Entwicklung einer Software Produktlinie zur flexiblen Erstellung von Umfeldwahrnehmungssystemen

Sebastian Ohl
Markus Maurer

Braunschweig : Institut für Regelungstechnik, 2011

Veröffentlicht: 07.03.2011

http://www.digibib.tu-bs.de/?docid=00038402

Entwicklung einer Software Produktlinie zur flexiblen Erstellung von Umfeldwahrnehmungssystemen

Dipl.-Inform. Sebastian Ohl

Prof. Dr.-Ing. Markus Maurer

12. Januar 2010

1 Einleitung

In der aktuellen Entwicklung von Fahrerassistenzsystemen ist der Trend zu immer komplexeren und gleichzeitig flexibleren Fahrerassistenzsystemen zu beobachten (Maurer, 2009). Der Einsatz von Systemen in unterschiedlichen Projekten und Fahrzeugklassen führt zu neuen Herausforderungen für die maschinelle Wahrnehmung. Diese äußert sich, neben den funktionalen Erweiterungen, in einem erheblich größeren Wartungs- und Entwicklungsaufwand. Durch Wiederverwendung von einzelnen Komponenten steigt zudem der Integrations- und Testaufwand erheblich, da unterschiedliche, möglicherweise projektspezifische Anforderungen, umgesetzt werden müssen. Ein System zur flexiblen Zusammenstellung von Komponenten zu neuen, speziell auf das Projekt angepassten Produkten erscheint hier sinnvoll und nötig, um Auswirkungen von Änderungen an Komponenten unmittelbar abschätzen, gezielt umsetzen zu können und so die Komplexität beherrschbar zu machen.

Umfeldwahrnehmungssysteme bilden für viele Fahrerassistenzsysteme die Basis, auf der Entscheidungen getroffen werden. Eine Umfeldwahrnehmung verarbeitet Daten von Umfeldsensoren und bereitet diese für die eigentliche Applikation geeignet auf. Beispielsweise werden nicht messbare Größen bestimmt oder mehrere Sensoren zu einem konsistenten Bild des Umfelds fusioniert. Durch die Eigenschaft als Konvertierungsinstanz von Sensordaten hin zu Eingangsdaten der Applikation ist es möglich, einheitliche Schnittstellen definieren und so den Einsatz in unterschiedlichen Projekten erheblich vereinfachen.

Der Einsatz einer einheitlichen Umfeldwahrnehmung, welche alle Sensoren und Applikationen gleichzeitig bedienen kann, stellt eine große Herausforderung für die Entwicklung dar. Einfacher erscheint es, eine Umfeldwahrnehmung in ,,kleine" Einzelteile (Architekturkomponenten) aufzuteilen und nur die für die Funktion notwendigen Komponenten in eine speziell für eine Applikation zusammengestellte Umfeldwahrnehmung einzubringen.

Die Schwierigkeit liegt damit jedoch in der Definition der Komponentenaufteilung, der Schnittstellen zwischen den Komponenten sowie in der Zusammenstellung der Komponenten zu einem konkreten Produkt. Software Produktlinien bieten eine Lösungsmöglichkeit für diese Problemstellung.

Als Software Produktlinie wird die Zusammenfassung mehrerer individueller Instanzen eines Softwareprodukts einer gemeinsamen Basis bezeichnet. Eine Software Produktlinie ist dabei die Anwendung von Produktlinien, wie sie bereits seit langem beispielsweise im Automobilbau eingesetzt werden, auf Softwareprodukte. Ähnlich wie bei verschiedenen Modellvarianten einer Fahrzeugplattform wird hier mit Hilfe von definierten Regeln und klar definierten Schnittstellen aus Architekturkomponenten ein individuell angepasstes Produkt zusammengestellt. Ferner ist die Umsetzung spezieller Anforderungen durch einzelne, projektspezifische Komponenten möglich, die anstelle von Komponenten der Produktlinie eingesetzt werden.

Fallstudien, wie z.B. von der Robert Bosch GmbH (Botterweck u. a., 2008; Thiel und Hein, 2002) im CAFÉ Projekt (Chastek, 2002) oder Beiträge der Daimler AG (Buhr und Kolagari, 2004), haben gezeigt, dass sich Software Produktlinien im Fahrzeugumfeld einsetzen lassen, um die Wiederverwendung von Komponenten zu steigern.

In diesem Beitrag wird eine Architektur für Umfeldwahrnehmungssysteme im Kraftfahrzeug vorgestellt, die diesen Ansatz anwendet. In Abschnitt 2 wird ein kurzer Überblick über Software Produktlinien gegeben. Anschließend wird in Abschnitt 3 die Systemarchitektur vorgestellt und im Folgenden die Zusammenstellung von Produkten dargestellt. Ein Fallbeispiel und ein Ausblick schließen den Beitrag ab.

2 Software Produktlinien

In den 1990er Jahren wurde vor allem Software für einzelne Projekte (Individualsoftware) entwickelt. Diese wurde speziell auf den Kunden zugeschnitten und hatte somit nur

einen geringen Wiederverwendungswert. Die Entwicklung dieser Art von Software wurde aus Kostengründen erheblich zurückgefahren und Software für einen großen Nutzerkreis, sogenannte Standardsoftware, wurde der vorherrschende Produkttyp. Durch die damit verbundene Massenfertigung wurden Entwicklungs- und Wartungskosten erheblich reduziert. Leider wurde mit der weitgehenden Aufgabe der Entwicklung von Individualsoftware auch die Möglichkeit zur Anpassung an spezielle Domänen aufgeben.

