Praktikum Automotive Software Engineering: Best Practices

Michael Uelschen, Hochschule Osnabrück

m.uelschen@hs-osnabrueck.de

Zusammenfassung

Die Entwicklung von Software für Steuergeräte (ECU, electronic control unit) im Fahrzeug unterliegt speziellen Randbedingungen und Vorgehensweisen, die üblicherweise nicht Bestandteil in der einführenden Lehre des Software-Engineering sind. Dieser Beitrag untersucht die Frage, inwieweit die spezifischen Ausprägungen des Automotive Software Engineering (ASE) in der Ausbildung Studierender innerhalb eines Moduls vermittelt werden können.

Der vorgestellte Ansatz verzichtet bewusst auf den vollständigen Durchlauf aller Phasen im Entwicklungszyklus und setzt stattdessen einen Schwerpunkt auf spezifische Vorgehensweisen und Methoden bei der Software-Entwicklung im automobilen Bereich.

Einleitung

In den folgenden Abschnitten wird zuerst in die Thematik der Steuergeräte-Entwicklung eingeführt und die Bedeutung der Ausbildung im Bereich ASE hervorgehoben. Es werden die Unterschiede zum generellen Software Engineering herausgestellt.

Aus der arbeitsteiligen Vorgehensweise in der Automobilindustrie leiten sich Lernziele in Form von Kompetenzen für ein, im weiteren Verlauf des Beitrags vorgestelltes, Praktikum an der Hochschule Osnabrück ab. Die Erfahrungen und Beobachtungen während des Praktikums für Mechatronik-Studierende fließen in Empfehlungen zur Gestaltung einer entsprechenden Lehreinheit ein.

Automotive Software Engineering

Der Begriff Automotive Software Engineering (Schäuffele und Zurawka, 2012) umfasst Methoden, Werkzeuge und Vorgehensprozesse bei der Entwicklung von Software für Steuergeräte im Fahrzeug.

Hierbei handelt es sich im Allgemeinen um eingebettete Systeme, die Echtzeitanforderungen unterliegen. Durch die software-gesteuerte Verknüpfung von Sensorik und Aktuatorik lassen sich vielfältige Steuerungs- und Regelungsaufgaben realisieren. Hierbei gewinnen insbesondere Fahrerassistenzsysteme wie Parkassistent, Spurwarner oder adaptive Lichtregelungen zunehmend an Bedeutung (Winner et al., 2012).

Neben dem Entwurf neuer und der Verbesserung vorhandener Fahrfunktionen ist die Integration von Infotainment und Multimedia-Anwendungen eine wachsende Aufgabe des ASE. An der Schnittstelle zwischen der klassischen Automobilelektronik auf der einen und der Konsumgüterindustrie auf der anderen Seite entstehen zahlreiche neue Herausforderungen, die als *Innovation Cycle Dilemma* bezeichnet werden (Lucke et al., 2007).

Aufgrund der vielfältigen Fahrerinformations- und –assistenzsysteme gewinnt die Entwicklung einer geeigneten Fahrer-Fahrzeug-Schnittstelle an Bedeutung, um Informationen situationsabhängig unter Minimierung möglicher Ablenkungen bereitzustellen. Die Entwicklung multimodaler Mensch-Maschine-Schnittstellen für den Einsatz im Fahrzeug ist Gegenstand aktueller Forschung und Entwicklung (Pfleging et al., 2014).

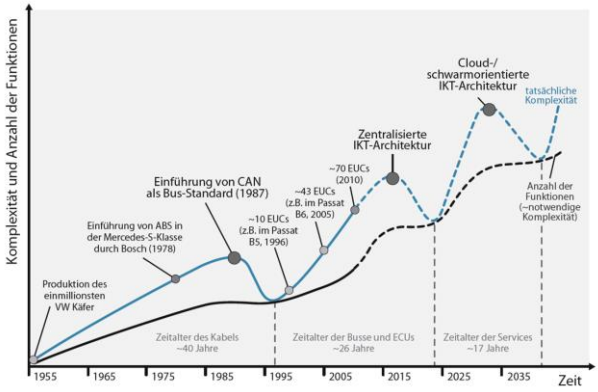

Abb. 1: Komplexität eines Fahrzeugs

Treiber und Herausforderungen

Broy zeichnet in (Broy, 2006) die evolutionären Schritte bei der Software-Entwicklung im Fahrzeug in den letzten 30 Jahren nach. Insbesondere durch die Einführung von fahrzeugspezifischen Netzwerken führen die bis zu 100 verteilten Steuergeräte innerhalb eines Automobils mehrere tausend

Funktionen aus. Moderne Fahrzeuge enthalten Software bis zu 100 Millionen Codezeilen (Mössinger, 2010).

Abbildung 1 zeigt die zunehmende Komplexität im Fahrzeug (Fortiss, 2011). Zusätzlich ist die tatsächliche Komplexität skizziert, die nicht aufgrund von Funktionalitäten, sondern durch Architektur- und Entwurfsentscheidungen entsteht. Ein Ende dieser Entwicklung ist zurzeit nicht absehbar. Hierbei sind die Herausforderungen im Bereich der Elektromobilität und der unterschiedlichen Stufen des autonomen Fahrens wesentliche technologische Treiber. Nach Schäuffele und Zurawka ist das „Automobil zum technisch komplexesten Konsumgut geworden".

