Modellierung adaptiver eingebetteter Systeme

Mario Trapp
Fachbereich Informatik
TU Kaiserslautern
trapp@informatik.uni-kl.de

Bernd Schürmann
Fachbereich Informatik
TU Kaiserslautern
schuerma@informatik.uni-kl.de

Philipp Wörz
CS-SM/EVD1
Robert Bosch GmbH
philipp.woerz@de.bosch.com

Abstract: Ein elementares Problem bei der Entwicklung adaptiver Systeme besteht darin, dass das Adaptionsverhalten eines Systems durch sehr komplexe Zusammenhänge bestimmt ist. Eine zur Beherrschung dieser Zusammenhänge notwendige formale Modellierung des Adaptionsverhaltens ist allerdings nicht verfügbar. Deshalb wurden in einer Kooperation zwischen der TU Kaiserslautern und der Robert Bosch GmbH Techniken zur Modellierung adaptiver eingebetteter Systeme entwickelt. Als Fallbeispiel diente dazu das elektronische Stabilitätsprogramm ESP. Die entwickelten Modellierungstechniken haben sich in Fallstudien bewiesen und werden in Zukunft in der ESP-Entwicklung, an der mehrere Hundert Softwareentwickler beteiligt sind, eingesetzt werden.
Die Entwicklung adaptiver eingebetteter Systeme gewinnt zurzeit zunehmend an Bedeutung. In diesem Beitrag möchten wir daher die bereits vorhandenen grundlegenden Erkenntnisse unserer Untersuchungen darlegen. Dazu werden wir insbesondere allgemeine Modellierungsprinzipien vorstellen, die sich im Rahmen des Projektes als sinnvoll erwiesen haben.

1 Einleitung

Die Entwicklung adaptiver eingebetteter Systeme gewinnt zurzeit vor allem aufgrund neuer Anwendungsgebiete wie z.B. *ambient intelligence* oder *ubiquitous computing* stark an Bedeutung. Allerdings handelt es sich dabei nicht um einen völlig neuen Ansatz. Um den hohen Qualitätsansprüchen an eingebettete Systeme zu genügen, wird z.B. die dynamische Adaption bereits seit vielen Jahren eingesetzt, um *graceful degradation* zu erreichen.

Dazu passen adaptive eingebettete Systeme ihre Funktionalität bei einem Komponentenausfall so an, dass mit den verbleibenden Komponenten solange wie möglich zumindest eine eingeschränkte Funktionalität aufrechterhalten werden kann. Betrachtet man z.B. die Fahrdynamikregelung, so ist die Gierrate eines Fahrzeugs eine essentielle Größe. Fällt der Gierratensensor aus, so kann die Gierrate alternativ anhand von Radumdrehungen geschätzt werden, allerdings ist dann die Reglung nur noch in einer verminderten Qualität möglich.
Eine Adaption des Systems ist allerdings nicht nur aufgrund der Verfügbarkeit, sondern allgemein aufgrund der Qualität von Komponenten notwendig. So wird z.B. die Messung der Querbeschleunigung eines Fahrzeugs in vielen Fahrsituationen durch die Erdbeschleunigung verfälscht. In diesen Fällen muss sich das System adaptieren, um auch in diesen Fahrsituationen den gegebenen Qualitätsansprüchen genügen zu können.

Aktuelle Versionen des ESP sind bereits in der Lage, sich dynamisch an ihre Umgebung anzupassen. Allerdings ist die dynamische Adaption mit einer enormen Komplexität verbunden. Zum Beispiel hat eine aus den Raddrehzahlen berechnete Gierrate ganz andere Qualitätseigenschaften als eine mit einem Gierratensensor gemessene Gierrate. Bei einem Fehler des Gierratensensors ist es also nicht ausreichend, einfach nur die Komponente zur Bestimmung der Gierrate zu adaptieren. Vielmehr löst die Änderungen der Qualität einer Größe eine Kettenreaktion aus, die eine Adaption vieler Systemkomponenten erforderlich macht. Die diesen Kettenreaktionen zugrunde liegenden Abhängigkeiten zwischen Systemkomponenten lassen sich bei komplexen eingebetteten Systemen kaum manuell beherrschen. Ein Hauptproblem bei der Entwicklung adaptiver eingebetteter Systeme besteht zurzeit deshalb darin, dass eine zur Beherrschung der Komplexität benötigte explizite, modellbasierte Spezifikation der Adaption nicht verfügbar ist.

Deshalb wurde im Rahmen des Forschungsprojektes CHAMELEON[1] die Modellierung adaptiver eingebetteter Systeme untersucht. Im Rahmen des Projektes wurden allgemeine Konzepte der Modellierung von dynamischer Adaption entwickelt, die sich in bestehende Modellierungssprachen integrieren lassen. Zurzeit werden diese Ansätze in den Entwicklungsprozess der ESP-Entwicklung der Robert Bosch GmbH eingearbeitet und werden künftig in der Serienentwicklung eingesetzt.
In diesem Beitrag möchten wir die grundlegenden Erkenntnisse des Projektes darlegen. Dazu werden wir grundlegende Modellierungsprinzipien, aber auch notwendige Erweiterungen von bestehenden Modellierungsansätzen aufzeigen, die sich im Rahmen des Projektes als sinnvoll erwiesen haben. Insbesondere werden wir auf die folgenden, elementaren Aspekte der Modellierung adaptiver Systeme eingehen, die sich im Rahmen unserer Untersuchungen als äußerst wichtig herausgestellt haben:

- *Anforderungsmodellierung* (Abschnitt 4.1)
  Ähnlich wie bei Produktlinienansätzen sind die Anforderungen an adaptive Systeme variabel, da aufgrund der Adaption zur Laufzeit nicht immer alle Anforderungen gleichzeitig erfüllt werden können. Deshalb ist es notwendig, gültige Adaptionsvarianten zu definieren.

