführlichere Modellierung von CBS in Bezug auf die Projektziele wird in späteren Phasen im Forschungsprojekt VEIA durch die jeweiligen Projektpartner erfolgen, vgl. hierzu [GR06].

Die nachfolgende Aufarbeitung des Fallbeispiels zur Illustration der entsprechenden Entwicklungsartefakte gemäß Kapitel 2 bzw. Kapitel 3 bezieht sich auf Informationen aus folgenden, von BMW zur Verfügung gestellten Dokumenten:

- BMW-Lastenheft zu CBS 5 [BMW05a],
- Servicehandbuch CBS für E87, E90 und E91 [BMW05b],
- Rhapsody-Modell zum CBS-Client.

Es ist zu beachten, dass die vorgestellten Artefakte hauptsächlich durch Reengineering gewonnen wurden. Bei einer Vorwärtsmodellierung würden bestimmte Artefakte anders aussehen oder anders motiviert werden. Während des Forschungsprojekts wird das Reengineering durch Engineering ersetzt werden.

Aufgabe und Verwendungszweck des Lastenhefts

Das Lastenheft enthält Informationen, die im Verlauf während der Prozessphasen 1 bis 4 (vgl. Abschnitte 3.1 bis 3.4) festgelegt werden müssen. Im Lastenheft selbst wird dies wie folgt beschrieben [BMW05a, S. 15]:

> Das vorliegende Systemlastenheft dient zur Darlegung der technischen Anforderungen an eine zu entwickelnde Systemfunktionalität.
>
> Je nach Vermerk in der Änderungsdokumentation dieses Dokumentes dient es als technische Unterlage zur Anfrage (LH-Anfrage) eines Angebots für eine Entwicklungsleistung oder als technische Definition des Entwicklungsziels (LH-Entwicklungsziel) in Ergänzung zum Entwicklungsvertrag / Entwicklungsauftrag. Das LH dokumentiert den jeweils aktuellen Entwicklungsstand des Projektes und bildet die Grundlage für Anfragen und Verhandlungen.

Die im Lastenheft gestellten Forderungen betreffen grundsätzlich die funktionellen Eigenschaften. Die Festlegung der Systemfunktion und Steuerung der Umsetzung liegt bei der CBS-Fachabteilung, die Verantwortung für die Umsetzung der Komponentenfunktionalitäten bei den jeweiligen Entwicklungsfachstellen.

Die Verantwortung für das CBS-Produktkonzept, d. h., welche Funktionen in welcher Ausprägung im Fahrzeug integriert werden, liegt bei der Fachstelle »Servicetechnik, Marktentwicklung, Wartung und Instandsetzung«.

Aufgabe und Verwendungszweck des Servicehandbuchs

Das Servicehandbuch ist eine Dokumentation für die BMW-Werkstätten und den BMW-Service und enthält Informationen, die nach einer Produktentwicklung im Sinne der Dokumentation bzw. eines Nutzerhandbuchs erstellt werden. Rückwirkend lassen sich jedoch daraus Informationen herausziehen, die bei der Festlegung von strategischen Zielen, Rahmenbedingungen, Produktlinienbeschreibungen, Diensten und einer initialen technischen Architektur gemäß Phase 1 (Abschnitt 3.1) und Phase 2 (Abschnitt 3.2) relevant sind.

Im Unterschied zum Lastenheft beschreibt das vorliegende Dokument die CBS-Vorgängerversion 4.

Aufgabe und Verwendungszweck des Rhapsody-Modells

Das Rhapsody-Modell zum CBS-Client repräsentiert Entwicklungsstände aus Phase 4 (Abschnitt 3.4). Es spezifiziert die Softwarearchitektur für CBS-Clients: die Schnittstellenfestlegungen und Verhaltensbeschreibungen. Ziel ist, ausführbare, simulierbare und damit leichter nachvollziehbare Spezifikationen, die Teil des Lastenhefts werden, zu erhalten. Außerdem wird »serienreifer« Referenzcode aus dem Rhapsody-Modell generiert, der dem entsprechenden Zulieferer als Referenz zur Verfügung gestellt wird.

In den folgenden Abschnitten werden nun einzelne Entwicklungssichten und Abstraktionsebenen auf die »Funktionalität« CBS (Kontext, Anforderungen und Lösungsentwürfe) gemäß den in den Kapiteln zuvor skizzierten Artefakten aufgezeigt und zur Diskussion gestellt. Sie stellen Problembeschreibungen und Anforderungen für die im Forschungsprojekt zu entwickelnde Methodik aus Sicht der Fallstudie dar.

4.1 Strategische Ziele / Rahmenbedingungen

Bei BMW wurde vor einiger Zeit ein neues Wartungskonzept namens `Condition Based Service (CBS)` eingeführt, das in der Anfangszeit `Bedarfsorientierter Service (BOS)` hieß. Ziel war die Abkehr vom bisherigen Regelservice zu einem Service, der sich am Verschleiß orientiert [BMW05a, S. 17].

Der Regelservice sieht feste Wartungs- und Serviceintervalle vor, dagegen wird mit CBS ein verbrauchsorientierter Service eingeführt, der an dem tatsächlichen Verschleiß angepasste Wartungs- und Serviceintervalle ermöglicht. Sowohl bei Wartung als auch bei Service sind – direkt oder indirekt – Fahrer und Vertrieb bzw. Service betroffen bzw. beteiligt.

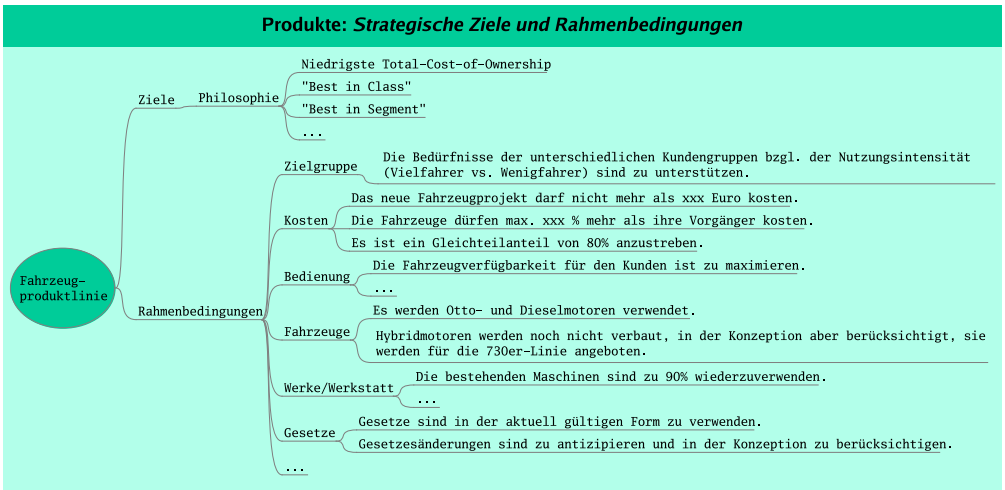
Der tatsächliche Verschleiß soll

1 auf Basis bereits vorhandener Sensorik und / oder
2 auf Basis von Verschleißmodellen,

ggf. unter Zuhilfenahme von Fahrprofilen, ermittelt bzw. berechnet werden.

Strategische Ziele und Rahmenbedingungen für ein Fahrzeugprojekt sind in Abbildung 11 aufgelistet.

Abbildung 11 Produkte: Strategische Ziele und Rahmenbedingungen (Mindmap, beispielhafter Aussschnitt).



Für CBS sind solche Ziele und Rahmenbedingungen in [BMW05a, S. 17f.] (»CBS-Philosophie«) angegeben. Sie dienen dazu, die Grundidee sowie Vorgaben für die

Umsetzung dieser Funktionalität darzulegen:

Die Strategie zur Umsetzung lautet: »es sollten möglichst viele bereits on-board verfügbare Informationen für die Ermittlung des besten Servicezeitpunktes herangezogen werden, d. h., dass virtuelle Sensorik möglichst gegenüber der Auswahl neuer, hardwaremäßiger Sensoren vorgezogen wird«.

Das CBS-Konzept bringt dem Kunden spürbare Verbesserungen:

- Durch die Messung des Verschleißes können bisher notwendige Sicherheitsreserven verringert werden. Dadurch verlängert sich die Nutzungsdauer eines Bauteils.
- Die durch den Kunden bestimmbare Zusammenlegung von verschiedenen Serviceterminen verbessert die Mobilität und erhöht die Verfügbarkeit des Fahrzeuges.
- Durch die Messung und Auswertung des tatsächlichen Verschleißes wird die schonende Fahrweise des Kunden in Form von niedriger Cost-Of-Ownership belohnt.

CBS verfolgt folgende Ziele:

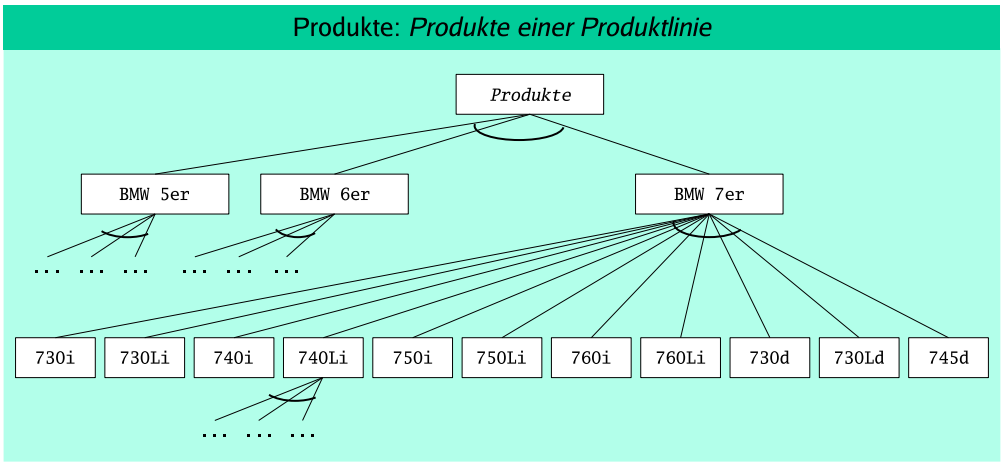
- Individuelles Wartungskonzept abhängig vom Verschleiß.
- Planbarkeit, Optimierung und Entscheidung von Wartungsterminen durch intelligente Intervall-Prognose der verschleißerfassenden Steuergeräte.
- Anzeige von Wartungsumfängen im Fahrzeug, sortiert nach Fälligkeit.
- Qualifizierte Beratung des Kunden durch Visualisierung der Fahrzeugdaten im Service-Annahme-Modul.