In (Pretschner et al., 2007) sind die vielfältigen Herausforderungen des ASE zusammengefasst, die teilweise auch in anderen Domänen außerhalb der Automobilindustrie existieren.

Bedeutung

Aufgrund der zunehmenden Wertschöpfung im Fahrzeug durch Software (Weinmann, 2003), nimmt eine systematische Vorgehensweise bei der Entwicklung von Steuergeräten einen zunehmenden Stellenwert ein.

Die Automobilindustrie ist einer der wichtigsten Industriezweige in Deutschland. Die größten Automobilhersteller und –zulieferer haben hier ihren Standort (insbesondere aber nicht nur in Baden-Württemberg und in Bayern) und sind interessante Arbeitgeber für Ingenieure und Informatiker. Zusätzlich gibt es eine sehr große Anzahl von, i.allg. unbekannten, klein- und mittelständischen Unternehmen (*hidden champions*), die wichtige Komponenten entwickeln oder fertigen.

Eine Berücksichtigung in der Ausbildung Studierender reflektiert somit die industrielle Relevanz und entspricht somit dem Anspruch einer praxisnahen Berufsvorbereitung.

Neben der Entwicklung von Personen- und Lastkraftfahrzeugen entsteht inzwischen ein weiterer, eigenständiger Bereich innerhalb der ASE. Durch die fortschreitende Industrialisierung in der Landwirtschaft findet zunehmend die Entwicklung von Steuergeräten in der Landmaschinentechnik statt. Diese übernehmen nicht nur reine Fahrfunktionen, sondern automatisieren agrar-spezifische Prozesse.

Einige der weltgrößten Unternehmen der Landtechnik sind in der Region Osnabrück und Nordwestdeutschland ansässig, u.a. Krone, Amazone, Grimme und Claas.

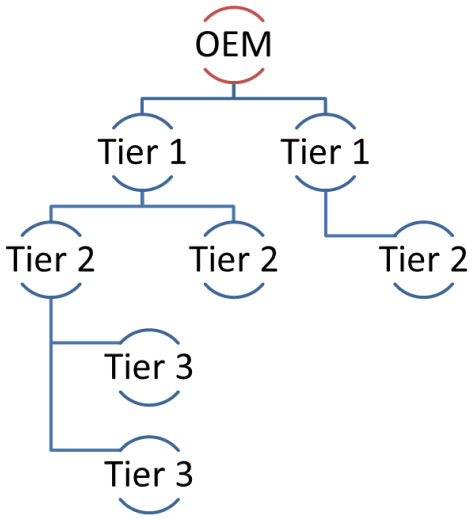

Abb. 2: Hierarchie der Automobilzulieferer

Arbeitsteilige Entwicklung

Die Entwicklung von Fahrzeugen, nicht nur der Steuergeräte, wird sehr stark durch Automobilzulieferer getrieben. Abbildung 2 zeigt hierbei eine dreistufige Hierarchie, bei der an der Spitze der Automobilhersteller (OEM, Original Equipment Manufacturer) als Auftraggeber steht. Die einzelnen Zulieferer ordnen sich in die Schichten Tier 1 bis Tier 3 ein, wobei Unternehmen, die der oberen beiden Schichten angehören, üblicherweise eine eigene Fertigung unterhalten. Entwicklungsdienstleister finden sich in der dritten Schicht wieder. Der OEM beauftragt einen Zulieferer, für eine bestimmte Funktionalität ein Steuergerät entsprechend den Vorgaben eines Lastenheftes zu entwickeln. Der zeitliche Entwicklungsrahmen wird durch den Termin des Neuanlaufs eines Fahrzeugs bestimmt. Aufgrund hoher Umsatzausfälle, die schnell die Millioneneurogrenze erreichen, ist eine Verschiebung von Neuanläufen ausgeschlossen, beispielsweise aufgrund fehlerhafter Software. Zur Absicherung der Terminplanung und der Risikominimierung setzt der OEM häufig auf einen zweiten Anbieter (*second source*).

Aufgrund dieses Vorgehens erfolgt die Entwicklung von Steuergeräten hochgradig parallel sowie räumlich getrennt bei unterschiedlichen Zulieferern. Da das neue Fahrzeug erst zu einem sehr späten Zeitpunkt im Entwicklungsprozess zur Verfügung steht, ist ein nicht unerheblicher Aufwand notwendig, um das umgebene System in frühen Phasen zu simulieren.

Vernetzung

Bei dieser sehr arbeitsteiligen Vorgehensweise in der Automobilindustrie ist es wichtig, dass die Schnittstellen der kommunizierenden Steuergeräte

zu einem sehr frühen Zeitpunkt im Entwicklungsprozess definiert sind. Die Kommunikation, der an die verschiedenen Netzwerke (LIN, CAN, FlexRay, Ethernet) angeschlossenen Komponenten, ist nachrichtenbasiert. Ein Nachrichtenkatalog beschreibt die nach außen sichtbare Funktionalität eines Netzknotens (Zimmermann und Schmidgall, 2014).