- *Modellierung der Qualität* (Abschnitt 4.2)
  Da die dynamische Adaption oft auf der Verfügbarkeit und Qualität von Systemkomponenten beruht, ist eine adäquate Modellierung der Qualität von entscheidender Bedeutung.

- *Verteilte Modellierung, zentrale Verwaltung der Adaption* (Abschnitt 4.3)
  Um das Adaptionsverhalten einer Komponente festlegen zu können, wird ein profundes Verständnis der Funktionalität dieser Komponente benötigt. Die Modellierung der Adaption muss modular, dezentral durch die Komponentenentwickler erfolgen können. Es muss aber auch das Adaptionsverhalten des Gesamtsystems betrachtet werden, um ein global optimiertes Adaptionsverhalten zu erreichen (Vermeidung der Explosion von Adaptionsvarianten oder Verklemmungen, Kostenmi-

[1] Das Forschungsprojekt Chameleon wurde im Rahmen des Sonderforschungsbereichs 501 (Entwicklung großer Systeme mit generischen Methoden) begonnen und wird seit 2002 in einer Kooperation zwischen der TU Kaiserslautern und der Robert Bosch GmbH fortgeführt.

nimierung etc.). Deshalb muss das Adaptionsverhalten zentral verwaltet werden. Dazu muss das Adaptionsverhalten des Gesamtsystems anhand der verteilten Spezifikation des Adaptionsverhaltens der Komponenten automatisch ableitbar und analysierbar sein.

2 Einordnung der Arbeiten

2.1 Klassifikation dynamischer Adaption

Die Entwicklung und vor allem die Modellierung adaptiver Systeme ist noch ein sehr junges, aber auch sehr breites Forschungsgebiet. Entsprechend wird die dynamische Adaption häufig als Randgebiet unterschiedlichster Forschungsgebiete mit entsprechend unterschiedlichen Zielsetzungen untersucht. Um unsere Arbeiten einordnen zu können, werden wir deshalb zunächst die Konzepte der dynamischen Adaption klassifizieren (siehe Abb. 1).

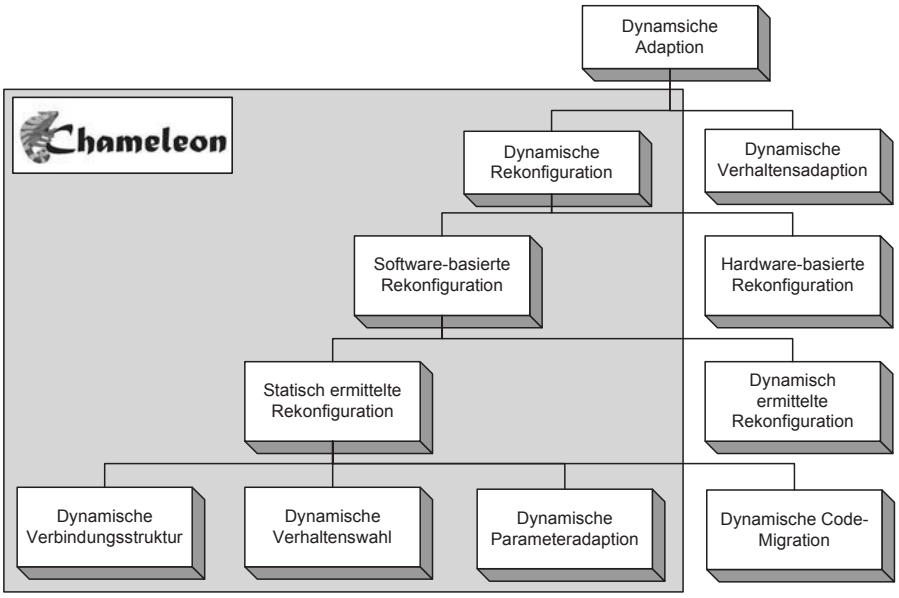

Abbildung 1: Klassifikation dynamischer Adaption

Prinzipiell kann man zwischen zwei Klassen von dynamischer Adaption unterscheiden: *dynamische Rekonfiguration* und *dynamische Verhaltensadaption*. Bei der dynamischen Verhaltensadaption wird die Funktionalität des Systems selbst, d.h. die Algorithmen, zur Laufzeit angepasst. Zu dieser Klasse gehören z.B. künstliche neuronale Netze oder evolutionäre Algorithmen. Systeme mit dynamischer Verhaltensadaption bieten die bestmögliche Flexibilität und können sich auch unvorhergesehenen Situationen anpassen.

Sie haben aber den Nachteil, dass das Adaptionsverhalten nicht vorhersagbar und somit nicht testbar ist und kommen deshalb in der von uns betrachteten Domäne nicht in Frage.