4.2 Produktlinienbeschreibung

Im Zuge der Definition der zu entwickelnden Fahrzeugproduktlinie (Abschnitt 3.2) müssen einerseits die Produkte benannt werden, die in Produktlinie gebaut werden sollen, andererseits die Merkmale aller Produkte dahingehend analysiert werden, welche Gemeinsamkeiten und welche Unterschieden zwischen den Produkten bestehen (sollen). Die Produktlinienbeschreibung ist Ausgangspunkt der Konfiguration der anderen Entwicklungsartefakte.

In Abbildung 12 werden die Produkte einer Produktlinie (hier für Fahrzeugtypen gemäß [BMW06]) als Merkmalsbaum (vgl. z. B. [KCH+90, CE00]) dargestellt. Dies

Abbildung 12 Produkte: Fahrzeuge (Ausschnitt).



kann, wie abgebildet, über mehrere Hierarchieebenen erfolgen. Zum Beispiel: `BMW 7er` sind entweder `730i`, `730Li`, ... oder `745d`. Neben `BMW 7er` könnten auch der `BMW 5er` und `BMW 6er` zur Produktlinie gehören.

In Abbildung 13 wird ein Ausschnitt der Ausstattungsmerkmale für die 7er-Produktlinie, vgl. [BMW06], als Merkmalsbaum dargestellt: Jedes Merkmal wird dahingehend charakterisiert, ob jedes Produkt der Produktlinie dieses Merkmal besitzt (obligatorisches Merkmal; *mandatory feature*), ob nur manche Produkte das Merkmal besitzen (optionales Merkmal), oder ob es alternative Ausprägungen eines Merkmals in den verschiedenen Produkten gibt (XOR-Merkmale).

In Abbildung 14 wird ein Ausschnitt der Ausstattungsmerkmale bzgl. der `CBS`-Funktionalität dargestellt: welche Verschleißteile beobachtet werden sollen und welcher Zusammenhang zu anderen Produktmerkmalen besteht; zum Beispiel kann das Verschleißteil `Zündkerzen` nur beobachtet werden, wenn das Fahrzeug einen `Ottomotor` besitzt.

4.3 Dienstlandkarte und Beschreibung der Dienste

Die Dienstlandkarte dient der Strukturierung der Anforderungen bzgl. der zu realisierenden Funktionalität, die in Form von *Diensten* erfasst werden. In der Dienstlandkarte werden strategische Ziele und Rahmenbedingungen aufgegriffen, konkretisiert und konsolidiert. Insbesondere die Dienstlandkarte ist Teil der

Abbildung 13 Produktmerkmale (Ausschnitt).

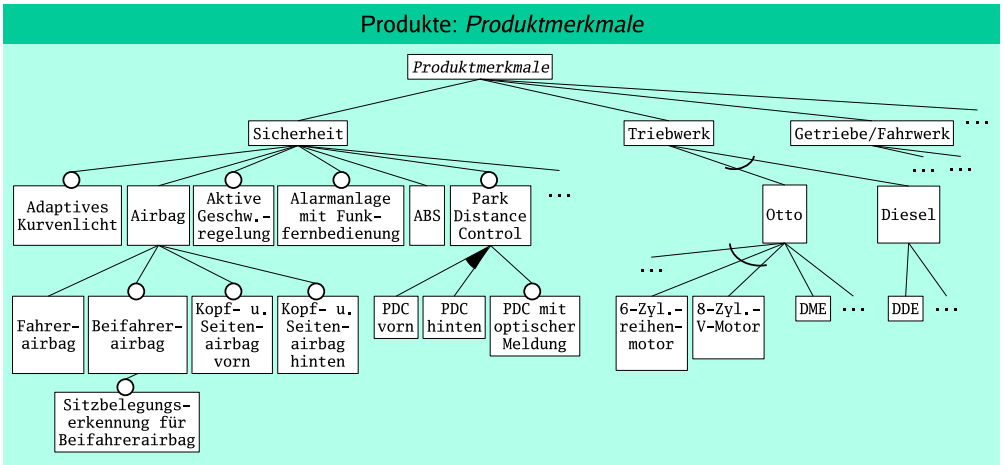
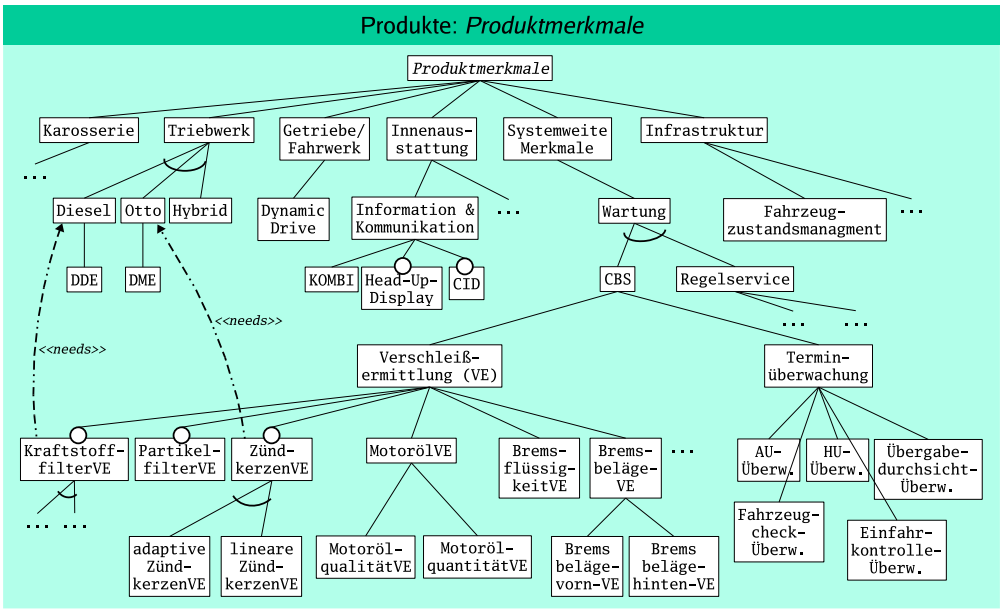


Abbildung 14 Produktmerkmale, für CBS detailliert (Ausschnitt).

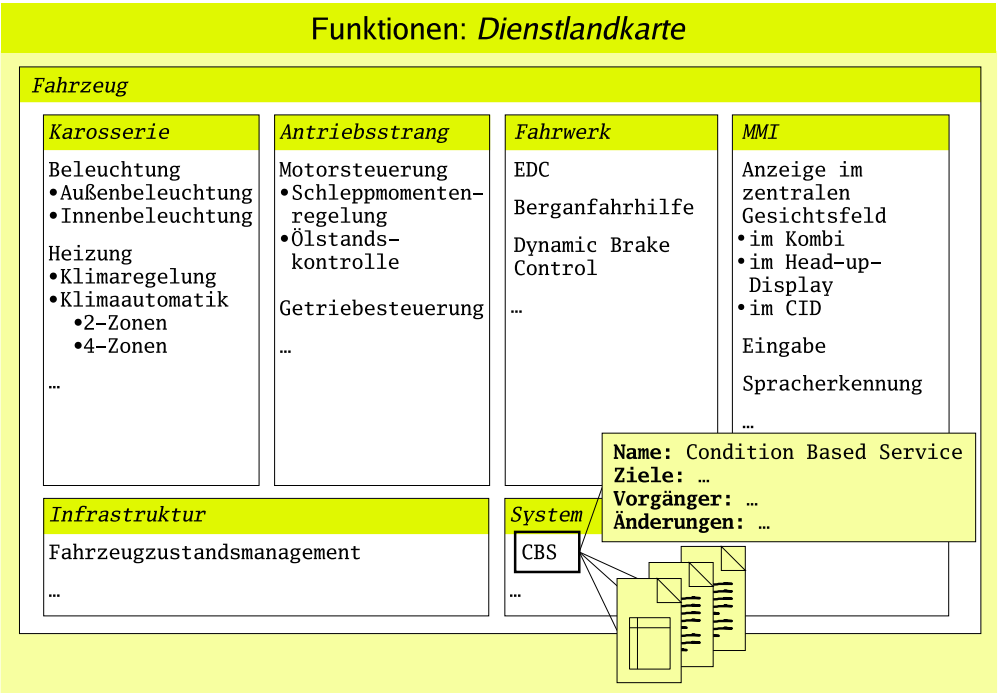


Forschung, die hier dargestellten Abbildungen sind daher als Forschungszwischenresultate, nicht als fertige Ergebnisse zu betrachten.

Ziel ist, die zu realisierende Produktlinie und ihre zugehörigen Produkten bzgl. der Funktionalität und den Randbedingungen zu *definieren*. Der Dienst `Condition Based Service (CBS)` ist dabei ein Dienst von vielen Diensten in dieser Systembetrachtung.

Die Abbildungen 15 und 16 zeigen zwei Beispiele für eine Dienstlandkarte: Abbildung 15 eine schematische Darstellung und Abbildung 16 eine Darstellung als Mindmap.

Abbildung 15 Dienstlandkarte (Schema).



Die detaillierte Betrachtung jedes einzelnen Dienstes in der Dienstlandkarte (d. h. der nach »außen« sichtbaren Funktionalität) dient zur Erfassung der für den Dienst benötigten Informationen, wie sie verarbeitet werden und wer daran beteiligt ist.

In Abbildung 17 ist für den Dienst `CBS` dargestellt, welche Teilfunktionalität dazu gehört und welche Systemschnittstelle für `CBS` notwendig ist. Die in Abbildung 17 dargestellten Teildienste werden nachfolgend näher spezifiziert.

Abbildung 16 Dienstlandkarte (Illustration als Mindmap).