Die Nachrichten verändern den inneren Zustand eines Steuergeräts, so dass die Modellierung des Verhaltens neben der reinen Funktionsentwicklung einen wesentlichen Schwerpunkt bildet. Die Entwicklung gliedert sich hierbei in mehreren Stufen: zuerst wird das zu entwickelnde Steuergerät sowie die beteiligten Netzknoten simuliert. Danach wird das betreffende Steuergerät durch eine reale Komponente ersetzt (Restbussimulation). Im letzten Schritt wird das Steuergerät mit anderen realen Komponenten im Fahrzeug verbaut und anschließend erprobt.

Generelles Software Engineering

Im folgenden Abschnitt wird auf weitere Unterschiede eingegangen, die einen direkten Einfluss auf die Software-Entwicklung haben.

Grundsätzlich werden im ASE dieselben Aspekte und Phasen wie im generellen Software Engineering betrachtet (siehe beispielsweise in (Sommerville, 2012)). Viele Methoden lassen sich auf die ECU-Entwicklung übertragen.

Neben der Anwendungsdomäne, also den vielfältigen Fahrfunktionen, ergeben sich Unterschiede u.a. aufgrund weiterer Randbedingungen:

- Hoher Kosten- und Wettbewerbsdruck beeinflusst Entwurfsentscheidungen (beispielsweise geringe Prozessorleistung und Speicherkapazität).
- Unterschiedliche Klimazonen führen zu rauen Umgebungssituationen, auf die Hardware und Software reagieren müssen.
- Lange Produktzyklenzeiten stellen hohe Anforderungen an das Konfigurationsmanagement sowie an die Kompatibilität (siehe hierzu (Ritter, 2004)).

Es existiert eine Reihe von Branchenstandards, u.a. im Bereich der Entwicklungs- sowie Qualitätsverbesserungsprozesse, der Werkzeuge sowie der Architekturentwürfe:

- Die Software im Fahrzeug unterliegt zunehmend strengen Sicherheitsanforderungen, die konform zu ISO 26262 entwickelt werden muss (Gebhardt et al., 2013).
- Die Anstrengungen zur Qualitätsverbesserung orientieren sich an CMMI (Chrissis et al., 2011) oder Automotive SPICE (Höhn et al., 2009).
- Das AUTOSAR-Rahmenwerk definiert den grundsätzlichen Aufbau von automobilen Software-Anwendungen im Fahrzeug (Schäuffele und Zurawka, 2012).

Das Vorgehensmodell folgt in der Automobilindustrie dem V-Modell. Dieses wird um mehrere Iterationen oder Musterstände A bis D ergänzt, wobei das D-Muster den Serienstand und somit den Abschluss im Entwicklungsprozess darstellt. Agile Methoden werden bisher erst sehr wenig eingesetzt.

Die Software-Entwicklung im automobilen Bereich ist zunehmend modellbasiert und wird durch eine Reihe von proprietären Werkzeugen unterstützt.

Grundlagen

Für die ASE-Lehre bilden neben Programmierkenntnissen in der Sprache C/C++ die folgenden Themenbereiche die Grundlage:

- Regelungs- und Steuerungstechnik: Beschreibung kontinuierlicher Systeme
- Eingebettete Echtzeitsysteme
    - Ereignisbasierte und reaktive Systeme
    - Zustandsmaschinen beschreiben das dynamische Verhalten
    - Verwendung von Mikrocontrollern mit Peripherie
    - Weiche und harte Echtzeitanforderungen
    - Betriebssysteme: Berücksichtigung zeitlicher Anforderungen durch prioritätenbasiertes Scheduling
- Verteilte Systeme
    - Topologie/Gateway-Funktionalität
    - Echtzeitfähigkeit
    - Nachrichtenbasierte Kommunikation

Möglichkeiten in der Lehre

Einige Hochschulen in Deutschland bieten einen spezialisierten Master-Studiengang im Bereich Automotive Software Engineering an, u.a. die Universität Chemnitz und die Technische Universität München. Die Hochschule Landshut bietet einen Bachelor-Studiengang Automobilinformatik an. In vielen anderen fahrzeugspezifischen Studiengängen an Hochschulen in Deutschland ist die Belegung eines Moduls ASE verpflichtend oder kann aus dem Wahlpflichtkatalog ausgewählt werden. Auch fächerübergreifende Studiengänge bieten eine verstärkte Kompetenzbildung im Bereich Automotive Software an (beispielsweise Wirtschaft-

singenieurwesen Automotive an der Bergischen Universität Wuppertal).

Dieser Beitrag skizziert ein Praktikum, welches einige Aspekte des ASE vermittelt. Die inhaltlichen Anforderungen benötigen ein entsprechendes Vorwissen, so dass in erster Linie Bachelor-Studierende im letzten Studienjahr als Teilnehmer in Frage kommen.

Beschreibung des Moduls

Im folgenden Abschnitt wird das Praktikum mit der begleitenden Vorlesung vorgestellt und die zu vermittelnden Kompetenzen abgeleitet.