Auch hier bieten Software Produktlinien bieten einen vielversprechenden Ansatz, um Individualsoftware mit der Idee von Standardsoftware zu kombinieren. Hierbei werden Standardkomponenten(Architekturkomponenten) entwickelt, mit denen sich ein individuelles Produkt zusammenstellen lässt. Die Entwicklung einer Software Produktlinie basiert auf zwei fundamentalen Prinzipien:

- der Trennung von Domain- und Applicationengineering
- der expliziten Beschreibung von Variabilität

Unter Domäne wird ein Bereich verstanden, der gut wiederverwendbare Funktionalität in einem Teilbereich der Produktlinie enthält. Ein Produkt ist letztendlich eine konkrete Zusammenstellung einer Produktlinie für ein spezielles Projekt. Sowohl dem Domain- als auch dem Applicationengineering gemein sind die Phasen des in der Organisation gelebten Entwicklungsprozesses (z.B. (IABG, 2009), (Kruchten, 1999)). Im Domainengineering werden die gemeinsamen und variablen Bestandteile einer Produktlinie entworfen und entwickelt. Hierbei entstehen Architekturkomponenten, welche an bestimmten Punkten, den sogenannten Variationspunkten, verändert werden können. Die Herausforderung im Domainengineering liegt in der Abbildung aller relevanten Teile der Domäne in der Architektur. In Abschnitt 3 wird dieser Teil bezogen auf die Produktlinie Umfeldwahrnehmungssoftware vorgestellt.

Hingegen werden im Applicationengineering einzelne konkrete Produkte für ein spezielles Projekt erstellt. Diese setzen sich größtenteils aus den im Domainengineering entwickelten Komponenten zusammen und besitzen nur einen geringen Teil an individuell für das Projekt erstellten Komponenten.

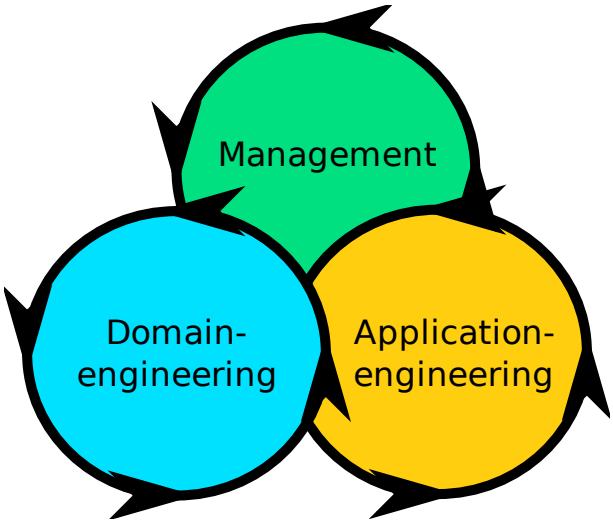

Abbildung 1: Softwareproduktlinienprozess (Northrop und et al., 2007)

Das Domain- und das Applicationengineering werden durch eine Managementkomponente (siehe Abbildung 1) verbunden. Diese koordiniert die beiden Bereiche und sorgt so für eine zielgerichtete Entwicklung der Produktlinie. Werden durch Projekte neue Anforderungen an die Produktentwickler herangetragen, so ist es Aufgabe des Managements zu entscheiden, ob diese Teil der Produktlinie werden oder lediglich Teil des projektspezifischen Produkts bleiben. Ferner lässt sich durch die Trennung von Domain- und Applicationengineering beispielsweise eine proaktive Evolution der Produktlinie umsetzen (Böckle u. a., 2004). Hierbei können unabhängig von Projekten Komponenten entwickelt und erweitert werden, um zukünftigen Anforderungen zu entsprechen.

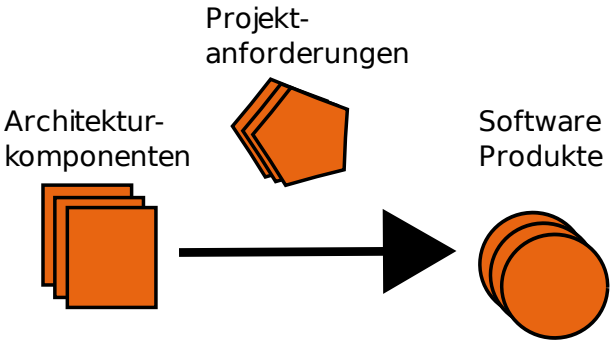

Abbildung 2: Erstellung von konkreten Produkten

Zur Erstellung von konkreten Produkten werden die Architekturkomponenten mit den Anforderungen des Projekts kombiniert (siehe Abbildung 2). An den durch das Domai-

nengineering vorgesehenen Variationspunkten werden entsprechend der Anforderungen Entscheidungen getroffen, so dass ein an das Projekt angepasstes Produkt entsteht. In der Regel fordert der Entwicklungsprozess, dass jede Architekturkomponente durch eigene Testfälle abgesichert ist. Somit sind nur noch Tests der Kombinationen der Komponenten notwendig. Dies vereinfacht auch die Entwicklung von neuen Funktionen, die, wenn sie Teil der Produktlinie sind und innerhalb einer Architekturkomponente entwickelt wurden, ohne großen Aufwand allen Produkten zugänglich gemacht werden können, die diese Komponente enthalten.

3 Architektur des Systems

Die Architektur des Umfeldwahrnehmungssystems ist die Basis für die Anwendung des Systems als Software Produktlinie. Dazu ist das System in mehrere Architekturkomponenten aufgeteilt (siehe Abbildung 3), die über eine Pipeline-Architektur (Buschmann u. a., 1996) kommunizieren.

Die Aufteilung in Komponenten leitet sich vor allem aus den Beziehungen zu anderen Komponenten ab. Hier sind zunächst die einzelnen Sensoren als einzelne Komponenten beschrieben. Sie sind unabhängig voneinander entwickelbar und kommunizieren mit dem Rest des Systems nur über eine definierte Schnittstelle. Die zentrale Komponente des Umfeldwahrnehmungssystems ist eine objektbasierte Sensordatenfusion, die in ihrem internen Ablauf diverse Variationspunkte besitzt.