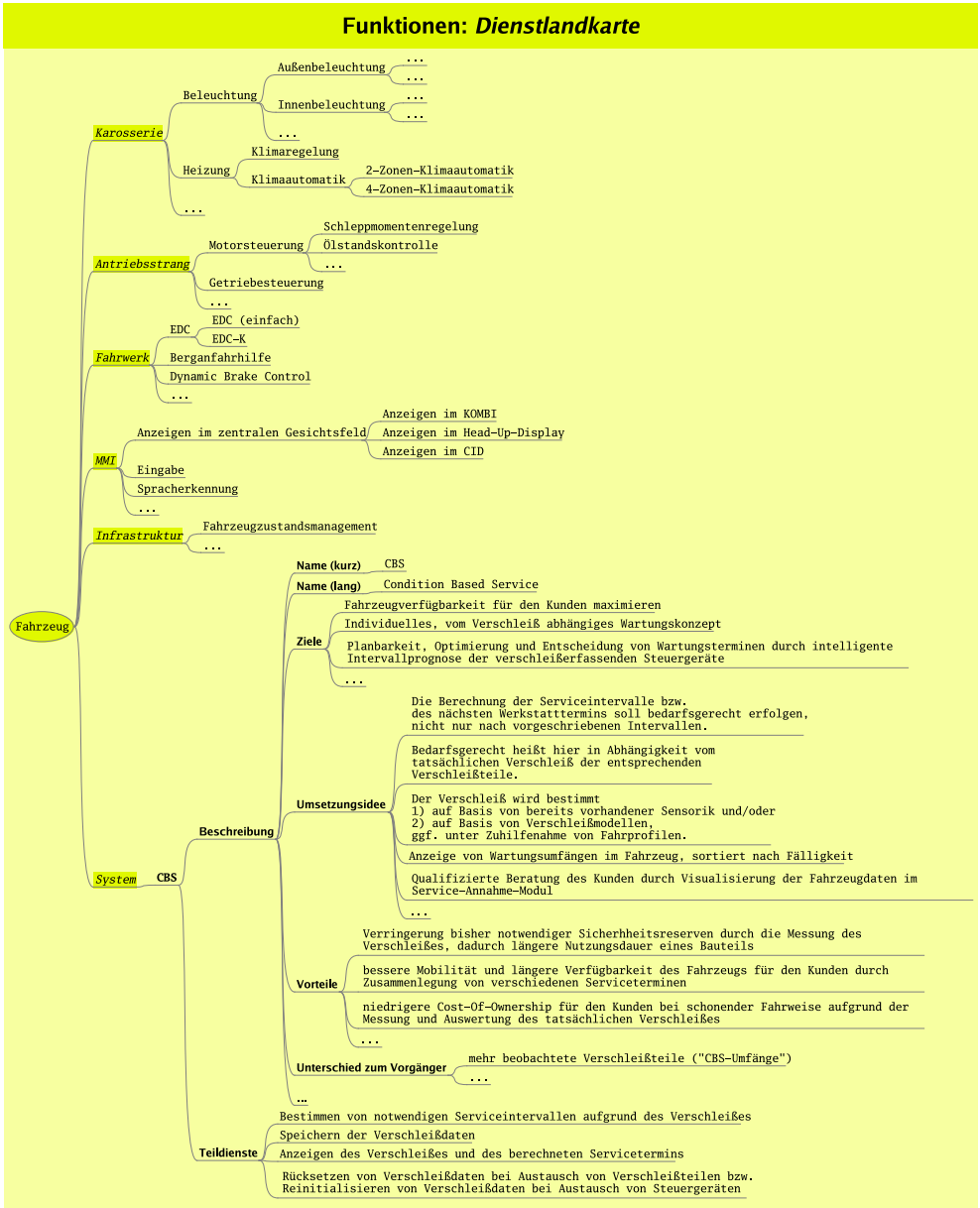
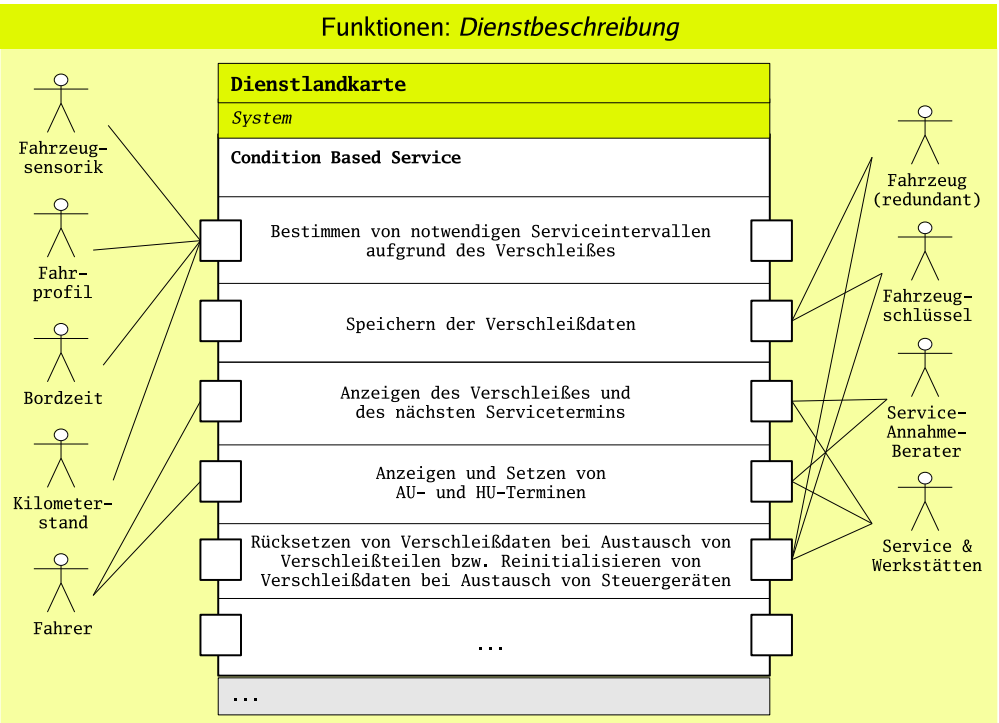


Abbildung 17 Dienstbeschreibung CBS.



Teildienst »Bestimmen von notwendigen Serviceintervallen aufgrund des Verschleißes«

Aufgabe:

- Erfassen des Verschleißes: Es gibt verschiedene Methoden der Verschleißerfassung: verbrauchsorientiert (auf Basis von Sensordaten), zeitabhängig und/oder wegstreckenabhängig.
- Bewerten des Verschleißes
- Bestimmen von Serviceintervallen: Sortieren, Zusammenfassen, Optimieren der Termine[3]

[3] In der bisherigen Realisierung von CBS wird dieser Umfang vom CBS-Master umgesetzt. Ein CBS-Client schlägt für das jeweilige Verschleißteil einen Servicetermin vor und übermittelt diesen an den Master.

Schnittstellen:

- `Fahrzeugsensorik:` Der Teildienst benötigt Informationen über den aktuellen Verschleiß von Verschleißteilen. Dafür werden vorhandene (reale oder virtuelle) Sensoren des Fahrzeugs verwendet. Wenn möglich, sollen keine »neuen«, extra für CBS einzubauende Sensoren verwendet werden.
- `Fahrprofil:` ggf. ist das Fahrprofil für die Hochrechnung des Verschleißes zu berücksichtigen.
- `Bordzeit:` Für einen zeitabhängigen Verschleiß wird die Bordzeit benötigt.
- `Kilometerstand:` Für einen Verschleiß, der von der bisher zurückgelegten Wegstrecke abhängt, wird der Kilometerstand benötigt.

Teildienst »`Speichern der Verschleißdaten`«

Aufgabe:

- »kontinuierlich« die erhobenen Daten persistent machen

Schnittstellen:

- `Fahrzeug (redundant):` Die erfassten Verschleißdaten und berechneten Serviceintervalle sind persistent abzuspeichern, um später ausgelesen sowie bei/nach Reparaturen wiederhergestellt werden zu können.
- `Fahrzeugschlüssel:` Für das persistente Speichern von erfassten Verschleißdaten und berechneten Serviceintervallen wird der Fahrzeugschlüssel genutzt. Der Fahrzeugschlüssel wird in der Service-Annahme ausgelesen.

Teildienst »`Anzeigen des Verschleißes und des nächsten Servicetermins`«

Aufgabe:

- (einfache oder höherwertige) Anzeige der Verschleißdaten und des nächsten Servicetermins

Schnittstellen:

- `Fahrer:` Dem Fahrer sind relevante Verschleißdaten und Servicetermine adäquat anzuzeigen. Unter Umständen sind verschiedene Varianten der Anzeige vorzusehen (Standardanzeige vs. Anzeige bei Komfort-Ausstattung).

– `Service-Annahme-Berater / Service und Werkstätten:` Der Service-Annahme-Berater sowie Service und Werkstätten müssen die Verschleißdaten und die berechneten Servicetermine geeignet auslesen können (via Fahrzeugschlüssel und `On-Board-Diagnose (OBD)`).

Teildienst »Anzeigen und Setzen von AU- und HU-Terminen«

Aufgabe:

– Abgas- und Hauptuntersuchungen sind gesetzlich vorgeschrieben. Die Termine werden von `CBS` verwaltet und müssen entsprechend eingegeben und angezeigt werden können.

Schnittstellen:

– `Fahrer:` Der Fahrer kann sich die AU- und HU-Termine anzeigen lassen, ggf. darf er sie auch eingeben.

– `Service-Annahme-Berater:` Der Service-Annahme-Berater kann sich die AU- und HU-Termine anzeigen lassen, er darf sie auch eingeben.

– `Service und Werkstätten:` Der Service und die Werkstätten können die AU- und HU-Termine eingeben und sich anzeigen lassen.

Teildienst »Rücksetzen von Verschleißdaten bei Austausch von Verschleißteilen bzw. Reinitialisieren von Verschleißdaten bei Austausch von Steuergeräten«

Aufgabe:

– Neusetzen der Verfügbarkeit der Verschleißteile (Initialisierung, Zurücksetzen): wird ein Verschleißteil ausgetaucht, so muss das entsprechende Steuergerät darüber informiert werden und neu initialisiert werden. Wird ein Steuergerät ausgetauscht, so muss das neue Steuergerät mit der bisher berechneten Verfügbarkeit initialisiert werden.

Schnittstellen:

– `Service und Werkstätten:` Beim Austausch eines Verschleißteils oder des zugehörigen Steuergeräts müssen die entsprechenden Verschleißdaten und Servicetermine durch den Service und die Werkstätten neu gesetzt oder wiederhergestellt werden können.

– `Fahrzeugschlüssel:` Ggf. sind Verschleißdaten, die auf dem Fahrzeugschlüssel gespeichert sind, wiederherzustellen.

– `Fahrzeug (redundant):` Ggf. sind Verschleißdaten, die anderweitig im Fahrzeug gesichert wurden, wiederherzustellen.