Teilnehmer

Das Praktikum ist Bestandteil des Moduls *Embedded Systems und Software Engineering* für Studierende im sechssemestrigen Studiengangs Mechatronik an der Hochschule Osnabrück. Das Modul im fünften Fachsemester besteht aus einer Vorlesung (2 SWS) und einem begleitenden Praktikum (2 SWS) und entspricht 5 Leistungspunkten nach ECTS.

Der Mechatronik-Studiengang an der Hochschule Osnabrück ist nicht anwendungsspezifisch und in der Ausbildung breit angelegt. Aufgrund der eingangs beschriebenen Bedeutung der Automobilindustrie in der Region Osnabrück werden Vorgehensweisen, Werkzeuge und Methoden praxisnah vermittelt.

Lernziele

Aufgrund des Zeitrahmens verzichtet das Praktikum auf den vollständigen Durchlauf aller Phasen im Software-Entwicklungsprozess. Stattdessen werden einzelne Schwerpunkte gesetzt, die sich aus dem spezifischen automobilen Vorgehen ergeben. Die nachfolgende Definition von Lernzielen anhand von Kompetenzfeldern orientiert sich an der Darstellung von (Böttcher und Thurner, 2011).

Sachkompetenz

Die Studierenden lernen das Vorgehensmodell in der Automobilindustrie kennen. Hierbei werden aufgrund der arbeitsteiligen Fahrzeugentwicklung die abgeleiteten Konsequenzen verdeutlicht. Neben einer integrierten Entwicklungsumgebung für das eingebettete System (µVision, Fa. Keil) wird ein Werkzeug zur Modellierung verteilter Systeme von Steuergeräten vermittelt. Dieses Werkzeug (CANoe, Fa. Vector) kann in unterschiedlichen Phasen (u.a. im Entwurf, Simulation, Test) des Entwicklungsprozesses eingesetzt werden.

Methodenkompetenz

Das Praktikum soll insbesondere das *Denken in Schnittstellen* als Methode zur Abstraktion und zur Modularisierung fördern. Die Betrachtung der einzelnen Steuergeräte als *Black Box* macht die Funktionalität an einer Schnittstelle sichtbar und vermeidet den Blick auf interne Details oder Abläufe in frühen Phasen der Entwicklung. Dieser Ansatz reduziert die Komplexität und lässt dadurch große Software-Systeme handhabbar werden.

Als zweiten Ansatz zur Reduzierung der Komplexität soll das *Denken in Zustandsmaschinen,* das Beschreiben von Verhalten, vertieft werden. Während Schnittstellen im Allgemeinen nur eine statische Sicht auf eine Komponente oder System beschreiben, ist für ein vollständiges Verständnis die dynamische Sicht entscheidend. Zustandsdiagramme ggfs. in Kombination mit Sequenzdiagrammen sind durch die Studierenden hierbei einzusetzen.

Sozialkompetenz

Die arbeitsteilige Entwicklung wird im Praktikum nachgestellt. Hierzu entwickeln mehrere Teams unterschiedliche Steuergeräte, die gemeinsam in ein Fahrzeug integriert werden. Da die Abnahme des Praktikums hierbei nur als Ganzes erfolgt, sind die Studierenden aufgefordert, über die eigenen Teamgrenzen hinweg offene Punkte anzusprechen und Lösungen gemeinsam zu erarbeiteten. Hierbei bilden die Schnittstellen üblicherweise die Reibungspunkte in der Zusammenarbeit. Zudem müssen die Teams sich zeitlich koordinieren, um gemeinsame Tests und Erprobungen durchzuführen.

Vorlesung

Nicht alle der vorgenannten Grundlagen können in diesem Bachelor-Modul vorausgesetzt werden und sind daher Bestandteil des ersten Teils der begleitenden Vorlesung. Die Schwerpunkte sind neben dem grundsätzlichen Aufbau von Echtzeitsystemen hierbei: Hardware-nahe Software-Entwicklung und Betriebssysteme (Scheduling-Verfahren sowie Zugriff auf gemeinsame Ressourcen).

Um die Studierenden auf die Bedeutung der Automobilindustrie in Deutschland hinzuführen, beginnt der zweite Teil mit einem Exkurs zum Thema Technologien *Made in Germany*. Hierbei werden exemplarisch die Errungenschaften und Erfindungen von Werner von Siemens (Dynamoelektrisches Prinzip, 1867), Robert Bosch (Zündkerze, 1902) und Konrad Zuse (programmgesteuerter Rechner, 1941) vorgestellt. Hieran schließt sich eine Frage zur Aktivierung der Studierenden an: *Was sind aus Ihrer Sicht die wichtigsten Erfindungen aus Deutschland der letzten 30 Jahre?*

Nur wenige der Studierenden haben hierzu eine eigene Vorstellung. Offensichtlich sind die meisten Technologien wie Suchmaschinen, soziale Netzwerke oder moderne Mobiltelefone von amerikanischen Unternehmen erfunden und entwickelt worden. Nur wenigen Studierenden fällt spontan das

in Deutschland entwickelte MP3-Format (Karlheinz Brandenburg, 1987) ein. Dass das *Controller Area Network* oder abgekürzt CAN (Siegfried Dais und Uwe Kiencke, 1985) eine deutsche Technologie ist, die inzwischen in jedem Fahrzeug weltweit eingesetzt wird, ist den Studierenden unbekannt.