3.1 Architekturkomponente: Sensordatendekoder

Zur Anbindung der Sensoren wird zunächst eine Abstraktion vom Bustyp (z.B. CAN oder Ethernet) durchgeführt. Dies ermöglicht, die Dekodierung von Daten von Sensoren, die in der Lage sind, diese über unterschiedliche Bustypen zu versenden, nur eimal zu implementieren. Auf diese Weise können die Komponenten in unterschiedlichen Netzwerklayouts eingesetzt werden.

Die Aufgabe eines Sensordatendekoders ist die Konvertierung des Sensorprotokolls an eine einheitliche Schnittstelle. Für Sensoren, die bereits eine Objekthypothesenbildung durchführen, wird als Schnittstelle eine Objekthypothesenliste vorgesehen. Diese kann unterschiedliche Zustandsvektoren aufnehmen. Je nach Sensortyp können so beispielsweise

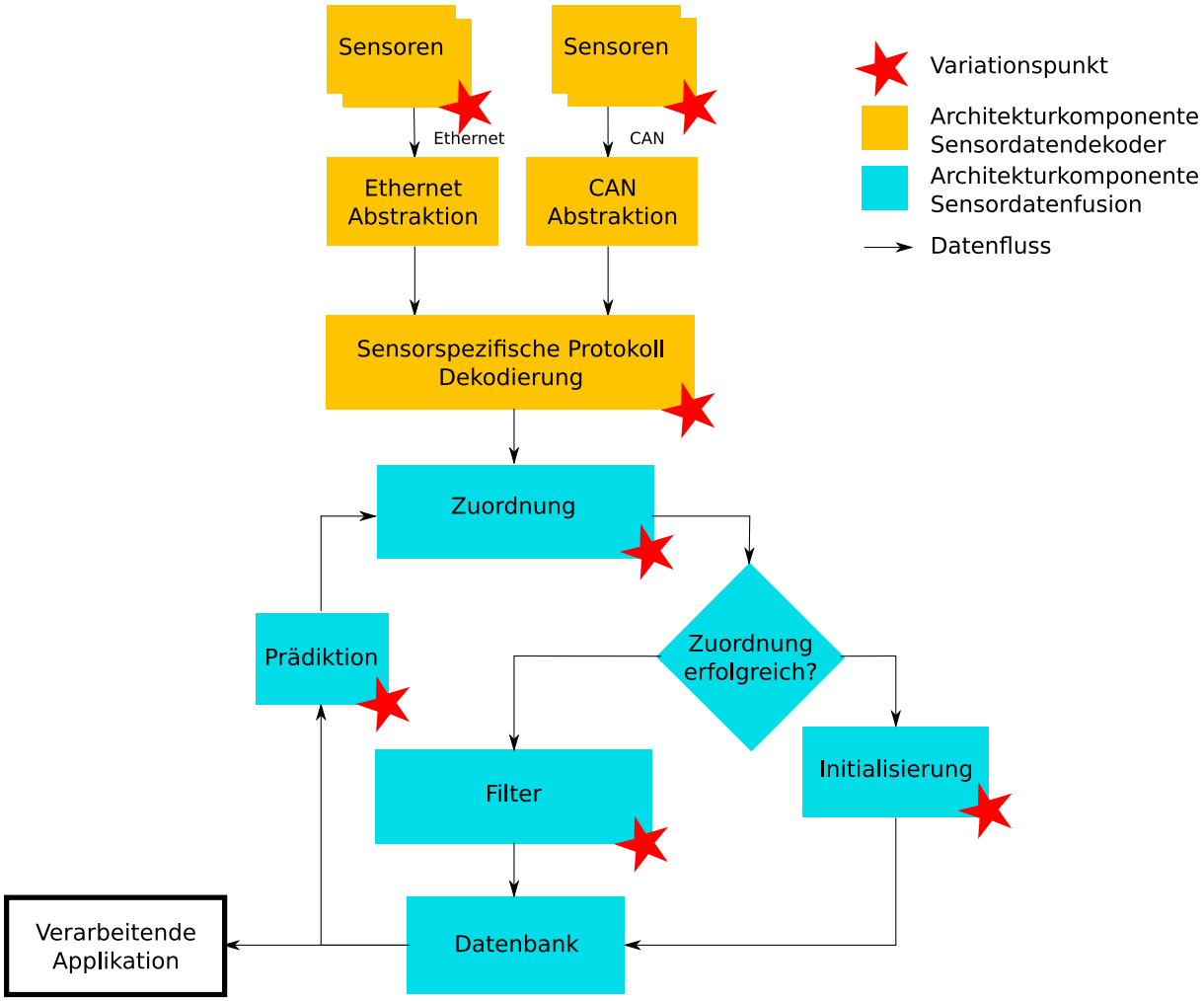

Abbildung 3: Architektur aus Sicht des Datenflusses. Das Objekthypothesenmodell stellt einen weiteren Variationspunkt dar.

Objekthypothesen als x-y-Koordinate oder auch als offener Polygonzug an die weiterarbeitenden Stufen übergeben werden. Für Sensoren, die keine Objekthypothesenbildung durchführen, wird eine einfache sortierte Punktliste festgelegt.

3.2 Architekturkomponente: Objektbasierte Sensordatenfusion

Die objektbasierte Sensordatenfusion (SDF) ist eine klassische pipelinebasierte SDF (Effertz, 2009) bestehend aus Prädiktion, Zuordnung, Filterung und Trackinitialisierung. Hierbei wird eine größtmögliche Unabhängigkeit zwischen den einzelnen Elementen sichergestellt, um eine Anpassung an viele unterschiedliche Anwendungsfälle sicherzustellen.