Bei der dynamischen Rekonfiguration wird die Adaption dadurch erreicht, dass das System zur Laufzeit in vordefinierte Konfigurationen versetzt wird. So kann dynamisch die für die gegebene Situation beste Konfiguration gewählt werden. Da alle möglichen Konfigurationen zur Entwicklungszeit definiert sind, ist das Adaptionsverhalten des Systems deterministisch festgelegt und somit testbar.

Auch wenn der Begriff Rekonfiguration oft mit der *hardware-basierten Rekonfiguration* (insbesondere FPLDs[2]) gleichgesetzt wird, liegt der Fokus unserer Forschung auf der *software-basierten Rekonfiguration*. Aber auch hier lassen sich zwei mögliche Ansätze unterscheiden. Ein Ansatz besteht darin, die Komplexität der Adaption zu beherrschen, indem diese für den Entwickler transparent bleibt. Dies wird dadurch erreicht, dass das Rekonfigurationsframework zur Laufzeit selbstständig entscheidet, wie das System zu konfigurieren ist (*dynamisch ermittelte Rekonfiguration*). Dadurch kann der Entwicklungsaufwand entscheidend verringert werden. Allerdings birgt dieser Ansatz auch zwei entscheidende Nachteile. Erstens sind die benötigten Rekonfigurationsframeworks sehr komplex und entsprechend groß und langsam, was sie für viele Einsatzgebiete ausscheiden lässt. Noch wichtiger ist aber, dass das System auch hier nicht oder nur schwer getestet werden kann. Da aber die vollständige Testbarkeit des Systems in den untersuchten Domänen gefordert wird, haben wir die *statisch ermittelte Rekonfiguration* untersucht. Dabei wird das Adaptionsverhalten des Systems zur Entwicklungszeit vollständig spezifiziert und ist dementsprechend vollständig analysier- und testbar. Zwar wird dadurch die Flexibilität eingeschränkt, aber im Rahmen des Projektes hat sich herausgestellt, dass selbst die Flexibilität der statischen Definition so groß ist, dass diese bei weitem nicht ausgenutzt wird.

Zur Umsetzung der dynamischen Rekonfiguration haben sich vier Verfahren etabliert. Bei einer *dynamischen Verbindungsstruktur* werden die Verbindungen zwischen Systemkomponenten zur Laufzeit geändert, wodurch eine Rekonfiguration auf Systemebene ermöglicht wird. Auf Komponentenebene kommt die *dynamische Verhaltenswahl* zum Einsatz. Für eine Komponente werden verschiedene Verhaltensvarianten (z.B. Ermittlung der Gierrate mittels eines Messwerts oder mittels einer Schätzung basierend auf den Raddrehzahlen) definiert. Zur Laufzeit wird dann die am besten passende Variante dynamisch ausgewählt. Um innerhalb einer Verhaltensvariante eine feinere Anpassung zu erreichen, wird die *dynamische Parameteradaption* eingesetzt. Dabei werden die Parameter einer Funktionalität dynamisch an die Umgebungsbedingungen angepasst (z.B. Anpassung von Filterkoeffizienten abhängig vom Verfahren der Gierratenbestimmung).
Eine weitere Möglichkeit der Adaption besteht darin, die Zuordnung von Funktionalitäten zu Hardwareknoten zur Laufzeit anzupassen. Fällt z.B. der Mikrokontroller, auf dem das Antiblockiersystem (ABS) ausgeführt wird, aus, so soll das ABS-Programm möglicherweise auf den Mikroprozessor des Radios übertragen und dort ausgeführt werden [1]. Die Anwendung dieses Ansatzes ist allerdings nur sehr langfristig gesehen möglich, da die dafür notwendige Entwicklung einer offenen und einheitlichen Architektur für

[2] Field Programmable Logic Devices

eingebettete Systeme im Automobil gerade erst begonnen hat [2]. Deshalb wurde die *dynamische Code-Migration* im Forschungsprojekt CHAMELEON nicht untersucht.

2.2 Stand der Technik

Die Modellierung dynamischer Rekonfiguration von Softwaresystemen ist ein noch relativ junges Forschungsgebiet. Entsprechend wenige Forschungsgruppen beschäftigen sich im Kern mit diesem Thema. Die Projekte, die sich mit dynamischer Rekonfiguration beschäftigen, sind auf die Gestaltung eines Rekonfigurationsframeworks fokussiert.

Im Rahmen des an der Carnegie Mellon University durchgeführten Projekts RoSES [1] wurde ein Rekonfigurationsframework entwickelt, das eine dynamisch ermittelte Rekonfiguration ermöglicht. Das Ziel des Projektes besteht ebenfalls darin, graceful degradation zu erreichen. Zur Adaption wird hauptsächlich die dynamische Code-Migration angewandt.

Im Rahmen des Projektes DepAuDE [3] wurde die getrennte Beschreibung der eigentlichen Funktionalität und der Aktionen, die bei einem Fehler ausgeführt werden, untersucht. Es wurde angestrebt, dass man bekannte Fehlertoleranz-Mechanismen wie z.B. Watchdog- oder Voter-Funktionalitäten zunächst projektunabhängig und somit wiederverwendbar beschreiben konnte. Mit der Skriptsprache ARIEL war es dann möglich zu definieren, welche dieser vordefinierten Mechanismen beim Auftreten von Fehlern ausgeführt werden sollen.