In den nachfolgenden Lehreinheiten werden die CAN-Bus Grundlagen vermittelt und somit die Voraussetzungen für das Praktikum geschaffen.

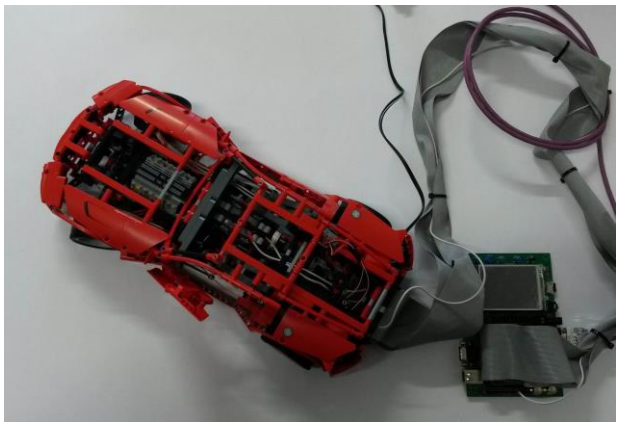

Abb. 3: Erweitertes Modellfahrzeug mit CAN

Praktikum und Aufgabenstellung

Eine vollständige Funktionsentwicklung eines Steuergerätes ist aufgrund der zur Verfügung stehenden Zeit und der erforderlichen Voraussetzungen nicht Schwerpunkt des Moduls. Stattdessen wird ein einfaches Beleuchtungssystem (Blinker, Fahrlicht, etc.) als anschauliches Beispiel eingesetzt. Dieses bietet den Vorteil, dass die Anforderungen einfach darzustellen sind, da viele der Studierenden einen Führerschein und ggfs. ein eigenes Fahrzeug besitzen.

Abbildung 3 zeigt das im Praktikum eingesetzte Modellfahrzeug (Lego Super Car 8070). Dieses um eine Lichtsteuerung sowie eine Motoransteuerung für die Fahrertür erweitert worden. Der dargestellte Mikrocontroller empfängt über den CAN-Bus Nachrichten zur Ansteuerung sowie versendet periodisch Sensordaten (Helligkeit).

Insgesamt sind 6 Steuergeräte in einer vereinfachten Ausbaustufe in einem Fahrzeug zu integrieren:

- Bedieninstrument
- Getriebe
- ABS (Bremse)
- Zentralverriegelung
- Zündschloss
- Licht

Die zuletzt aufgeführten ECU werden als Simulation bzw. als eingebettetes System für das Modellfahrzeug vorgegeben. In Abbildung 4 ist der voll-

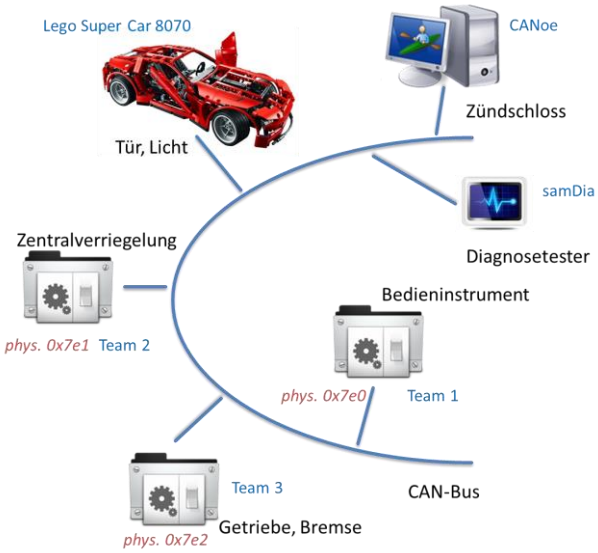

Abb. 4: Arbeitsteilige Entwicklung im Praktikum

ständige Praktikumsaufbau skizziert. An einem gemeinsamen CAN-Bus ist das Lego Super Car 8070, die CANoe-Simulation für das Zündschloss und 4 weitere Steuergeräte angeschlossen, die das Bedieninstrument, die Zentralverriegelung und das Getriebe sowie die Bremse realisieren.

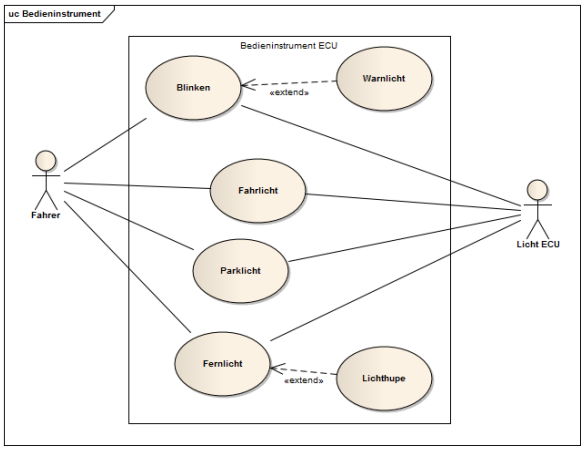

Abb. 5: Anwendungsfälle für Bedieninstrument

Die Phase der Anforderungsanalyse im Entwicklungsprozess wird durch Vorgabe von Anwendungsfällen (siehe Abbildung 5) und textuellen Beschreibungen verkürzt. Nichtsdestotrotz ergeben sich Spielräume in der Ausgestaltung der Lichtfunktionen (beispielsweise Abschaltung des Abblendlichtes während Fernlicht).