Die vier Grundelemente sowie das verwendete Objekthypothesenmodell stellen Varia-

tionspunkte innerhalb der Komponente dar. Welche konkreten Elemente letztendlich in dem Produkt vorhanden sind, hängt maßgeblich von den Anforderungen des Projekts und den vorhandenen Sensoren ab. Ein Feature-Modell zur Zusammenstellung sowie konkrete Beispiele werden in den nächsten Abschnitten vorgestellt.

Folgende Variationspunkte sind Teil der SDF:

- Die Initialisierung von neuen Tracks ist in der Lage, aus jedem Eingangssensordatum das Modell der SDF zu erzeugen. Hier können auch weitere Daten (z.B. Kartendaten oder Inertialdaten) hinzugezogen werden, um beispielsweise fehlende Messdaten (z.B. Geschwindigkeiten) zu ersetzen.

- Die Prädiktion bildet einen Track auf den aktuellen Messzeitpunkt ab und selektiert beispielsweise anhand der Sichtbereiche des Sensors eine Menge von Tracks für die Zuordnung aus der aktuellen Trackdatenbank.

- Die Zuordnung erhält eine Menge von Messdaten sowie das Ergebnis der Prädiktion und erzeugt daraus eine Menge von Tupeln aus Messdatum und Track sowie eine Menge von nicht zugeordneten Messdaten.

- Die durch die Zuordnung erzeugten Tupel werden mit Hilfe eines Filters zusammengeführt und das Ergebnis in der Trackdatenbank gespeichert.

- Das Modell der durch die Sensordatenfusion beschriebenen Objekthypothesen bildet die Ausgangsschnittstelle zur Applikation.

Für jeden der Variationspunkte gibt es eine Mehrzahl von Implementierungen, die je nach Anforderung eingesetzt und miteinander kombiniert werden können.

4 Zusammenstellung eines konkreten Produkts

Die in dem vorangegangenen Abschnitt eingeführten Architekturkomponenten müssen zur Erstellung einer konkreten Ausprägung als Produkt geeignet kombiniert werden. Die anzuwendenden Regeln lassen sich durch ein Feature-Modell (Schobbens u. a., 2006) beschreiben. Hier wird ein einfacher gerichteter Graph aufgestellt, dessen Kanten jeweils zu logischen Operationen zusammengefasst werden können. Die Zusammenstellung ergibt

sich schließlich aus der Selektion einzelner Kanten, welche durch die Anforderungen der Applikation und des Fahrzeugs gegeben sind.

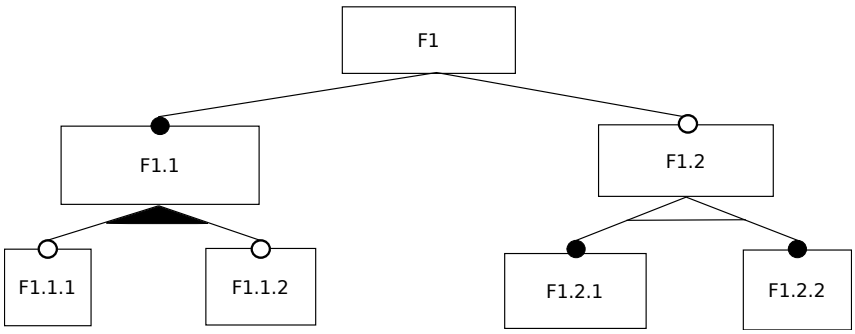

Abbildung 4: Notation eines Feature-Modells

Ein Feature-Modell ist ein gerichteter Graph, der Knoten zur Gruppierung nutzt (siehe Abbildung 4). Über Verbindungen zwischen den Knoten können OR- (ausgefüllte Dreiecke) oder XOR(Nicht gefüllte Dreiecke) beschrieben werden. Zusätzlich geben Kreise am Ende einer Kante optionale Komponenten (nicht ausgefüllter Kreis) oder verpflichtende Komponenten (ausgefüllter Kreis) an. Im Vergleich zu Gerhard Goos und van Leeuwen (2004) werden in diesem Beitrag nicht nur Baum- sondern auch zyklenbehaftete Graphen zugelassen, solange mindestens ein Knoten keine Eingangskanten besitzt.

Um ein Produkt zusammenzustellen, wird die nötige Menge von Kanten ausgewählt. Es gibt zwei Arten von Auswahlen: Selektionen und Vorselektionen. Eine Selektion ist ein Element, das zur Umsetzung der Funktion unbedingt im System vorhanden sein muss. Eine Vorselektion ist ein Element, für das es andere Möglichkeiten der Umsetzung gibt. Hier ist später eine Abwägung zu treffen.

Zur Auswertung des Graphen werden zunächst Kanten selektiert, um Anforderungen der Applikation oder beispielsweise zu nutzende Sensoren im Fahrzeug zu beschreiben. Anschließend wird beginnend von Knoten ohne Eingangskanten mit einer Tiefensuche begonnen und entsprechend der Typen der Kanten weitere Kanten selektiert oder vorselektiert bis keine Auswahlen mehr möglich sind.

Sind nach Anwendung des Algorithmus nur Selektionen durchgeführt worden, so ist ein konkretes Produkt erzeugt worden. Wurden jedoch auch Vorselektionen durchgeführt, so ist eine Auswahl innerhalb der Regeln des Abhängigkeitsgraphen zu treffen. Hierzu wird die Menge aller möglichen Permutationen der Vorselektionen gebildet, welche den

Abhängigkeitsgraphen erfüllen. Jedes dieser Produkte erfüllt die durch Anwendung und Rahmenbedingungen vorgegebenen Anforderungen. Um eine fundierte Auswahl der potentiellen Produkte in dieser Menge zu finden, können Metriken angewendet werden, welche die Qualität der Produkte abschätzen. Das Produkt mit der höchsten Qualität bezogen auf die Anforderungen der Applikation wird schließlich die gesuchte Ausprägung des Produkts sein.