4.4 Funktionsnetz / Funktionsbeschreibung CBS

Die funktionale Dekomposition des Gesamtsystems in funktionale Subsysteme und Teilfunktionen wird mit Hilfe des Funktionsnetzes angegeben. Im Unterschied zur Dienstlandkarte werden auch interne Funktionen berücksichtigt. Funktionen im Funktionsnetz spezifizieren die Umsetzung der gewünschten, in der Dienstlandkarte definierten Funktionalität, indem festgelegt wird, wie mit Abhängigkeiten zwischen Diensten umgegangen wird, beispielsweise durch Controller-Funktionen und Client-Server-Beziehungen zwischen Funktionen.

Das Funktionsnetz dient der funktionalen Integration einzelner Funktionen und Subsysteme, die in den einzelnen Fachstellen entworfen werden. Mit Funktionsnetz und Funktionsmodellen wird eine zu gestaltende Softwarearchitektur vorbereitet.

In den Abbildungen 18 und 19 wird die Funktionalität, die bzgl. CBS in der Dienstebene skizziert wurde, durch Funktionen konkretisiert. In Anbetracht dessen, dass es verschleißteilabhängige Teilfunktionen gibt, wird bereits auf Funktionsebene eine Client-Server-Architektur vorbereitet, indem zwischen einem zentralen Master und verteilten Clients unterschieden wird. Diese funktionale Dekomposition (Client- vs. Masteranteile) ist eine *Entwurfsentscheidung* beim Übergang von der Dienstlandkarte zum Funktionsnetz und muss entsprechend dokumentiert und begründet werden.

Beide Abbildungen verdeutlichen auch, dass es unterschiedliche Perspektiven auf die Funktionalität von CBS gibt: Zum einen gibt es die Perspektive eines Fahrzeugprojekts, in dem spezifisch festgelegt wird, welche Verschleißteile wie beobachtet werden sollen, vgl. hierzu Abbildung 18 mit den CBS-Umfängen für die Produktlinie L6 lt. [BMW05a, S. 46f.]. Zum anderen gibt es die Perspektive einer Softwareproduktlinie für CBS mit dem Ziel, eine universelle Lösung zu entwickeln, die für alle bzw. mehrere Fahrzeugproduktlinien verwendbar ist. Auf Funktionsnetzebene zeigt die Skizze in Abbildung 19 dies in erster Annäherung, indem die prinzipielle funktionale Architektur für CBS musterhaft dargestellt wird – unabhängig von beispielsweise der jeweiligen Anzahl der CBS-Umfänge in einem Fahrzeug.

Abbildung 18 CBS-Funktion: Funktionsaufteilung in Master- und Clientanteile (L6-Umfang, Ausschnitt).

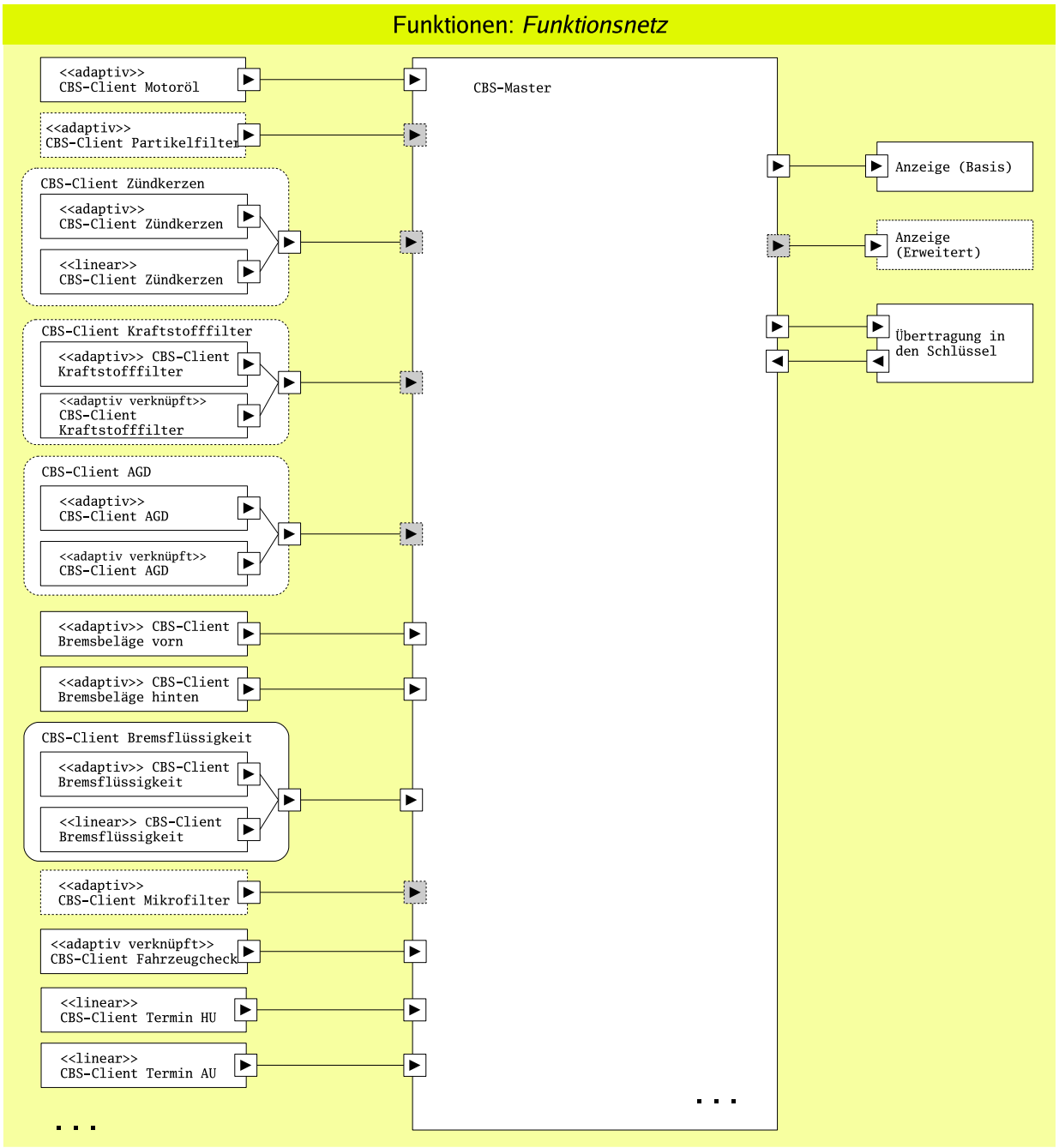
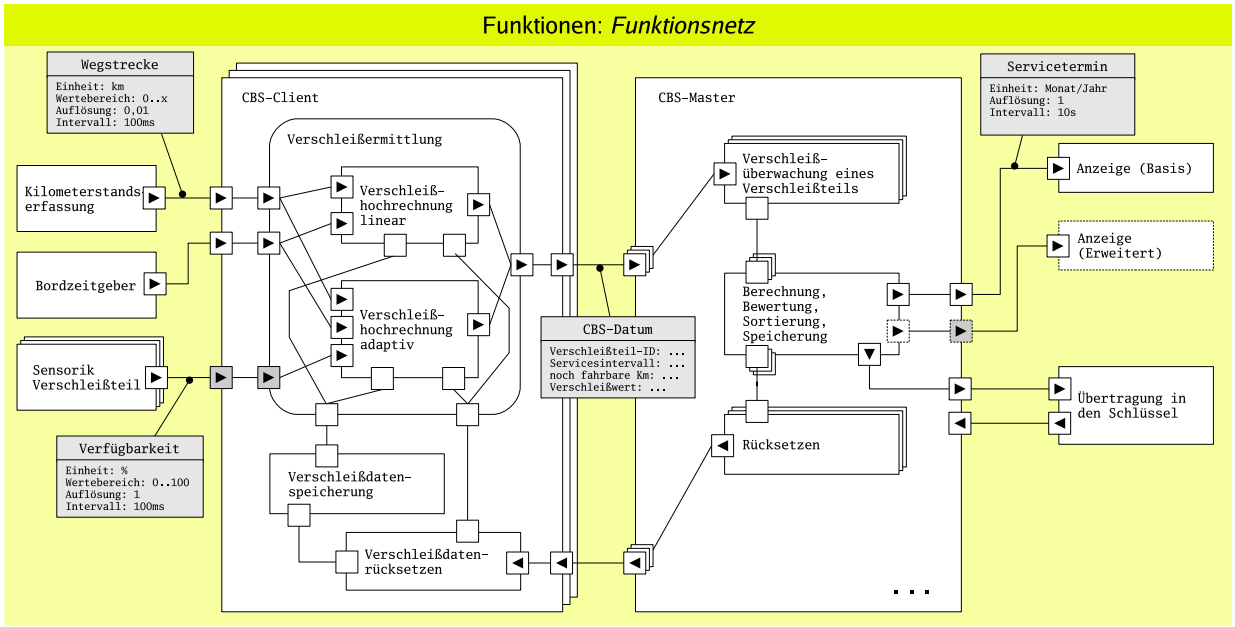


Abbildung 19 CBS-Funktion: prinzipielle Funktionsaufteilung in Master- und Clientanteile (Muster).



In den Abbildungen wird bereits funktionale Varianz dargestellt. Gemäß der im Produktmerkmalsmodell festgelegten Verschleißermittlungsmethoden für die jeweiligen Verschleißteile gibt es beispielweise zwei funktionale Alternativen für den CBS-Umfang `Zündkerzen`: linear und adaptiv.

Wie in Abbildung 19 dargestellt ist, verarbeiten die beiden Verschleißermittlungsmethoden unterschiedliche Informationsumfänge: während die lineare Variante sich auf die zurückgelegte Wegstrecke, die aktuelle Bordzeit und ggf. auf das (anderweitig) ermittelte Fahrprofil bezieht, wird in der adaptiven Variante zudem auf die Sensorik des Verschleißteils zurückgegriffen.

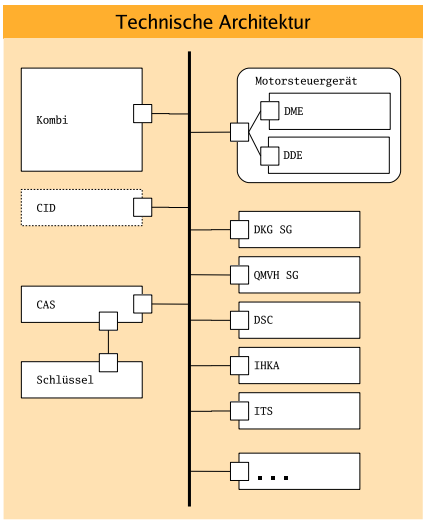
In Abbildung 19 wird auch beispielhaft gezeigt, wie Signale in Funktionsnetze einzutragen sind. Aus Gründen der Übersichtlichkeit und der noch unvollständigen Spezifikation sind nur vier Signale zu sehen. Sind alle Signale eingetragen, so kann z. B. für ein Ausgangssignal festgestellt werden, von welchen Eingangssignalen es beeinflusst wird.