Aus den vorgegebenen Anforderungen haben die einzelnen Teams das entsprechende Steuergerät zu entwickeln. Die Realisierung erfolgt in vier Phasen entsprechend dem in der Automobilindustrie angewendeten Vorgehen:

1. Vollständige Simulation: die grundsätzliche Funktionsweise des zu entwickelnden

Steuergeräts wird überprüft und die Interaktion mit dem Lego Super Car 8070 getestet (siehe Abbildung 6).

2. Reales Steuergerät mit simulierter Umgebung: die Simulation des zu entwickelnden Steuergeräts wird durch eine Implementierung auf einem eingebetteten System ersetzt.
3. Reale Steuergeräte (ohne Simulation): die simulierte Umgebung wird durch das Fahrzeug (Steuergerät für Beleuchtung und Tür) sowie durch weitere Steuergeräte ersetzt. Die Zündung wird weiterhin über CANoe simuliert.
4. Diagnose: Die Funktionalität des Steuergerätes wird um Möglichkeiten zur Diagnose erweitert. Hierbei wird in Abhängigkeit des Steuergerätes auf Anfragen des Diagnosetesters geantwortet.

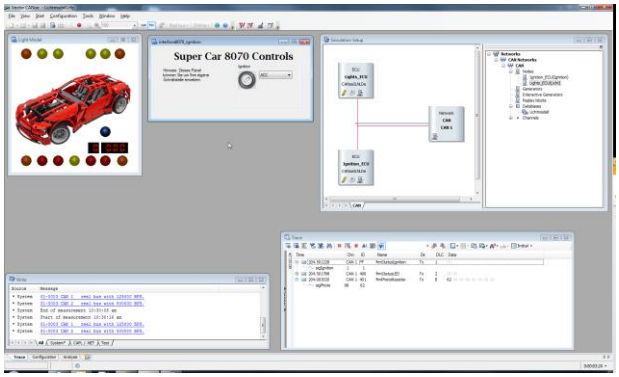

Abb. 6: Modellierung und Simulation mit CANoe

Durch die detaillierte Vorgabe der Anforderungen können die Studierenden eine manuelle Verifikation in den unterschiedlichen Phasen ihrer Software-Entwicklung durchführen. Zusätzlich werden simulierte Komponenten bereitgestellt, so dass Tests teilautomatisiert werden können.

Erfahrungen und Empfehlungen

Die folgenden Abschnitte fassen die Beobachtungen und Erfahrungen zusammen. Hieraus lassen sich Empfehlungen für ähnliche Praktika ableiten.

Konzentration auf wenige Aspekte

Ein häufiger Ansatz in der Lehre als auch in der Literatur ist die Vorstellung von verschiedenen Entwicklungsprozessen, vom Wasserfallmodell, iterative Ansätzen, V-Modell oder agilen Ansätzen. Hierbei werden die einzelnen Phasen vertieft. Oftmals werden zusätzlich bestimmte Methoden, beispielsweise zur Modellierung in UML von Software, miteingeführt.

Aus den Lernzielen abgeleitet beschränkt sich die durchgeführten Praktikumsversuche auf der Vorstellung sowie der Vertiefung nachfolgender Aspekte:

- Die Beschreibung von Verhalten durch Zustandsmaschinen; hierbei entstehen auch verteilte Zustandsmaschinen, da der Systemzustand nicht zentral sondern verteilt in den einzelnen Netzknoten vorliegt.
- Durch das Zusammenspiel von jeweils drei Teams ergibt sich eine arbeitsteilige Entwicklung der Steuergeräte.
- Entwurf von Schnittstellen und die Umsetzung von Nachrichten für den CAN-Bus.
- Entwurfsprozess: vollständige Simulation der Netzknoten, Kombination aus einer Zielhardware und der Restbussimulation und gemeinsame Integration aller Steuergeräte.

Die Fokusbestimmung macht einen Auswahlprozess notwendig, der auch die Voraussetzungen der Studierenden berücksichtigt. So kann beispielsweise bei Informatikern der Schwerpunkt eher auf der Modellierung von Softwarekomponenten, bei Ingenieuren ggfs. eher im Bereich der Steuerungs- und Reglungstechnik liegen.