In [4] werden so genannte Containment Units vorgestellt. Mit deren Hilfe ist es möglich, die Qualität einer Funktionalität aufgrund verschiedener Kriterien zu überwachen und ggf. abzuschalten und, wenn möglich, durch eine alternative Funktionalität zu ersetzen. Auch hier liegt der Fokus in der Entwicklung eines geeigneten Frameworks.

In [5] wird ein auf Agenten basierendes Rekonfigurationsframework vorgestellt, das dynamische Parameteradaption, Verhaltenswahl und Code-Migration ermöglicht.

3 Gesamtprozess

Bevor wir auf die eingangs erwähnten Aspekte der Modellierung dynamischer Adaption eingehen, werden wir anhand des in CHAMELEON verwendeten Referenzvorgehensmodells (siehe Abb. 2) zunächst einen kurzen Überblick über den Gesamtprozess der Entwicklung adaptiver Systeme geben, um die Modellierungstechniken einordnen zu können.

Viele Anwendungsgebiete für eingebettete Systeme sind durch eine sehr hohe Variantenvielfalt geprägt. Deshalb wurde im Rahmen des Forschungsprojektes von der Entwicklung von Produktlinien [6] ausgegangen.

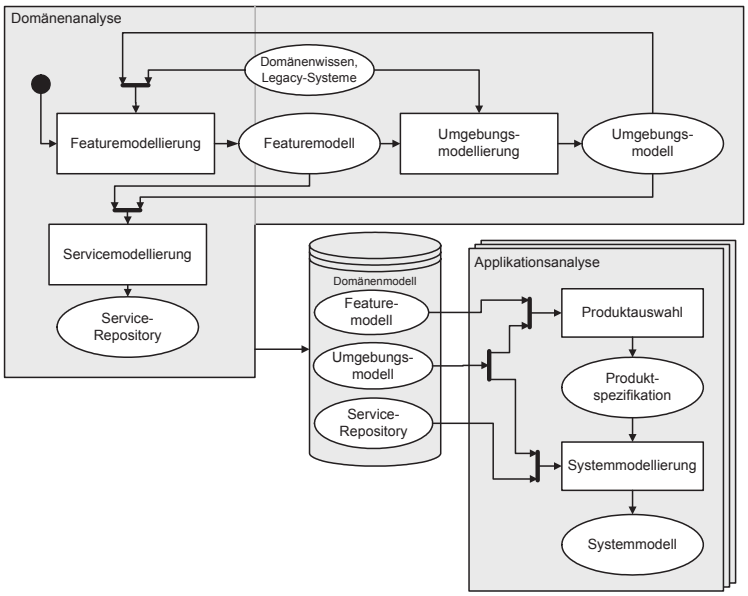

Abbildung 2: Referenzvorgehensmodell

Im Rahmen der Domänenanalyse werden die Eigenschaften der betrachteten Domäne analysiert. Die Featuremodellierung erfolgt zunächst nach einem dafür geeigneten Vorgehensmodell, wie z.B. FODA [7]. Durch Featuremodelle werden gültige Produkte, d.h. statische Systemkonfigurationen definiert. Für die Entwicklung adaptiver Systeme ist es in diesem Schritt zusätzlich notwendig, gültige dynamische Systemkonfigurationen, d.h. die Anforderungen an das Adaptionsverhalten zu definieren. Wir haben dazu die verwendeten Featuremodelle so erweitert, dass es zusätzlich möglich ist, gültige dynamische Konfigurationen zu definieren.

Nachdem in der Featuremodellierung zunächst die initialen Anforderungen an die dynamische Adaption festgelegt wurden, müssen im nächsten Schritt technisch mögliche bzw. sinnvolle Adaptionsvarianten identifiziert werden. Allerdings bieten Featuremodelle dazu keinerlei Unterstützung.
Zur Identifikation von technisch möglichen Adaptionsvarianten eines eingebetteten Systems ist ein profundes Verständnis der physikalischen Eigenschaften der betrachteten Domäne erforderlich. Um das dazu benötigte Domänenwissen formal modellieren und anschließend analysieren zu können, wurde die Domänenanalyse um die Umgebungsmodellierung erweitert [8]. Da sich die Feature- und Umgebungsmodellierung gegenseitig stark beeinflussen und jeweils neue Impulse liefern, entsteht hier ein iterativer Prozess, der solange wiederholt wird, bis das geforderte Verhalten mit den technischen Möglichkeiten in Einklang gebracht wurde.
Das Umgebungsmodell ist ein formales Modell, das alle Möglichkeiten zur Realisierung der geforderten Features beinhaltet. Dadurch ist es z.B. möglich, bereits während der Domänenanalyse zu analysieren, bis zu welchen qualitativen Bedingungen ein Feature aufrechterhalten werden kann, welche Sensuatorik benötigt oder mit welcher Wahr-

scheinlichkeit ein Feature „ausfallen“ wird. Umgekehrt lässt sich analysieren, wie sich Änderungen in der Verfügbarkeit oder Qualität von z.B. Sensuatoren auf die Realisierbarkeit der Features auswirken. Dadurch lassen sich sehr früh Fehler und Probleme identifizieren und sehr viele Adaptionsvarianten können ausgeschlossen werden, was den Entwicklungsaufwand entscheidend verringern kann.