Im Folgenden werden zwei Metriken vorgestellt, mit denen eine Auswahl der Vorselektionen möglich ist. Es lassen sich auch andere, beispielsweise projektspezifische Metriken definieren, um diesen letzten Schritt durchzuführen.

Als Qualität wird hier die Komplexität der einzelnen Algorithmen gewählt. Um eine möglichst hohe Qualität zu erzielen, ist Abdeckung der Anforderungen bei einer geringen Komplexität wünschenswert.

Die Komplexität O der SDF bestimmt sich aus der Summe der Komplexitäten der Einzelverarbeitungsstufen (Prädiktion O_p, Assoziation O_a, Filterung O_f und Initialisierung O_i).

Die Bestimmung der Komplexität der Assoziation und der Initialisierung hängt maßgeblich von den eingesetzten Sensoren ab. Die Assoziation muss im aufwändigsten Fall die maximale Anzahl der durch die Sensoren gelieferten Objekthypothesen N_s mit der maximal durch die SDF verarbeitbaren Objekthypothesen N_{sdf} assoziieren. Die Wahrscheinlichkeit für eine neu erzeugte Objekthypothese eines Sensors P_s bestimmt die Komplexität der Initialisierung. Die Komplexität des Filters hängt vor allem von der Anzahl der maximal verarbeitbaren Objekthypothesen sowie von der Anzahl der gehaltenen Größen im Zustandsvektor N_x ab. Die Gesamtkomplexität ergibt sich damit zu:

$$O = O_f \cdot N_{sdf} \cdot N_x + O_a \cdot \sum_{s \in Sensoren} (N_s) \cdot N_{sdf} + \prod_{s \in Sensoren} (P_s) \cdot O_i$$

5 Fallbeispiel

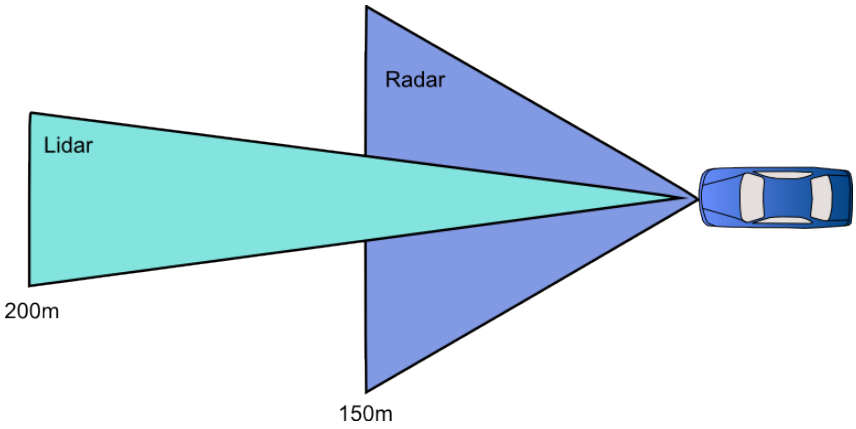

Abbildung 5: Sensorabdeckung des Fallbeispiels

Im Folgenden soll das Verfahren an einem einfachen Beispiel vorgestellt werden. Hierzu werden zwei Projekte betrachtet, die auf dem gleichen Versuchsträger betrieben werden sollen. Der Versuchsträger ist mit einem Lidar- und einem Radarsensor im Frontbereich ausgerüstet (siehe Abbildung 5).

Der Lidarsensor führt bereits eine Objekthypothesenbildung durch und beschreibt Messdaten über eine senkrecht zum Sensor ausgerichtete Linie. Es können maximal 10 Objekthypothesen gleichzeitig gebildet werden. Der Sichtbereich des Sensors ist mit 15° Öffnungswinkel und einer Reichweite von 200m angegeben.

Der Radarsensor führt ebenfalls eine Objekthypothesenbildung durch und beschreibt diese durch einen x-y-Punkt sowie einen Geschwindigkeitsvektor. Es können maximal 20 Objekthypothesen gleichzeitig gebildet werden. Der Öffnungswinkel des Sensors beträgt 60° bei einer Reichweite von 150m.

Das erste Projekt beschreibt eine adaptive Geschwindigkeitsregelung, welche als Eingangsdatum punktförmige Objekthypothesen mit lateraler Geschwindigkeit benötigt, um den Abstand zum nächsten vorderen Fahrzeug optimal einregeln zu können.

Das zweite Projekt stellt ein Sicherheitssystem für niedrige Geschwindigkeiten dar. Dieses verhindert das Anfahren, wenn der Bereich unmittelbar vor dem Fahrzeug durch ein Hindernis versperrt ist. Hierzu benötigt das System die Ausdehnung der Hindernisse, welche durch Objekthypothesen in Linienform beschrieben werden, sowie die Standardabweichung, um eine Güte der gefilterten Daten mit in die Entscheidung zur Freigabe der

Längsaktorik einzubeziehen.

Ziel in beiden Projekten ist der Wunsch nach einer möglichst geringen Rechenlast durch die Sensorverarbeitung.