4.5 Technische Architektur

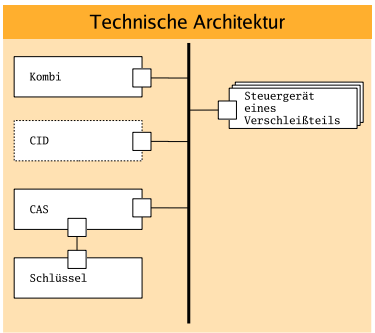
Die für CBS relevante technische Architektur, bestehend aus der Bustopologie und Steuergeräten, ist in den Abbildungen 20(a) und 20(b) skizziert. Auch hier lassen sich die Perspektiven »spezifische Fahrzeugproduktlinie« und »prinzipielle technische Architektur« (für eine Softwareproduktlinie) unterscheiden: Abbildung 20(a) zeigt einen Ausschnitt der technischen Architektur für die Produktlinie `L6`. Abbildung 20(b) verallgemeinert dies und stellt die prinzipielle technische Architektur dar, die für `CBS` relevant ist.

Abbildung 20 Technische Architektur, relevante Anteile für CBS.

(a) *L6-Umfang.*



(b) *Prinzipielle Architektur (Muster).*



Die technische Architektur berücksichtigt die in der Produktlinienbeschreibung vorgegebene Varianz bezüglich des optionalen Vorhandenseins von Steuergeräten (zum Beispiel `Central Information Display (CID)` für Komfortausführungen) oder bezüglich Alternativen bei Steuergeräten (zum Beispiel der Varianzpunkt `Motorsteuergerät` mit den Alternativen `Digitale Motor Elektronik (DME)` und `Digitale Diesel Elektronik (DDE)` aufgrund von Ottomotor- und Dieselmotorfahrzeugen in der Fahrzeugproduktlinie).

4.6 Verteilung des Funktionsnetzes auf die technische Architektur

Die Beziehung zwischen Funktionsnetz und technischer Architektur wird mit Hilfe von Verteilungsmodellen dargestellt: Die Abbildung von Funktionen auf die technische Architektur gibt an, durch welches Steuergerät die entsprechende Funktionalität umgesetzt werden soll, d. h. auf welchem Steuergerät entsprechende Software installiert werden muss.

Die Verteilung von Funktionen auf die technische Architektur dient zum einen dazu, Anforderungen an die technische Architektur aufgrund von Funktionalität zu definieren, zum anderen, um frühe Bewertungen bzgl. des Zusammenspiels von Funktionsumfängen und technischer Infrastruktur vorzunehmen.

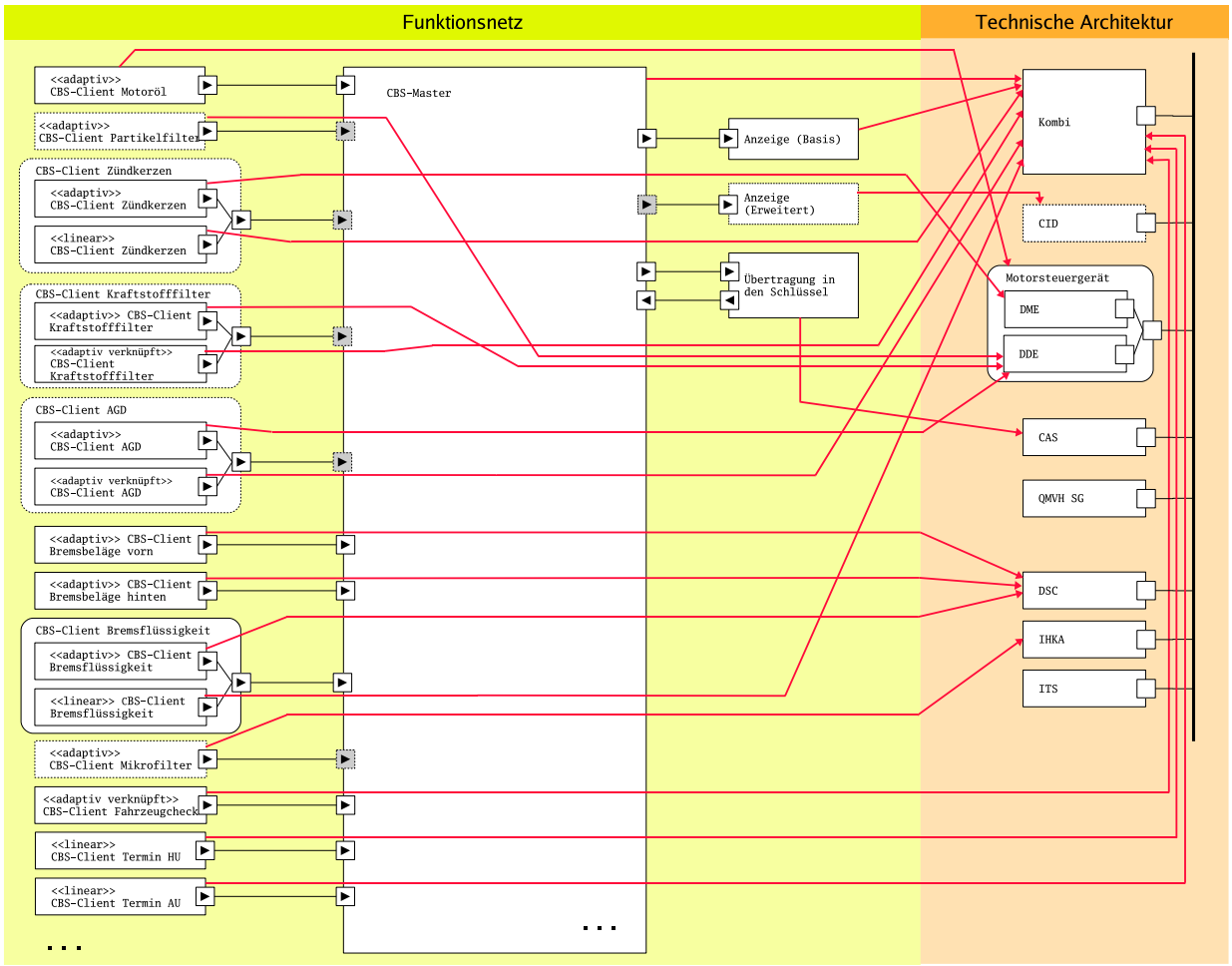
Die für die Produktlinie `L6` vorgesehene Funktionsverteilung bzgl. der CBS-Umfänge ist in Abbildung 21 dargestellt. Die prinzipiell vorgesehene Verteilung von `CBS`-Teilfunktionalitäten ist in Abbildung 22 skizziert.

In Abbildung 21 ist zu sehen, dass die Varianz im Funktionsnetz und in der technischen Architektur Auswirkungen auf die Verteilung haben kann bzw. für die Verteilung ausgenutzt werden kann. So wird die Funktion `CBS-Client Motoröl` dem Varianzpunkt `Motorsteuergerät` zugeordnet. Je nachdem, ob es sich um ein Otto- oder ein Dieselfahrzeug handelt, wird die Funktion demzufolge auf dem Steuergerät `DME` oder `DDE` untergebracht. Zündkerzen werden nur bei Ottofahrzeugen verwendet (vgl. Abbildung 14), demzufolge ist die Funktion `CBS-Client Zündkerzen` optional. Zugleich ist sie variierend (d. h. sie stellt einen funktionalen Varianzpunkt dar), da zwei unterschiedliche Verschleißermittlungsmethoden in der Produktlinie Anwendung finden sollen. Für die Funktionsverteilung wird festgelegt, dass die lineare Zündkerzenverschleißermittlung auf dem `Kombiinstrument` (`KOMBI`) unterzubringen ist, während die adaptive, d. h. eine auf Sensorik basierende Verschleißermittlung vom `DME` übernommen wird. Da diese Verteilung ein generelleres Muster ist, ist sie als solches in der prinzipiellen, musterhaften Funktionsverteilung in Abbildung 22 herausgehoben dargestellt.

Anmerkung

Ähnliche Verteilungsmodelle gibt es sowohl für die Dienstebene als auch für die Softwareebene, wobei zwischen den Verteilungsmodellen von Diensten und Funktionen sowie zwischen den Verteilungsmodellen von Funktionen und Software eine Verfeinerungsbeziehung besteht. Beispiele für Verteilungsmodelle von Diensten und von Software sind in diesem Dokument nicht angegeben.

Abbildung 21 Funktionsverteilung: Verteilung der CBS-Funktionalität auf die Topologie (L6-Umfang, Ausschnitt).

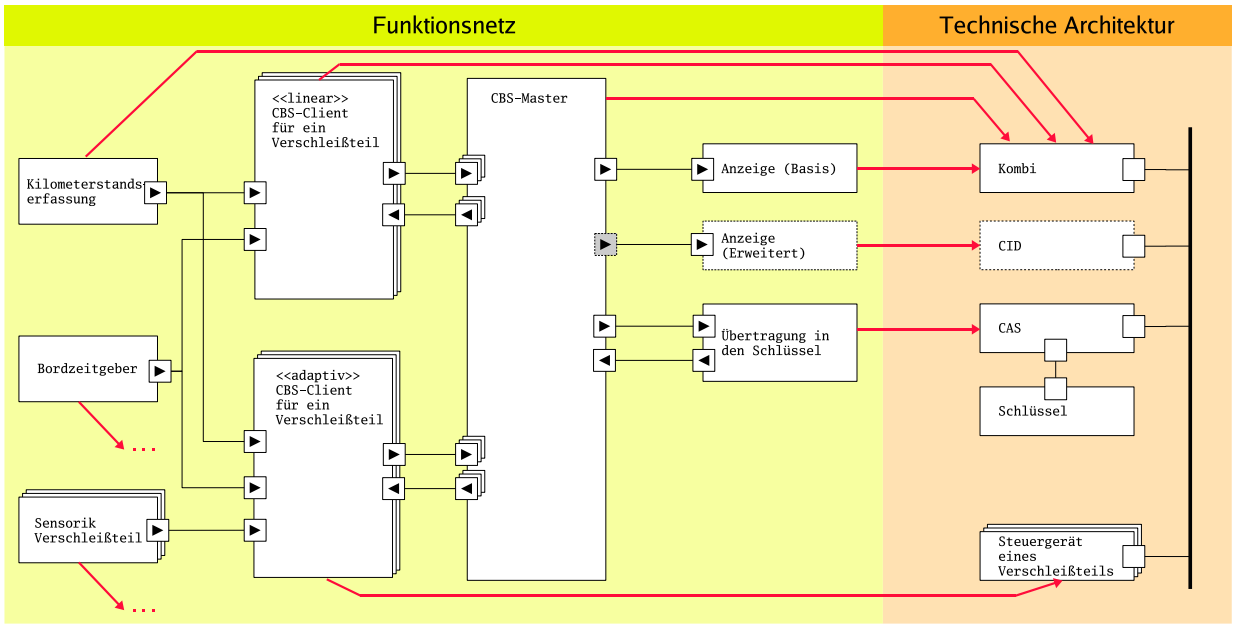


4.7 Software

Zur Veranschaulichung von Softwareaspekten sind im Folgenden Abbildungen angegeben, die auf den bisherigen Entwürfen von BMW basieren, die mit dem CASE-Tool Rhapsody erstellt wurden.