Sobald das Feature- und Umgebungsmodell in Einklang gebracht wurden, lässt sich ein Portfolio wieder verwendbarer Services erzeugen bzw. erweitern [9]. In CHAMELEON stehen Services als abstraktes Konzept für die funktionalen Einheiten der verwendeten Modellierungssprache. Services können z.B. als Komponenten oder Klassen in der UML, als Blöcke in SDL oder als Subsysteme in Matlab Simulink realisiert werden. Services laufen konkurrent zueinander und kommunizieren über Nachrichten. Dieses Modellierungsprinzip hat sich in den betrachteten, stark datenflussorientierten Domänen als sinnvoll erwiesen.

Wenn die Domänenanalyse abgeschlossen und das Featuremodell instanziiert ist, d.h. eine Produktspezifikation definiert wurde, kann das Systemmodell erstellt werden. Dazu werden zunächst die im Service-Repository definierten Services instanziiert und miteinander verbunden. Services, die nicht im Repository vorliegen, werden neu entwickelt. Bei den Verbindungen zwischen den Services ist allerdings zu bedenken, dass diese nicht statisch festgelegt werden können, sondern sich dynamisch anpassen müssen, d.h. die Systemmodellierung muss um dynamische Verbindungen erweitert werden.

4 Modellierungsprinzipien

4.1 Anforderungsmodellierung

Häufig ist bei der Entwicklung adaptiver Systeme nicht klar, welche Anforderungen an das Adaptionsverhalten des Systems gestellt werden. Deshalb war es zunächst notwendig die Anforderungsmodellierung zu erweitern, indem wir Featuremodelle zur Modellierung dynamischer Systemkonfigurationen erweitert haben.

Dazu sind wir zunächst von einem sehr einfachen statischen Featuremodell ausgegangen, in dem jedes Feature entweder optional oder verpflichtend ist und durch weitere Subfeatures verfeinert werden kann. Außerdem können zwischen zwei Features *requires*- und *mutually-exclusive* Beziehungen definiert werden. Wir haben uns auf diese Elemente beschränkt, da nur diese für die Modellierung dynamischer Systemkonfigurationen erweitert werden mussten.

Um dynamische Systemkonfigurationen definieren zu können, war es zunächst notwendig zu unterscheiden, ob ein Feature statisch und/oder dynamisch optional ist. Zum Beispiel muss das Feature *Fahrzeugstabilisierung* als Kernelement des ESPs in jedem ESP-Produkt enthalten sein. Zur Laufzeit kann dieses Feature aber im Rahmen einer dynamischen Adaption abgeschaltet werden. Es ist also nicht statisch aber dynamisch optional.

Außerdem muss auch für die typischen Abhängigkeiten zwischen Features (requires bzw. mutually exclusive) unterschieden werden, ob diese Abhängigkeiten statisch

und/oder dynamisch gültig sind. So kann z.B. ein ESP vereinfacht gesehen unter Verwendung der Bremsen und/oder des Motors eingreifen. Das Feature *Fahrzeugstabilisierung* lässt sich also in die drei Subfeatures *BremsMotoreingriff*, *Bremseingriff* und *Motoreingriff* aufteilen. Ein ESP kann eines oder mehrere dieser Features realisieren. Zur Laufzeit kann aber immer nur eines dieser Features realisiert werden. Während diese Features also statisch nicht in Beziehung stehen, muss dynamisch eine mutually-exclusive Beziehung definiert werden.

Am Beispiel der *Fahrzeugstabilisierung* lässt sich erkennen, dass Featuremodelle genutzt werden können, um Qualitätsabstufungen zu definieren, d.h. die drei Subfeatures der *Fahrzeugstabilisierung* stellen mögliche Qualitäten der Realisierung dar. Ein eigenes Modellierungselement für die Darstellung von Qualitätsabstufungen hat sich nicht als sinnvoll erwiesen. Dies liegt insbesondere daran, dass Qualitätsabstufungen gegenüber dem Kunden wie gewöhnliche Features behandelt, d.h. als Eigenschaft des Produktes verkauft werden. Im Beispiel kann also der Kunde auswählen, ob das von ihm gewünschte ESP neben der vollwertigen *Fahrzeugstabilisierung* eine oder mehrere Rückfallebenen haben soll. Natürlich ist es genauso gut denkbar, z.B. nur das Feature *Bremseingriff* als finanziell günstigere Lösung für einen Kleinwagen zu wählen.

4.2 Modellierung der Qualität

Basierend auf den Anforderungen an das Adaptionsverhalten muss dieses im Rahmen der Systementwicklung modelliert werden. Da die dynamische Adaption auf der Verfügbarkeit und Qualität von Komponenten basiert, ist ein geeignetes Modell der Qualität eine Grundvoraussetzung für die weiteren Modellierungsschritte. Bevor wir auf die Modellierung des eigentlichen Adaptionsverhaltens eingehen, soll deshalb in diesem Abschnitt zunächst die Modellierung der Qualität erläutert werden.

Ein üblicher Ansatz besteht darin, die Qualitäten an die Services zu koppeln. Dies ist allerdings nicht sinnvoll, da dann zur Definition des Adaptionsverhaltens eines Service die anderen Services des Systems bekannt sein müssen, d.h. eine modulare Definition der Adaption wäre nicht möglich. Deshalb werden in CHAMELEON die Qualitätsbeschreibungen nicht an die Funktionen sondern an die Daten gekoppelt.