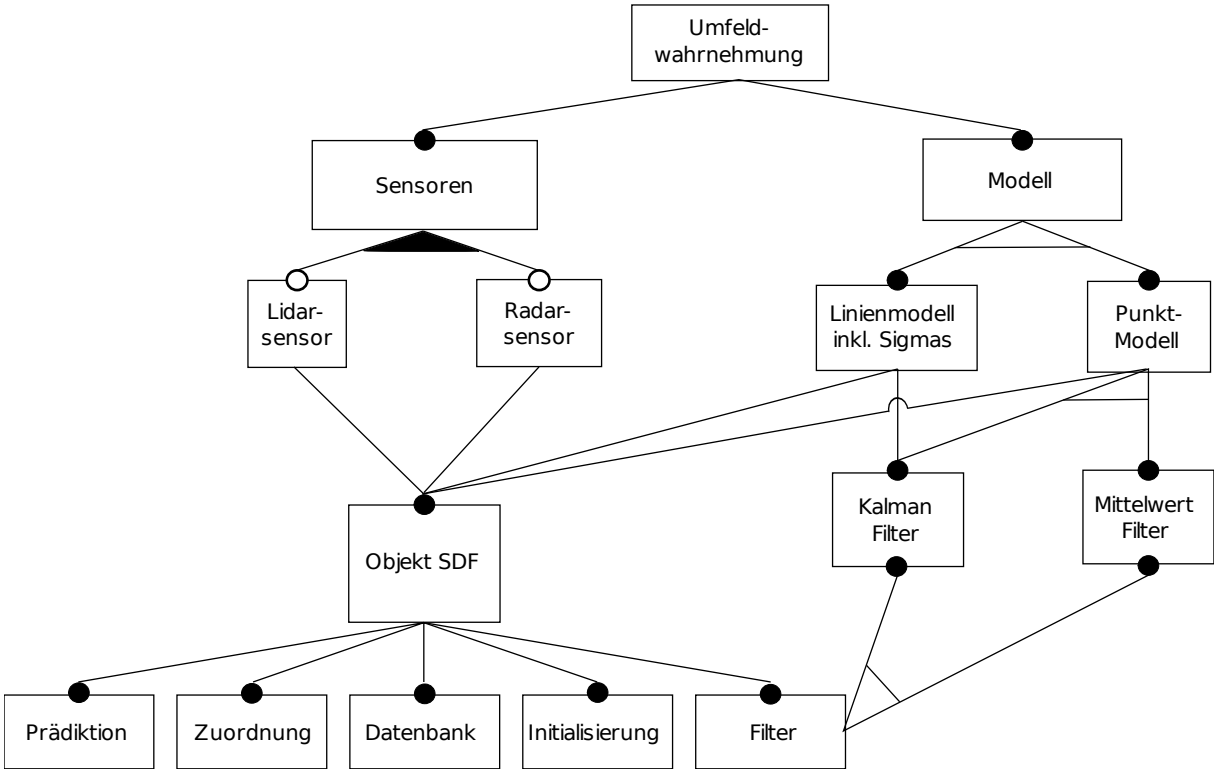

Abbildung 6: Feature-Modell

In Abbildung 6 ist ein vereinfachtes Feature-Modell des Systems aufgetragen. Zunächst wird dieses für das erste Projekt ausgewertet (siehe Abbildung 7), um zu einer Konfiguration zu gelangen. Zunächst werden die Kanten am Knoten Umfeldwahrnehmung (Sensoren und Modell) selektiert. Ausgehend davon lassen sich nun das Modell der SDF basierend auf der Anforderung des Projekts sowie die verwendeten Sensoren auswählen. Das gewählte Punktmodell sowie die durch die gewählten Sensoren vorgegebenen Eingangsdaten lassen hier die Möglichkeit, unterschiedliche Filter zu nutzen. Das Feature-Modell ist nicht eindeutig, da Vorselektionen durchgeführt wurden. Es besteht sowohl die Möglichkeit einen Kalman Filter als auch eine einfache gleitende Mittelwertbildung zu nutzen.

Zur Entscheidung über diese Unklarheit kann die in Abschnitt 4 vorgestellte Formel verwendet werden. Der Wunsch nach einer möglichst geringen Rechenlast ist hier ausschlaggebend. Der hier eingesetzte Kalman Filter besitzt durch die Matrizenoperationen eine Komplexität von $O(nlog^2n)$, während die Filterung via Mittelwertbildung nur eine