Die prinzipielle Architektur der Software (mittels Klassendiagramm), die den `CBS-Client` realisiert, ist in Abbildung 23 dargestellt. Ziel bei der Realisierung von `CBS-Clients` ist es, Softwarekomponenten unabhängig von der Steuergerätearchitektur zu erstellen. Dies wird zum Beispiel durch die Abstraktion `Rte_CBSClient` in der Schicht `Communication` vorbereitet.

Abbildung 22 Funktionsverteilung: Prinzipielle Verteilung der CBS-Funktionalität auf die Topologie (Muster).



Neben Klassendiagrammen gehören auch Verhaltensspezifikationen zu den einzelnen Klassen zur Softwarearchitekturbeschreibung. Die Verhaltensspezifikationen ermöglichen zum einen eine Simulation der Modelle, zum anderen können sie auch zur Generierung von (serienreifem) Softwarecode genutzt werden.

Abbildung 24 zeigt einen Ausschnitt aus der Verhaltensspezifikation der Klasse `CBSClient` mittels Statecharts. Abbildung 25 visualisiert den prinzipiellen Ablauf mittels Aktivitätsdiagramm für die in C implementierte Operation `recompute()` der Klasse `CBSClient`.

4.8 Artefaktzusammenhang: Varianzbetrachtungen, Verteilungsmodelle und Grundlagen für Metriken

Im Folgenden werden die zuvor illustrierten Entwicklungsartefakte im Zusammenhang diskutiert.

Im Rahmen des Forschungsprojekts Verteilte Entwicklung und Integration von Automotive-Produktlinien (VEIA) wird eine Methode erarbeitet, wie in den einzelnen Entwicklungsartefakten Varianz aufgrund des Produktlinienansatzes erfasst

Abbildung 23 Software: Prinzipielle Architektur der Software für den CBS-Client (mit Andeutung von Schichten).

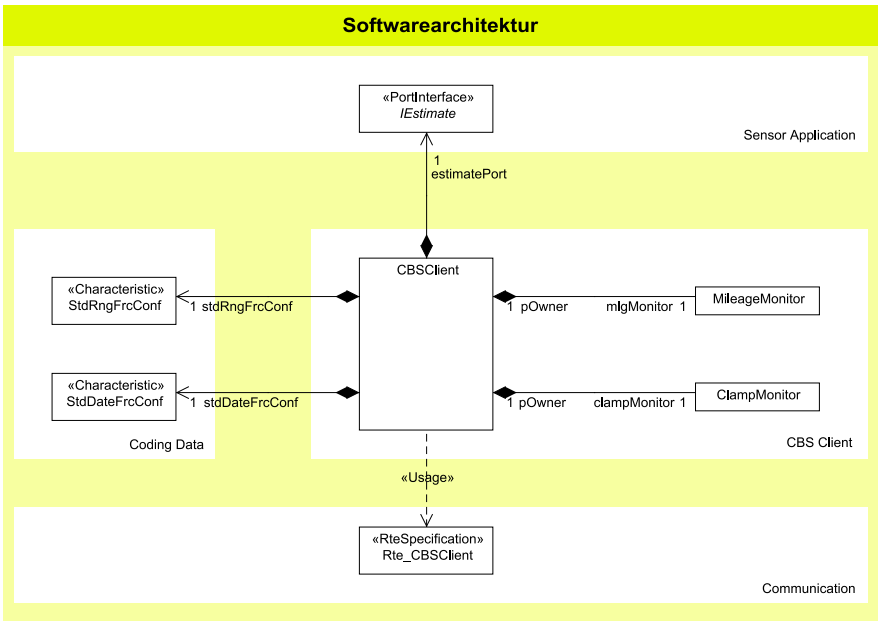


Abbildung 24 Software: Statechart der Klasse `CBSClient` (Ausschnitt).

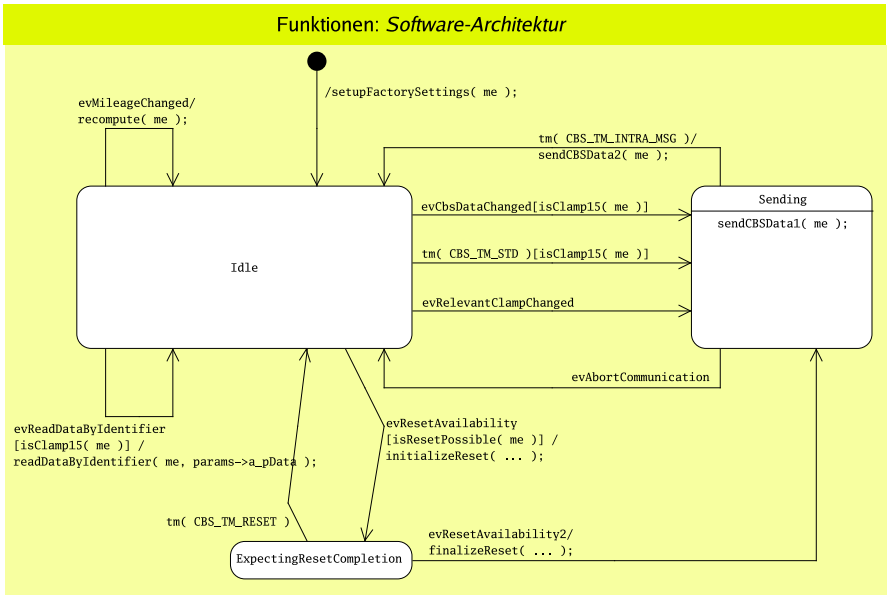


Abbildung 25 Software: Operationsbeschreibung `recompute()` der Klasse `CBSClient` (angedeutet).

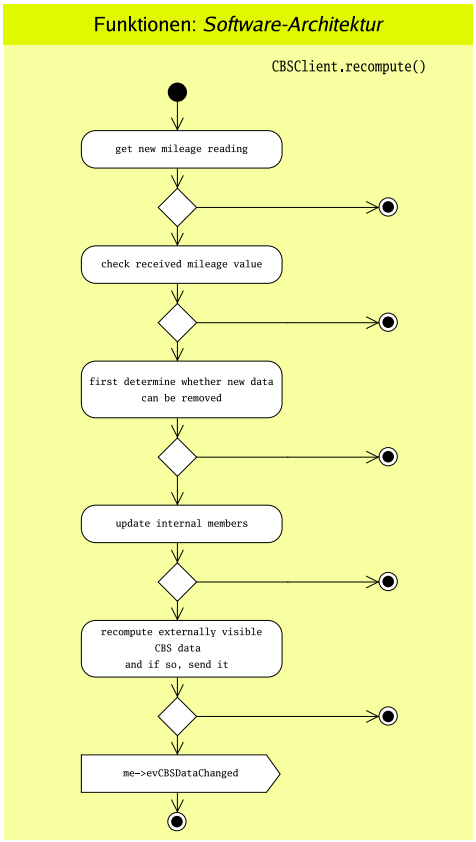
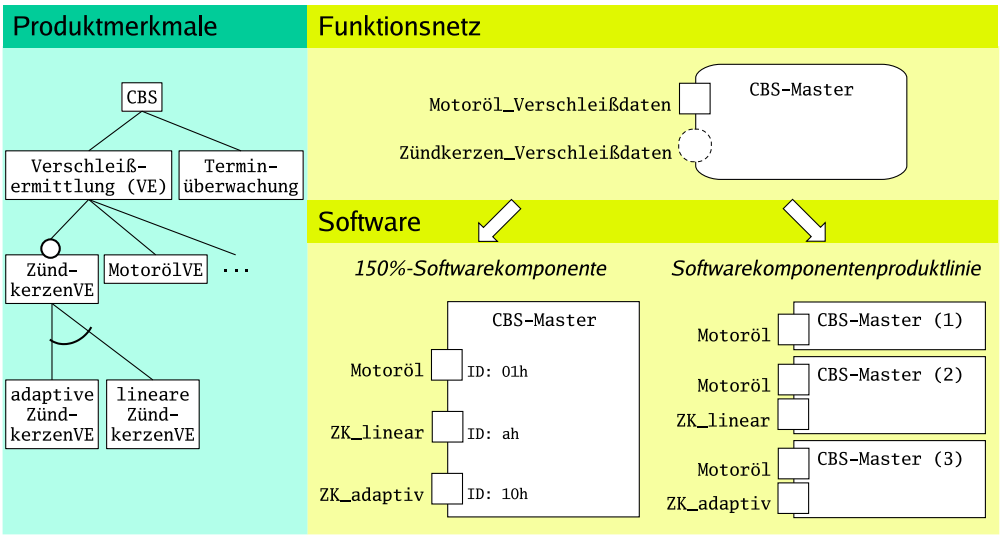


Abbildung 26 Vom Funktionsnetz zur Software.



und beschrieben wird und wie deren Zusammenhang zueinander ist. Abbildung 27 illustriert dies am Beispiel von Merkmalsmodell, Funktionsnetz und Softwarekomponente. Die unterschiedlichen CBS-Umfänge sowie die unterschiedlichen Verschleißermittlungsmethoden in den Produkten einer Produktlinie sind Ursache für die im Funktionsnetz dargestellte Varianz (in der Abbildung repräsentiert durch die variierende `CBS-Master`-Funktion mit dem optionalen und variierenden Port für die Zündkerzen-Verschleißdaten).