Dazu werden die Daten in CHAMELEON durch *Variablen* repräsentiert, deren Typ anstatt durch einfache Datentypen durch *Variablentypen* [8] definiert wird. Variablentypen beschreiben die (meist physikalischen) charakteristischen Größen der Umgebung, wie z.B. die *Fahrzeuggeschwindigkeit,* und werden im Rahmen der Umgebungsmodellierung definiert. Jedem Variablentyp lässt sich eine eindeutige Semantik zuordnen, die unabhängig davon ist, über welche Funktion der Wert bestimmt wurde. Dadurch sind Variablentypen lösungsunabhängige Fixpunkte der Domäne. Zusätzlich wird jeder Variablentyp mit einer Qualitätsbeschreibung versehen. Dadurch wird erreicht, dass für alle in einem System verwendeten Daten sowohl die Semantik, als auch die Qualität definiert ist. Dies hat zwei entscheidende Vorteile:

- Da auch die Service-Schnittstellen über Variablentypen definiert werden, haben Services automatisch semantisch definierte Schnittstellen. Da die Variablentypen zusätzlich mit Qualitäten versehen sind, können Services sehr leicht und flexibel innerhalb der

Domäne wieder verwendet werden.
Benötigt z.B. ein Service eine schlupffrei bestimmte Fahrzeuggeschwindigkeit, so ist es egal, wo und wie die Fahrzeuggeschwindigkeit bestimmt wurde, solange diese die geforderten Qualitätseigenschaften erfüllt.

- Durch die Zuordnung von Qualitäten zu Variablentypen ist es möglich, das Adaptionsverhalten eines Service völlig modular zu definieren, da alle zur Adaption notwendigen Informationen an der lokalen Schnittstelle eines Service verfügbar sind.

Die wichtigste Aufgabe der Qualitätsbeschreibung besteht darin, dass die Entwickler anhand dieser entscheiden können, wie die von ihnen entwickelte Funktionalität beeinflusst wird. Dazu hat sich eine zweistufige Qualitätsbeschreibung als sehr effektiv erwiesen. Zunächst muss dokumentiert werden, mit welchem Verfahren ein Wert bestimmt wurde und welche Qualitätseigenschaften dieses Verfahren hat. Dazu werden den Variablentypen mehrere *Variablenmodi* zugeordnet (z.B. gemessene oder berechnete Gierrate). Zusätzlich können für jeden Modus verfahrensspezifische *Variablenqualitäten* definiert werden, die angeben, wie gut der Wert zur Laufzeit mit diesem Verfahren bestimmt bzw. gestellt wird (z.B. Einfluss des Radschlupfs auf die berechnete Gierrate).

4.3 Verteilte Modellierung des Adaptionsverhaltens

Die Entwicklung komplexer Systeme wie dem ESP ist auf viele Entwicklungsteams verteilt, die jeweils einen Service modular entwickeln. Nur diese Entwicklungsteams haben das Know-how um das Adaptionsverhalten dieses Service definieren zu können, weshalb auch das Adaptionsverhalten modular modelliert werden muss. Dies wird durch die in Abschnitt 4.2 beschriebene Kopplung der Qualität an die Variablen ermöglicht.

Wie in Abschnitt 2.1 beschrieben, wird in CHAMELEON die dynamische Rekonfiguration betrachtet. Auf Serviceebene wird dazu die dynamische Verhaltenswahl und Parameteradaption unterstützt [9]. Beide Adaptionsverfahren erfolgen dabei aufgrund der Verfügbarkeit und Qualität der benötigten Ein- und Ausgangsvariablen. Dazu muss zunächst definiert werden, welche Variablen von einem Service benötigt (*required*) und welche geliefert (*provided*) werden. Ein Service liefert eine Variable, wenn der Dienst des Service darin besteht, den Wert der Variablen zu bestimmen oder zu stellen. Ein Service benötigt eine Variable, wenn der Service zur Erbringung seines Dienstes darauf angewiesen ist, dass der entsprechende Wert zur Verfügung steht oder gestellt werden kann.
Die Definition dieser required/provided-Schnittstelle ist insbesondere wichtig, da man nicht davon ausgehen kann, dass die Eingangsvariablen immer benötigt und die Ausgangsvariablen immer geliefert werden. So hat z.B. im ESP der Stabilitätsregler u.a. ein Bremsmoment als Ausgang, trotzdem wird dieses benötigt, da die Funktion des Reglers darauf angewiesen ist, dass dieses Bremsmoment gestellt wird. Anderenfalls muss er seine Funktionalität adaptieren, um z.B. nur über Motormomente regeln zu können.

Zur Modellierung der dynamischen Verhaltenswahl werden für jeden Service mehrere *Servicekonfigurationen* definiert, die jeweils eine Verhaltensvariante repräsentieren. Entsprechend wird jeder Konfiguration ein eigenes Modell des funktionalen Verhaltens zugeordnet. Zur Modellierung des funktionalen Verhaltens wurden existierende Ansätze und Standardwerkzeuge verwendet.