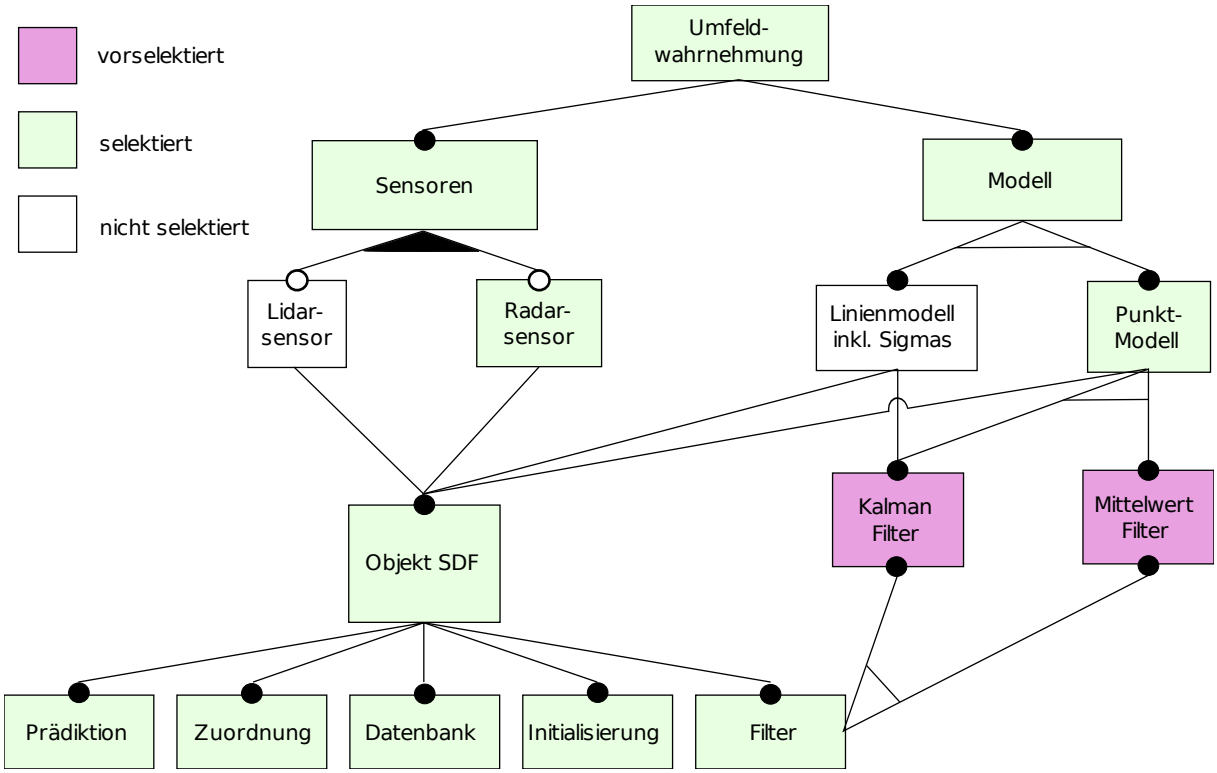

Abbildung 7: Ausgewertetes Feature-Modell für Projekt 1

Komplexität $O(n)$ besitzt. Der Mittelwert Filter entspricht somit den Anforderungen des Projekts.

Die Auswertung des Diagramms für das zweite Projekt (siehe Abbildung 8) folgt dem gleichen Algorithmus wie das erste Projekt. Zunächst werden wieder die Kanten “Sensoren” und “Modell” selektiert. Davon ausgehend muss jedoch der Forderung nach der Ausdehnung der Hindernisse Rechnung getragen werden. Aus diesem Grund wird das Linienmodell sowie der Lidarsensor selektiert. Die Wahl des Filters wird bei diesem Projekt bereits durch die Anforderungen aufgelöst. Die Applikation fordert eine Information über die Güte der Filterung. Diese kann durch den Mittelwert Filter nicht geliefert werden. Der Kalman Filter liefert hier jedoch Standardabweichungen der Zustandsgrößen.

6 Ausblick

Der hier vorgestellte Ansatz bietet große Vorteile bezüglich des variablen Einsatzes von Umfeldwahrnehmungssystemen in unterschiedlichen Versuchsträgern und Projekten. Der Einsatz von Software Produktlinien ermöglicht die Nutzung von individuell an das Pro-

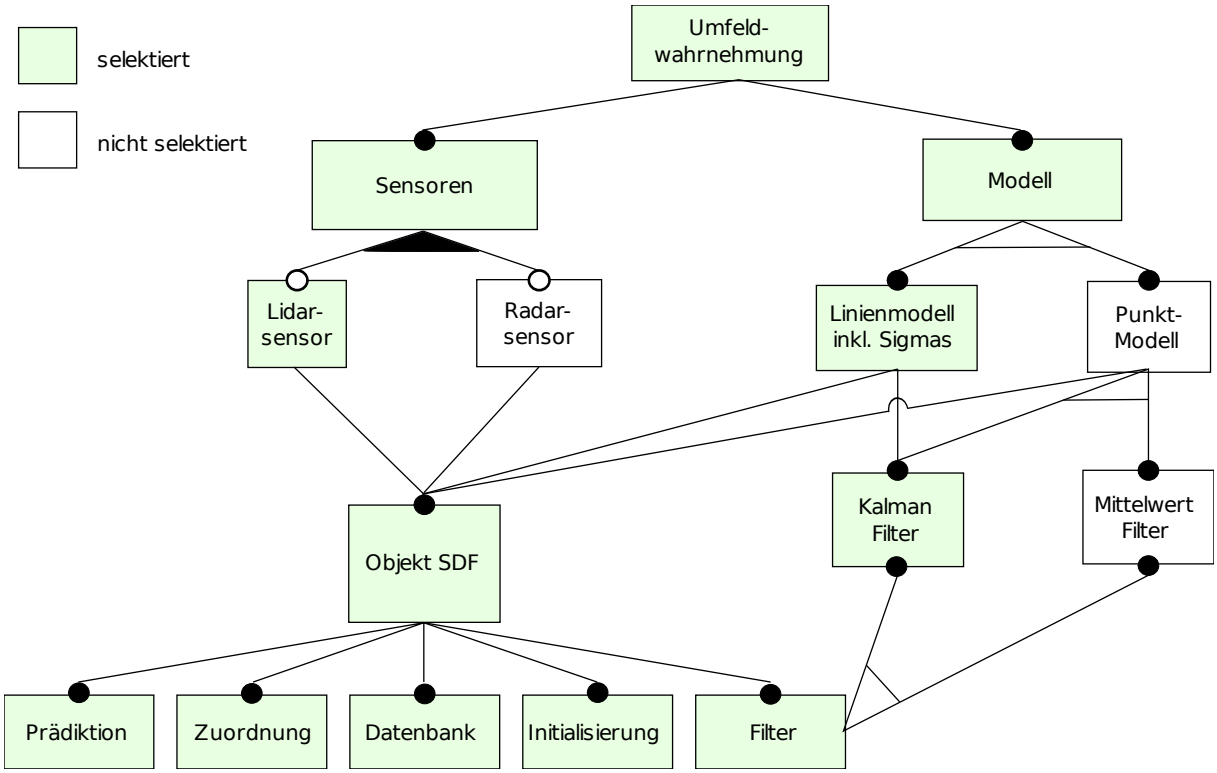

Abbildung 8: Ausgewertetes Feature-Modell für Projekt 2

jekt angepassten Umfeldwahrnehmungen bei gleichzeitiger flexibler Weiterentwicklung der Architekturkomponenten durch die Trennung von Produkt- und Komponentenentwicklungszyklus.