Prinzipiell ergeben sich in dieser dargestellten Situation zwei Umsetzungsmöglichkeiten der `CBS-Master`-Funktion durch Software:

1. mittels *150 %-Softwarekomponente* – so wie der `CBS-Master` heutzutage realisiert wird (vgl. [BMW05a, S. 46f.] für den `L6`-Umfang): sämtliche relevante Varianz aus dem Merkmalsmodell, aus dem Funktionsnetz sowie aus der technischen Architektur wird »ausmultipliziert« und in der Softwarekomponente, die den `CBS-Master` implementiert, vorgehalten,

2. mittels *Softwarekomponentenproduktlinie* – wodurch für jedes Fahrzeug / jeden Fahrzeugtyp eine spezifische Softwarekomponente erzeugt werden würde, die genau das abdeckt, was verbaut wurde.

Vorteil des *150 %-Ansatzes* ist, dass ein Fahrzeug im Nachhinein noch umgerüstet werden könnte, da entsprechende neu benötigte Funktionalität bereits installiert ist. Außerdem ergibt sich ein geringerer Aufwand in der Verwaltung

der Softwarekomponenten (es gibt nur eine). Nachteil ist sicherlich, dass die Softwarekomponente mehr Code enthält, als tatsächlich jemals benötigt wird (»toter Code«), dadurch die Softwarekomponente mehr Speicher benötigt und längere Flashzeiten verursacht. Unter Umständen ist auch ein erhöhter Parametrisierungsaufwand notwendig. Der Testaufwand ist auch komplizierter, da sichergestellt werden muss, dass kein deaktivierter Code fälschlicherweise ausgeführt wird.

Vorteil des *Softwarekomponentenansatzes* ist, dass Softwarekomponenten kleiner ausfallen, weniger »toten Code« enthalten, dadurch die Flashzeiten verkürzt werden können. Das Testen jeder einzelnen Softwarekomponente wird einfacher. Nachteil ist, dass mehr Softwarekomponenten entstehen, die verwaltet und getestet werden müssen. Außerdem muss nach Umrüstung eines Fahrzeugs unter Umständen die `CBS-Master`-Softwarekomponente ausgetauscht werden, um der neuen Fahrzeugkonfiguration gerecht zu werden.

Beide Umsetzungsmöglichkeiten stellen somit die beiden Extreme dar, die optimale Lösung liegt vielleicht dazwischen. Neben der Untersuchung, wie die Varianz auf Funktionsnetzebene idealerweise durch geeignete Softwarearchitekuren umgesetzt wird, wird im Forschungsprojekt die Bewertung solcher Umsetzungsmöglichkeiten eine wichtige Rolle spielen, um zu ermitteln, wann welche Möglichkeit die bessere ist.

Abbildung 27 Entwicklungsartefakte für `CBS` und ihr Zusammenspiel.

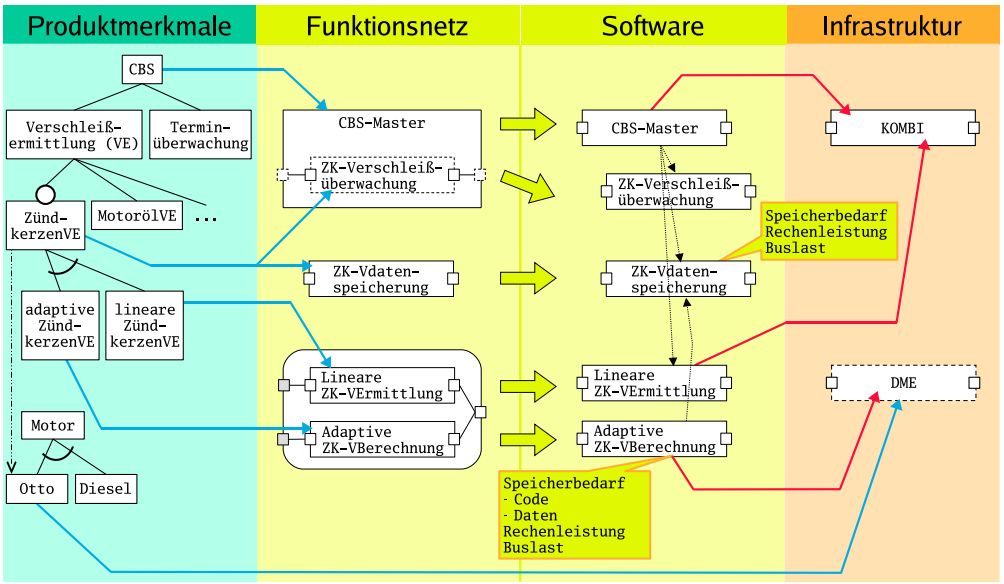


Abbildung 27 setzt das Beispiel in einem größeren Kontext fort, indem technische

Architektur und Verteilungen zusätzlich dargestellt werden. Wichtige Messgrößen auf Softwareebene, die zur Bewertung der Architektur herangezogen werden müssen, sind beispielsweise der Speicherbedarf einer Softwarekomponente aufgrund der Codegröße und aufgrund der zur Laufzeit anfallenden und abzuspeichernden Daten, die erforderliche Rechenleistung, die das Steuergerät bereitstellen muss oder die Buslast, die real oder potentiell erzeugt wird. Hat man solche Daten in einem Projekt einmal erfasst, so ist es möglich, sie auf höhere Abstraktionsebenen »zurückzuspiegeln« (zum Beispiel auf das Funktionsnetz), um eine frühestmögliche Architekturbewertung in einem Nachfolgeprojekt vornehmen zu können.

Ein Beispiel für das Resultat einer solchen Bewertung wäre, die Entscheidung für oder gegen eine Softwareproduktlinie abhängig von Speicherbedarf, Rechenleistung und Buslast treffen zu können.

5 Zusammenfassung und Ausblick

Der Grobentwurf des Referenzprozesses dient im Projekt »Verteilte Entwicklung und Integration von Automotive-Produktlinien« als Grundlage für die weiteren Arbeiten. Im Referenzprozess wurden Produktlinienbeschreibung, Dienstlandkarte, Funktionsnetz, Softwarearchitektur, technische Architektur, Konfigurationsmodelle und Verteilungsmodelle als zu erstellende Artefakte definiert.

Das zu entwickelnde System bzw. die Systemfamilie wird mit Merkmalen und Varianz in der Produktlinienbeschreibung erfasst. Die Dienstlandkarte, das Funktionsnetz sowie die Softwarearchitektur beschreiben die Funktion des Systems. Die Beschreibung der technischen Umsetzung in Hardware ist Aufgabe der technischen Architektur. Zwischen den einzelnen Artefakten dienen die Beziehungsmodelle – Konfigurations- und Verteilungsmodelle – als Bindeglied.

Damit sind verschiedene Artefakte definiert, die das zu entwickelnde System aus verschiedenen Blickwinkeln beschreiben. Die Konsistenz der Beschreibungen kann aufgrund der Querbeziehungen überprüft und zugesichert werden. Bewertungen des Systems sind über alle Artefakte möglich und damit sehr früh im Prozess verfügbar. Integration und Test werden über durchgängige Modelle und formale Beziehungen zwischen den Modellen unterstützt und können im linken Ast des V-Modells bereits vorbereitet werden.

Neben der reinen Artefaktbeschreibung wurde auch ein zeitlicher Prozess definiert, der angibt, welche Teilartefakte in welcher Reihenfolge zu erstellen sind. Dabei wurden sechs Phasen unterschieden, von denen die vier ersten genauer erläutert wurden. Die ersten vier Phasen finden beim OEM statt, die fünfte Phase beim Zulieferer. Die sechste Phase – Integration und Test im rechten Ast des V-Modells – ist wieder Aufgabe des OEM.

Im weiteren Verlauf von VEIA schließen sich vier Teilprojekte an: Anforderungsmodellierung, Architekturmodellierung, Modellmanagement und Werkzeugunterstützung. In diesen Teilprojekten entstehen Methoden, Notationen und Werkzeuge, um die definierten Artefakte zu erstellen, also um den Referenzprozess durchführen zu können. Fallstudien begleiten den weiteren Verlauf, um die erzielten Ergebnisse praxisnah evaluieren zu können.

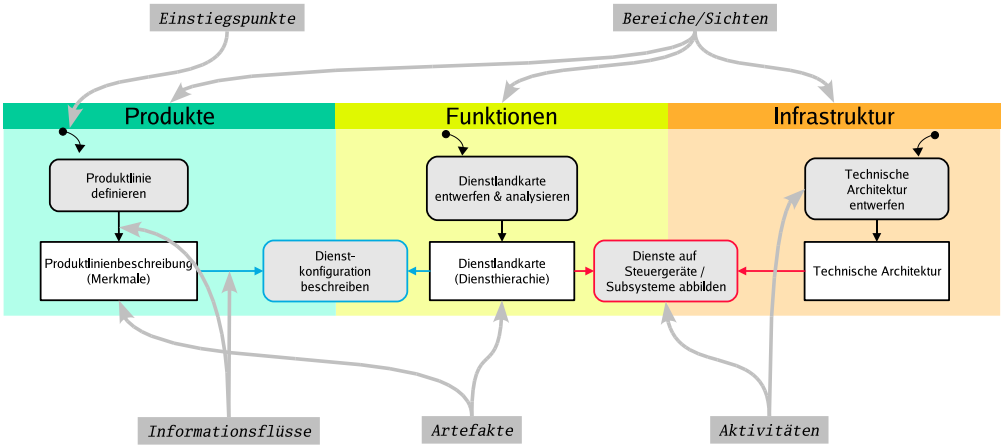
Notation

Nachfolgend werden die im Dokument neu verwendeten Notationen beschrieben. Die bereits in MOSES eingeführten Notationen sowie allgemein bekannte Elemente z. B. aus der Unified Modeling Language (UML) werden nicht noch einmal erläutert.

Artefaktbeschreibungen

Eine Übersicht der Elemente von Artefaktbeschreibungen zeigt Abbildung 28.

Abbildung 28 Artefaktbeschreibungen (Notation).