Zur Modellierung des Adaptionsverhaltens eines Service wird zunächst jeder Konfiguration ein *guard* zugeordnet, der angibt, unter welchen Bedingungen die Konfiguration aktiviert werden darf, d.h. welche Variablen in welcher Qualität benötigt werden, um die entsprechende Funktionalität bieten zu können (vgl. Abbildung 3).
Neben dem guard erhält jede Konfiguration eine Priorität. Können aufgrund der guards mehrere Konfigurationen gleichzeitig aktiviert werden, so wird die Konfiguration mit der höchsten Priorität ausgewählt.
Zusätzlich muss über *influences* definiert werden, wie die von dem Service gelieferten Variablen beeinflusst werden, wenn dieser in die entsprechende Konfiguration versetzt wird. Dazu werden für jede gelieferte Variable der aktuelle Modus und die Werte der entsprechenden Qualitätsattribute (meist in Abhängigkeit von den Werten und/oder Qualitäten der benötigten Variablen) festgelegt.

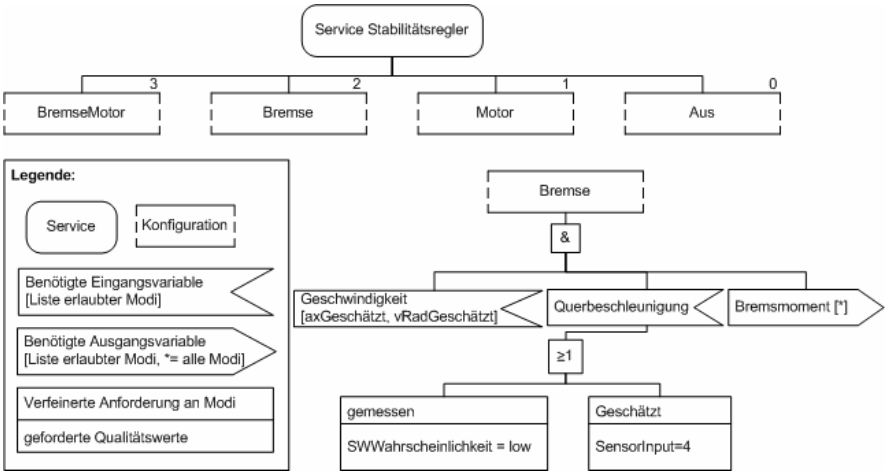

Abbildung 3: Konfigurationsbäume zeigen die Konfigurationen eines Service mit den zugehörigen *guards*.

Zur Modellierung der dynamischen Parameteradaption lassen sich für jede Konfiguration Parameter definieren. Für jeden dieser Parameter wird eine Abbildungsvorschrift definiert, die anhand der Qualität der benötigten Variablen den Parameterwert bestimmt.

4.4 Analyse des Adaptionsverhaltens

Aufgrund der in Abschnitt 4.3 beschriebenen modularen Definition des Adaptionsverhaltens der einzelnen Services, können diese bei der Analyse des Gesamtverhaltens des Systems als Blackbox betrachtet werden. Dies ist von entscheidender Bedeutung, da zur Bewertung des Adaptionsverhaltens des Gesamtsystems ein breites Wissen über die globalen Zusammenhänge erforderlich ist, weshalb das Adaptionsverhalten des Gesamtsystems zentral von einem eigenen Entwicklerteam verwaltet wird.

Im Rahmen der Systemmodellierung werden die einzelnen Serviceinstanzen verbunden. Dadurch ist neben dem Datenfluss auch der Qualitätsfluss innerhalb des Systems vollständig definiert: Die Variablen des Systems sind mit einer Qualität versehen und das

Adaptionsverhalten der Services definiert die Abbildung der Qualitäten (der Qualitätsfluss kann auch entgegen des Datenflusses verlaufen. Fällt z.B. ein Bremsaktuator aus, so wirkt sich dies entgegen des Datenflusses bis zum Stabilitätsregler aus, der sich dann entsprechend adaptieren muss.).
Für die Kommunikation mit dem Kunden ist es sinnvoll, den Bezug zum Featuremodell herzustellen. Dazu kann jedem Service und jeder Konfiguration ein realisiertes Feature zugeordnet werden.

Basierend auf dieser Definition des Qualitätsflusses wird die automatische Analyse des Adaptionsverhaltens des Gesamtsystems ermöglicht. Dabei wird analysiert, wie sich das System aufgrund einer Sequenz von qualitätsverändernden Ereignissen (Variable wird hinzugefügt, fällt aus oder ändert ihre Qualität) adaptiert. Dazu werden in einer zeitlichen Sequenz die Verfügbarkeit und die Qualität von Variablen geändert. Die Services, die diese Variablen benötigen, werden sich entsprechend adaptieren und die Qualität der von ihnen gelieferten Variablen anpassen, d.h. die eingangs erwähnten komplexen Kettenreaktionen können voll automatisch analysiert werden. Darauf basierend lässt sich zentral ein global sinnvolles Adaptationsverhalten des Gesamtsystems definieren.
Der Ablauf der Adaption lässt sich z.B. durch Adaptation Sequence Charts (ASC) darstellen (vgl. Abbildung 4). Anhand eines ASC lässt sich erkennen, welche Konfiguration die Services einnehmen, wie die Parameter adaptiert werden und wie sich dies auf die von ihnen gelieferten Variablen auswirkt. Aufgrund der Kopplung mit dem Featuremodell lässt sich auch erkennen, ob bzw. wie die Features von den Adaptionen beeinflusst werden.