Die vorgestellte Architektur ermöglicht es, je nach eingesetzten Algorithmen, eine große Anzahl von unterschiedlichen Applikationen mit den benötigten Umfelddaten zu versorgen. Durch die explizite Modellierung der Variationspunkte sowie der Abhängigkeiten zwischen den Architekturkomponenten ist eine einfache Erstellung von projektspezifischen Umfeldwahrnehmungen möglich. Der Einsatz von Qualitätsmetriken für Umfeldwahrnehmungen zur Auflösung von uneindeutigen Abhängigkeiten bietet ferner die Möglichkeit, den Prozess der Produkterstellung zu automatisieren, um so eine Vielzahl von Produkten gleichzeitig zu warten und von Weiterentwicklungen an den Architekturkomponenten profitieren zu lassen.

Im Projekt Stadtpilot der TU Braunschweig (Wille u. a., 2009) wird die Umfeldwahrnehmung derzeitig als Software Produktlinie aufgebaut, um den Versuchsträger flexibel auch für andere Projekte einsetzen zu können. In der aktuellen Ausbaustufe werden die Architekturkomponenten noch manuell kombiniert. Eine weitere Entwicklung in die Rich-

tung der Automatisierung ist hier der nächste Schritt zu einer flexiblen Umsetzung in mehreren Projekten. Die funktionale Weiterentwicklung der Architekturkomponenten des Projekts sowie die Entwicklung und der Einsatz von weiteren Metriken zur Qualitätsbestimmung der kombinierten Komponenten stellen die nächsten Schritte im Bereich Umfeldwahrnehmung des Projekt Stadtpilot da.

Literatur

[Botterweck u. a. 2008] Botterweck, Goetz ; Thiel, Steffen ; Cawley, Ciarán ; Nestor, Daren ; Preussner, André: Visual Configuration in Automotive Software Product Lines. In: *Computer Software and Applications Conference, Annual International* 0 (2008), S. 1070–1075. – ISSN 0730-3157

[Buhr und Kolagari 2004] Buhr, K. ; Kolagari, R. T.: Softwarebasierte Produktlinien - Szenarien für Automobilhersteller und Zulieferer. In: *GI - Softwaretechnik-Trends* 24 (2004), November, Nr. 4. – URL `http://pi.informatik.uni-siegen.de/stt/24_4/03_Technische_Beitraege/buhr.pdf`. – ISSN 0720-892

[Buschmann u. a. 1996] Buschmann, Frank ; Meunier, Regine ; Rohnert, Hans ; Sommerlad, Peter ; Stal, Michael: *Pattern-oriented software architecture - A system of patterns*. Wiley, 1996

[Böckle u. a. 2004] Böckle, G. ; Knauber, P. ; Pohl, K. ; Schmid, K.: *Software-Produktlinien*. dpunkt.verlag, 2004

[Chastek 2002] Chastek, G. J. (Hrsg.): *Systematic Integration of Variablility into Product Line Architectural Design.* Bd. *2379*. Springer, 2002. (Lecture Notes in Computer Science 2nd Software Product Line Conference (SPLC-2), San Diego, CA, USA). – Seite 130-153

[Effertz 2009] Effertz, Jan: *Autonome Fahrzeugführung in urbaner Umgebung durch Kombination objekt- und kartenbasierter Umfeldmodelle*, Institut für Regelungstechnik Technische Universität Braunschweig, Dissertation, February 2009

[Gerhard Goos und van Leeuwen 2004] Gerhard Goos, Juris H. (Hrsg.) ; Leeuwen, Jan van (Hrsg.): *Staged Configuration Using Feature Models*. Bd. *3154/2004*. Springer,

2004. (Lecture Notes in Computer Science Third International Conference, SPLC 2004, Boston, MA, USA, August 30-September 2, 2004). – Seite 162-164

[IABG 2009] IABG, Industrieanlagen-Betriebsgesellschaft m.: *V-Modell XT Version 1.3.* February 2009

[Kruchten 1999] Kruchten, Philippe: *Der Rational Unified Process - Eine Einführung.* Addison-Wesley, 1999

[Maurer 2009] Maurer, Markus: Entwurf und Test von Fahrerassistenzsystemen. In: *Handbuch Fahrerassistenzsysteme*, 2009, S. 43–54. – ISBN 978-3-8348-0287-3

[Northrop und et al. 2007] Northrop, Linda M. ; al., Paul C. C. et: *A Framework for Software Product Line Practice, Version 5.0.* July 2007

[Schobbens u. a. 2006] Schobbens, P.-Y. ; Heymans, P. ; Trigaux, J.-C.: Feature Diagrams: A Survey and a Formal Semantics. In: *Requirements Engineering, 14th IEEE International Conference*, Sept. 2006, S. 139–148. – ISSN 1090-705X

[Thiel und Hein 2002] Thiel, S. ; Hein, A.: Modelling and using product line variability in automotive systems. In: *Software, IEEE* 19 (2002), Jul/Aug, Nr. 4, S. 66–72. – ISSN 0740-7459

[Wille u. a. 2009] Wille, J. ; Matthaei, R. ; Ohl, S. ; Saust, F. ; Maurer, M. ; Schuhmacher, W. ; Homeier, K ; Nothdurft, T. ; Sasse, A. ; Hecker, P. ; Wolf, L.: Der Stadtpilot - Autonomes Fahren auf dem Braunschweiger Stadtring. In: *AAET 2009, Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel* Gesamtzentrum für Verkehr Braunschweig e.V. (Veranst.), 2009, S. 27–47. – ISBN 978-3-937655-19-2