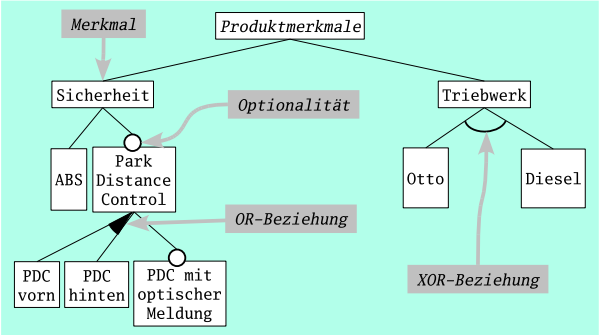
Vertikal angeordnet sind die Bereiche bzw. Sichten, die für ein E/E-System relevant sind. Artefakte werden durch weiße, eckige Kästen symbolisiert, Aktivitäten als graue Kästen mit abgerundeten Ecken. Der Informationsfluss von Aktivität zu Artefakt und umgekehrt wird durch Pfeile dargestellt. Analog zu Statecharts der UML dienen ausgefüllte Kreise mit Pfeilen als Darstellung der Einstiegspunkte in die Modellierung.

Produktlinienmodelle

Eine Übersicht der Elemente von Produktlinienmodellbeschreibungen zeigt Abbildung 29.

Die Produktmerkmale werden als weiße Kästen gezeichnet. Ein Strich zwischen

Abbildung 29 Produktlinienmodelle (Notation).



zwei Produktmerkmalen bedeutet: das oben stehende Merkmal setzt sich aus dem unten stehenden Merkmal zusammen (oder aus mehreren Merkmalen). Für optionale Merkmale wird ein Kreis an die Verbindung angefügt. Mehrere Untermerkmale können in einer OR- oder XOR-Beziehung stehen. Eine OR-Beziehung besagt, dass kein oder beliebig viele Merkmale genutzt werden können, bei der XOR-Beziehung muss genau ein Merkmal genommen werden.

Akronyme

AGD Ansauggeräuschdämpfer: reduziert das Ansauggeräusch eines Motors.

AU Abgasuntersuchung: gesetzlich vorgeschriebene Untersuchung, die sicherstellt, dass die Abgaswerte zugelassener Kraftfahrzeuge über den Nutzungszeitraum innerhalb der festgelegten Abgasnorm bleiben. Periodische Untersuchung mit festgelegten Zeiträumen.

AUTOSAR Automotive Open System Architecture: offene, standardisierte Softwarearchitektur für die Automobil-Industrie.

BOS Bedarfsorientierter Service: alter Name für die Systemfunktion CBS

CAS Car Access System: Steuergerät, das den Fahrzeugzugang steuert.

CBS Condition Based Service: Systemfunktion für die Bewertung des Verschleißes von Fahrzeugressourcen wie Motoröl, Zündkerzen etc. und die optimale Berechnung für den Werkstättenbesuch. CBS hieß bei BMW vorher BOS

CID Central Information Display: zentrale Anzeige in der Mittelkonsole.

DDE Digitale Diesel Elektronik: Motorsteuerungselektronik für Diesel-Motoren

DKG Doppelkupplungsgetriebe: spezielles Getriebe, das schnelleres Schalten ermöglicht. Für CBS wird das DKG-Getriebeöl überwacht.

DME Digitale Motor Elektronik: Motorsteuerungselektronik für Otto-Motoren

DSC Dynamische Stabilitäts-Control: optimiert sowohl die Fahrstabilität beim Anfahren und Beschleunigen als auch die Traktion.

EDC Elektronische Dämpfercontrol: erhöht den Fahrkomfort und die Fahrzeugsicherheit, indem sie die Dämpferkraft stufenlos einstellt und an die wechselnden Straßen-, Beladungs- und Fahrverhältnisse anpasst.

HU Hauptuntersuchung: soll die Mängelfreiheit von Verkehrsmitteln sicherstellen. Periodische Untersuchung mit festgelegten Zeiträumen.

IHKA Integrierte Heiz-Klima-Automatik: Automatikausführung einer Heizungsregelung

IHR Integrierte Heizungsregelung: Standardausführung einer Heizungsregelung.

ITS Intelligent Tire System: Steuergerät, das den Reifenverschleiß verarbeitet. (Quelle: http://www.conti-online.com/)

KOMBI Kombiinstrument: Anzeige, die zwischen Tachometer und Drehzahlmesser liegt.

MOSES Modellbasierte Systementwicklung: FuE-Projekt des Fraunhofer ISST im Auftrag der BMW AG, vgl. [Kle06].

MOST Media Oriented Systems Transport: serielles Bussystem zur Übertragung von Audio- und Video-, Sprach- und Datensignalen über Lichtwellenleiter oder über elektrische Leiter.

NO_x Stickoxid: Abgase, die bei der Verbrennung fossiler Brennstoffe entstehen. Für CBS werden NO_x-Additive überwacht.

OBD On-Board-Diagnose: Funktion der DME, die Störungen erkennen soll, bevor sie Schaden anrichten. Dabei werden Fehlermeldungen gespeichert und später auf dem Bildschirm des BMW Diagnose-Informations-Systems in der Werkstatt sichtbar gemacht.

PEP Produktentstehungsprozess: umfasst die betrieblichen Abläufe in den Bereichen Entwicklung, Konstruktion, Arbeitsplanung, Fertigung, Montage, Qualitätssicherung. (Quelle: CAD Web <http://www.blien.de/ralf/cad/db/pr_entst.htm>)

QMVH Quermomentenverteilung im Heck: teilaktive Hinterachskinematik, beeinflusst die Verteilung der Querkräfte über die Längsachse. Für CBS wird das Hinterachsgetriebeöl überwacht.

SAM Service Annahme Modul: Steuergerät, das u. a. CBS-Daten ausliest und anzeigt.

UML Unified Modeling Language: von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software und anderen Systemen.

VEIA Verteilte Entwicklung und Integration von Automotive-Produktlinien: vom BMBF im Rahmen der Forschungsoffensive »Software Engineering 2006«gefördertes Verbundprojekt

Folgende Akronyme treten in Abbildungen auf, nicht im Text: AU, AGD, CAS, DKG, DSC, HU, IHKA, IHR, ITS, NO_x, QMVH, SAM

Literatur

[BMW05a] BMW Group: *Lastenheft CBS 5*. Oktober 2005. – SAP-Doknr.: 10000943-000-03

[BMW05b] BMW Service: Technik: *Condition Based Service: E87, E90, E91*. Mai 2005

[BMW06] BMW Group: *BMW 7er-Produktkatalog*. September 2006. http://www.bmw.de/de/produkte/automobiles/7er/

[Bro05] *Broy*, Manfred: Service-oriented Systems Engineering: Specification and Design of Services and Layered Architectures. The Janus Approach. In: *Engineering Theories of Software Intensive Systems*, Springer, 2005, S. 47–81

[BS01] *Broy*, Manfred ; *Stølen*, Ketil: *Specification and development of interactive systems: Focus on streams, interfaces, and refinement*. New York : Springer-Verlag, 2001

[CE00] *Czarnecki*, Krzysztof ; *Eisenecker*, Ulrich W.: *Generative Programming – Methods, Tools and Applications*. Addison-Wesley, 2000 (Pearson Education)

[Fra03a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *MOSES 2: Modellierung von hierarchischen, vernetzten Funktionen. Eine Methodik für die BMW Group*. Juli 2003. – Projektabschlussbericht für BMW Group

[Fra03b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *MOSES 3.1: Architekturmodellierung, Teil 1, Metamodell für technische Architekturmodelle und Partitionierung von logischen Funktionen*. September 2003. – Projektabschlussbericht für BMW Group

[Fra04a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *MOSES 3.2: Architekturmodellierung, Teil 2, Erweiterte Modellierung und Bewertung von Partitionierungen*. Januar 2004. – Projektabschlussbericht für BMW Group

[Fra04b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *MOSES 4.1: Validierung und Umsetzung der modellbasierten Entwicklungsmethodik MOSES, Teil 1: Konfiguration, Produktlinien- und Domänenmodelle für Funktionsnetzwerke*. September 2004. – Projektabschlussbericht für BMW Group

[Fra05a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *Das MOSES-Gesamtmetamodell*. April 2005. – Projektabschlussbericht für BMW Group

[Fra05b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme: *MOSES 4.2: Validierung und Umsetzung der modellbasierten Entwicklungsmethodik MOSES, Teil 2: Produktlinien- und Domänenmodelle technischer Architekturen*. April 2005. – Projektabschlussbericht für BMW Group

[GHHJ06] *Gruler*, Alexander ; *Harhurin*, Alexander ; *Hartmann*, Judith ; *Jürgens*, Elmar: *VEIA – Ebenenhierarchie*. Juli 2006. – Draft, Juli 2006

[GR06] *Große-Rhode*, Martin: *Verteilte Entwicklung und Integration von Automotive-Produktlinien (VEIA)*. Projektantrag, Mai 2006. – Projektantrag im Rahmen der BMBF-Forschungsoffensive »Software Engineering 2006«

[KCH+90] *Kang*, Kyo C. ; *Cohen*, Sholom G. ; *Hess*, James A. ; *Novak*, William E. ; *Peterson*, A. S.: Feature-oriented domain analysis (FODA) – feasibility study / Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213. 1990. – Technical Report CMU/SEI-90-TR-21

[Kle06] *Kleinod*, Ekkart: Modellbasierte Systementwicklung in der Automobilindustrie – Das MOSES-Projekt / Fraunhofer-Institut für Software- und Systemtechnik, Berlin, Abteilung Verlässliche Technische Systeme. – ISST-Bericht 77/06. http://www.isst.fraunhofer.de/deutsch/download/10095_moses_gesamt.pdf

[KMM05] *Krüger*, Ingolf H. ; *Mathew*, Reena ; *Meisinger*, Michael: From Scenarios to Aspects: Exploring Product Lines. In: *Proc. 4th Int. Workshop on Scenarios and State Machines: Models, Algorithms and Tools*, 2005

[KSTW04] *Kof*, Leonid ; *Schätz*, Bernhard ; *Thaler*, Ingomar ; *Wisspeintner*, Alexander: Service-based development of embedded systems. In: *Net.Object Days Conference, OOSE Workshop, Erfurt, Germany*, 2004

[PBL05] *Pohl*, Klaus ; *Böckle*, Günter ; *Linden*, Frank J. van d.: *Software Product Line Engineering – Foundations, Principles and Techniques*. Springer-Verlag <http://www.software-productline.com/>

[SS03] *Schätz*, Bernhard ; *Salzmann*, Christian: Service-Based Systems Engineering: Consistent Combination of Services. In: *Proc. ICFEM 2003, 5th Int. Conf. on Formal Engineering Methods*, Springer, 2003 (LNCS 2885)