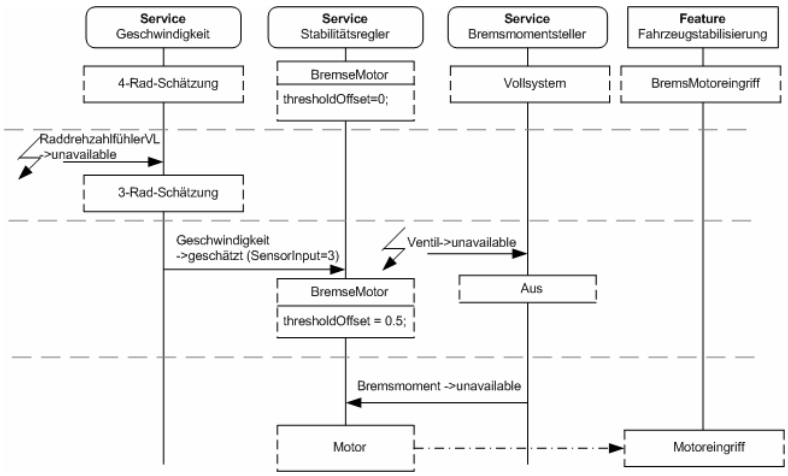

Abbildung 4: Beispiel eines Adaptation Sequence Charts (ASC)

5 Zusammenfassung

Der Entwicklung adaptiver eingebetteter Systeme kommt eine immer größere Bedeutung zu. Die modellbasierte Spezifikation der dynamischen Adaption ist dabei von enormer

Wichtigkeit. In diesem Beitrag haben wir deshalb entsprechende Modellierungstechniken vorgestellt, die auch den Anforderungen der Serienentwicklung gerecht werden.

Zunächst ist es wichtig, die Anforderungen an die dynamische Adaption zu definieren. Da die dynamische Adaption auch verstärkt zu einem durch den Kunden wählbaren Feature der Produktlinie wird, haben wir eine Erweiterung von Featuremodellen zur Modellierung der dynamischen Adaption vorgestellt. Für die Modellierung des Adaptionsverhaltens des Systems ist es von entscheidender Bedeutung, dass diese, wie auch die Modellierung der Funktionalität, modular durch verteilte Entwicklungsteams erfolgen kann. Basierend auf dieser modularen Modellierung muss das Adaptionsverhalten des Gesamtsystems automatisch analysierbar sein, um globale Aspekte zu untersuchen, die bei der modularen Modellierung keine Berücksichtigung finden können. Dazu wurde zunächst eine datenbasierte Qualitätsbeschreibung eingeführt, die die modulare Definition des Adaptionsverhaltens des Services ermöglicht. Durch das Adaptionsverhalten der einzelnen Services ist auch der Qualitätsfluss innerhalb des Systems formal definiert. Basierend auf diesem Qualitätsfluss lässt sich das Adaptionsverhalten des Gesamtsystems automatisch analysieren.

Mit den vorgestellten Techniken lassen sich also wieder verwendbare, modulare adaptive Komponenten entwickeln, aus denen ein adaptives System aufgebaut werden kann. Auch wenn die dynamische Adaption im Rahmen des vorgestellten Projektes primär eingesetzt wurde, um graceful degradation zu erreichen, so ist die Anwendung keineswegs darauf beschränkt. Eine direkte Anwendung findet sich z.B. in Produktlinien, da das System keinen Unterschied zwischen einer ausgefallenen bzw. nicht eingebauten Komponente erkennt. Aber z.B. auch eine Anwendung im eingangs erwähnten Bereich der Ambient Intelligence ist möglich.

6 Literaturverzeichnis

[1] W. Nace, P. Koopman, *A Graceful Degradation Framework for Distributed Embedded Systems*. Workshop on Reliability in Embedded Systems, October 28, 2001

[2] Offizielle Autosar-Website : http://www.autosar.org

[3] G. Dondossola, S. Donatelli, S. Bernardi, *Methodology for generation of modeling scenarios starting from the requisite specifications and its application to the collected requirements.* Ergebnisbericht Teilprojekt D1.3b des DepAuDE Projekts, 2002

[4] J.M. Cobleigh, B.S. Lerner, *Containment Units: A Hierarchically Composable Architecture for Adaptive Systems*. Proceedings of SIGSOFT 2002/FSE-10, SC, USA, 2002

[5] K.R. Dixon, T.Q. Pham, P.K. Khosla, *Port-Based Adaptable Agent Architecture*. Proceedings of IWSAS 2000, pp. 134-142, Springer Verlag Berlin, Heidelberg, 2002.

[6] G. Böckle, P. Knauber, K. Pohl, K. Schmid (Hrsg.), *Software-Produktlinien*. dpunkt.verlag, Mai 2004.

[7] K. C. Kang et al. *Feature-oriented domain analysis (FODA) feasibility study*. Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, Software Engineering Institute (SEI), 1990.

[8] M. Trapp, B. Schürmann, T. Tetteroo, *Variable-based Environment Model for Gracefully Degrading Embedded Systems*. 7th IASTED International Conference on Software Engineering and Applications – SEA 2003, Marina Del Rey, CA (USA), 2003

[9] M. Trapp, B. Schürmann, T. Tetteroo, *Service-Based Development of Dynamically Reconfiguring Embedded Systems*, IASTED International Conference on Software Engineering - SE 2003, Innsbruck, Austria, 2003