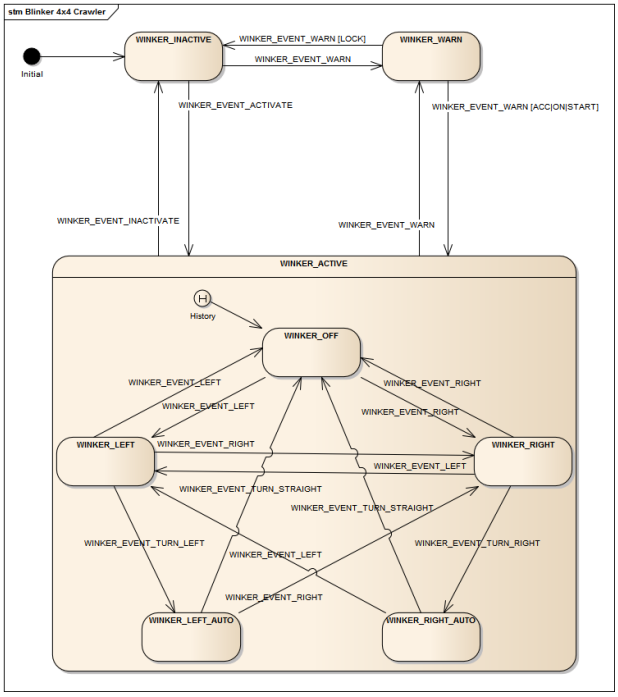

Abb. 7: Zustandsmaschine für Blinker

Abbildung 7 gibt beispielhaft die Zustandsmaschine für den Blinker mit einer automatischen Abschaltung wieder. Die Studierenden erlernen aus einer textuellen Anforderungsbeschreibung das Verhalten zu modellieren. Ausgehend von einfachen Zustandsmaschinen werden weiterführende Methoden (Hierarchie und Historie) vermittelt, so

dass komplexe Verhaltensmuster angemessen dargestellt werden können.

Dosierter Einsatz von Werkzeugen

Besonders in der modellbasierten Entwicklung im automotiven Umfeld existiert eine Reihe von sehr spezialisierten Werkzeugen. Ohne den Werkzeugeinsatz ist beispielsweise eine Software Entwicklung für AUTOSAR-kompatible Komponenten nicht denkbar.

Die folgenden Nachteile ergeben sich hieraus:

- Die Werkzeuge bieten im Allgemeinen keinen Lernmodus o.ä. an und orientieren sich an den konkreten Aufgaben in der beruflichen Praxis. Ein pädagogischer Einstieg ist nicht möglich.
- Die graphische Bedienschnittstelle ist stark technisch ausgerichtet und orientiert sich an den Belangen der Industrie (*Expertenwissen*).
- *A fool with a tool is still a fool*: Aufgrund der Abstraktion, die Werkzeuge bei der Entwicklung einführen, verlieren Studierende Transparenz in ihrem Vorgehen. Konsequenzen bleiben unklar. Bei Fehlern *funktioniert etwas nicht* und eine Ursachensuche findet dann nicht statt.
- Ein Zwiespalt in der Ausbildung: zum einen sollen Studierende Methoden und Kompetenzen erlernen und nicht zum *Werkzeugbediener* ausgebildet werden. Zum anderen soll praxisnah und berufsbefähigend ausgebildet werden. Hierzu gehören so dann auch Kenntnisse solcher industriellen Werkzeuge.

Schon der zeitliche Aufwand steht einem verstärkten Einsatz von Werkzeugen im Wege. Viele Werkzeughersteller bieten 3-5 tägige Schulungen zu Ihren Produkten an. Diese entspricht ungefähr dem Umfang einer vollständigen 2 SWS Veranstaltung.

Abb. 8: Physisches Lichtmodell auf Basis Lego

In dem vorliegen Modul werden neben einer integrierten Entwicklungsumgebung (im Wesentlichen nur zum Editieren und zum Übersetzen) ein weiteres Werkzeug CANoe eingesetzt, welches es ermöglicht, einen typischen Entwicklungszyklus im Automotive Software Engineering zu durchlaufen.

Verwendung von physischen Modellen

Ein immer wieder nicht zu unterschätzender Faktor ist die Benutzung von physischen Modellen in der Ausbildung von Studierenden. Dieser positive Effekt auf die Motivation der Studierenden wird auch anderweitig beobachtet (Uelschen und Eikerling, 2011). In dem Praktikum werden keine realen Steuergeräte oder Fahrzeugkomponenten entworfen, sondern dieses wird durch ein Lego-Technik Modell (siehe Abbildung 8) in sehr einfacher Weise ersetzt.

Obwohl die Entwicklung und der Test auch vollständig auf dem Desktop-Rechner in einer Simulation erfolgen können, hat der Einsatz von haptischen Modellen Vorteile:

- Die Studierenden werden aus ihrer *Lebenswelt* abgeholt. Sie haben die Modelle in Ihrer Jugendzeit in einer anderen Umgebung kennengelernt.
- Reale Modelle sind weniger abstrakt als simulierte Anzeigen auf dem Bildschirm und verdeutlichen den Studierenden einen Einsatz in der Praxis.

Das in (Nazareth und Wurm, 2013) verwendete Modellfahrzeug im Maßstab 1:18 wird als Plattform zur modellbasierten Entwicklung eingesetzt.

Im Wintersemester 2014/15 ist ein weiteres Fahrzeugmodell im Praktikum eingesetzt worden (siehe Abbildung 9), welches teilweise eine alternative Funktionsweise zur Beleuchtung realisiert (beispielsweise automatische Rückstellung der Blinker bei Geradeausfahrt). Zudem ist die Ansteuerung der LEDs platzsparend auf der Basis Arduino mit CAN-Shield realisiert.

Abb. 9: Alternativ eingesetztes Lichtmodell

Fazit

Der vorliegende Beitrag zeigt, wie spezifische Aspekte des Automotive Software Engineering in einem Bachelor-Modul integriert werden können. Der Schwerpunkt liegt hierbei auf der arbeitsteiligen Entwicklung, den Entwurf von Schnittstellen und der dynamischen Modellierung mit Zustandsmaschinen. Bei der Entwicklung von Steuergeräten zu Ausbildungszwecken kann im Allgemeinen nicht auf reale Komponenten der Automobilindustrie zurückgegriffen werden, da die Schnittstellen nicht öffentlich und notwendige Informationen beim OEM nicht erhältlich sind. Stattdessen muss auf einfache Modelle zurückgegriffen werden.

Das Praktikum ist in der vorgestellten Weise inzwischen dreimal erfolgreich durchgeführt worden. Zukünftig soll die Verifikationsphase vertieft werden, da sich herausgestellt hat, dass ein systematischer Test bei den Studierenden insbesondere in den ersten Phasen verkürzt wird.

Das Praktikum bietet zudem zahlreiche Möglichkeiten zur Erweiterungen. Zur Vertiefung von Zustandsmaschinen kann beispielsweise das Netzmanagement in die Aufgabenstellung integriert werden. Die Anforderungsanalyse ist durch die Vorgabe von konkreten Anwendungsfällen im aktuellen Praktikum stark verkürzt. Alternativ besteht die Möglichkeit, die Anforderungen aus der Straßenverkehrsordnung durch die Studierenden ableiten zu lassen.

Literatur

Böttcher, A., Thurner, V. (2011): Kompetenzorientierte Lehre im Software Engineering. In: Ludewig, J., Böttcher, A. (Hrsg.), Software Engineering im Unterricht der Hochschulen (SEUH), München, CEUR Workshop Proceedings Vol. 695, http://ceur-ws.org/Vol-695/, S. 33-39.

Broy, M. (2006): Challenges in Automotive Software Engineering. In: Proceedings of the 28th International Conference on Software Engineering (ICSE '06), ACM, S. 33-42.

Chrissis, M., Konrad, M., Shrum, S. (2011): CMMI for Development: Guidelines for Process Integration and Product Improvement, 3rd edition, Addison-Wesley.

Fortiss (2011): Mehr Software (im Wagen). Abschlussbericht des vom Bundesministerium für Wirtschaft und Technologie geförderten Verbundvorhabens eCar-IKT-Systemarchitektur für Elektromobilität.

Gebhardt, V., Rieger G., Mottok, J., Gießelbach C. (2013): Funktionale Sicherheit nach ISO 26262: Ein Praxisleitfaden zur Umsetzung, dpunkt.

Höhn, H., Sechser B., Dussa-Zieger K., Messnarz, R., Hindel B. (2009): Software Engineering nach Automotive SPICE, dpunkt.

Lucke, H., Schaper, D., Siepen, P., Uelschen, M., Wollborn, M. (2007): The Innovation Cycle Dilemma. In: Koschke, R., Herzog, O., Rödiger, K.-H., Ronthaler, M. (Hrsg.), INFORMATIK 2007, Band 2, S. 526-530.

Mössinger, J. (2010): Software in Automotive Systems. In: Software, IEEE, S. 92-94.

Nazareth, D., Wurm, C. (2013): Modellbasierte Entwicklung einer Lichtsteuerung für ein Rapid Prototyping System. In: Halang, W. (Hrsg.), Kommunikation unter Echtzeitbedingungen, Springer, S. 49-58.

Pfleging, B., Schneegass S., Kern, D., Schmidt, A. (2014): Vom Transportmittel zum rollenden Computer – Interaktion im Auto. In: Informatik Spektrum, GI, S. 418-422.

Pretschner, A., Broy, M., Krüger, I., Stauner, T. (2007): Software Engineering for Automotive Systems: A Roadmap. In: Future of Software Engineering 2007, IEEE, S. 55-71.

Ritter, S. (2004): Software auf der Straße: Herausforderungen des Software-Engineering in der Automobilindustrie. In: Objektspektrum, 4, S. 59-62.

Schäuffele, J., Zurawka, T. (2012): Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen, 5. Auflage, Springer.

Sommerville, I., (2012): Software Engineering, 9. Auflage, Pearson.

Uelschen, M., Eikerling, H.-J. (2011): An Introductory Course on Software Engineering on Self-Organization in Swarm Robotics. In: Proceedings of the 24th Conference on Software Engineering Education and Training (CSEE&T), IEEE, S. 333-342.

Weinmann, U. (2003): Software im Automobil – Anforderungen und Chancen. In: Siedersleben, J., Weber-Wulff, D. (Hrsg.), Software Engineering im Unterricht der Hochschulen (SEUH), dpunkt Verlag, Heidelberg, S. 1-7.

Winner, H., Hakuli, S., Wolf, G. (2012) (Hrgs.): Handbuch Fahrerassistenzsysteme , 2. Auflage Springer.

Zimmermann W., Schmidgall R. (2014): Bussysteme in der Fahrzeugtechnik, 5. Auflage, Springer.