Entwicklung des Elektronischen Stabilitäts-Programms (ESP) für Nutzfahrzeuge unter ASCET-SD

Dr. Detlef Zerfowski
Knorr-Bremse Systeme für Nutzfahrzeuge GmbH
Email: Detlef.Zerfowski@Knorr-Bremse.com

1 Klassische SW-Entwicklung für eingebettete Systeme

Die Disziplin des Software-Engineering ist im Vergleich zu anderen Ingenieurdisziplinen sehr jung und unterliegt aus diesem Grund noch großen Veränderungen. Die steigende Komplexität der Software steht dabei meist konträr zur SW-Qualität und zu einer möglichst kurzen Entwicklungszeit. Analog zu anderen Entwicklungsbereichen versucht man bei der SW-Entwicklung dieser Diskrepanz durch den Einsatz automatisierender Methoden und Verfahren zu begegnen. Eine besondere Stellung nimmt dabei die SW-Entwicklung für eingebettete Systeme ein. Im Vergleich zu PC-basierten Systemlösungen besitzen eingebettete Systeme in der Regel eine einfache (teilweise keine) Benutzerschnittstelle, sowie weniger komplexe Programmstrukturen. Erschwerende Randbedingungen für den Entwurf eingebetteter Systeme sind jedoch Anforderungen an das Echtzeitverhalten, die Systemsicherheit und die Robustheit gegenüber externen Einflüssen.
Die nicht standardisierte, anwendungsspezifische Hardware behindert zudem eine stärkere Automatisierung der SW-Entwicklung. So wird bis heute in vielen Anwendungen hardwareabhängiger Assemblercode implementiert. Einerseits ist dies durch die Kenntnisse und Gewohnheiten der Entwickler begründet, andererseits wurde der Assemblereinsatz durch die Restriktionen an die Anwendungssoftware bezüglich Speicheranforderungen und Laufzeitverhalten erzwungen.
Die offensichtlichen Nachteile der Assemblerprogrammierung wie

- geringe Portabilität und damit
- geringe Wiederverwendbarkeit (insbesondere bei einem Prozessorwechsel), sowie
- schlechte Wartbarkeit (Lesbarkeit),

führte in der Vergangenheit zum verstärkten Einsatz höherer Programmiersprachen wie C.
Verstärkt wurde dieser Trend mit der zunehmenden Verfügbarkeit und Zuverlässigkeit leistungsstarker Hochsprachen-Compiler für die unterschiedlichen Zielprozessoren.
Die zunehmende SW-Komplexität erfordert den Einsatz eines festgelegten organisatorischen Rahmens für den SW-Entwicklungsprozess. Ein solches Prozessmodell stellt das V-Modell dar, das neben den eigentlichen Entwicklungsschritten auch die Verifikation und Validation der Software beinhaltet.
Der Entwicklungsprozess ist dabei in mehrere Phasen unterteilt. Beginnend mit einer Anforderungsdefinition beziehungsweise Spezifikation wird über einen Grob- und Feinentwurf zur Modulimplementierung übergegangen. In jeder Entwurfsphase werden Testfälle definiert, die die einzelnen Entwicklungsschritte überprüfbar halten.

2 Einsatz von ASCET-RSF beim Anhängersteuermodul (ASM)

Im Rahmen der Entwicklung des Anhängersteuermoduls wurde bei Knorr-Bremse der erste Schritt in Richtung eines neuen Entwicklungsprozesses gegangen.
Die zentrale Aufgabe des Anhängersteuermoduls besteht darin den Bremsdruck des Anhängers zu kontrollieren und zu steuern. Insbesondere werden durch eine Koppelkraftregelung die mechanischen Kräfte zwischen Zugmaschine und Anhänger durch entsprechende Verteilung der Bremskräfte optimiert.

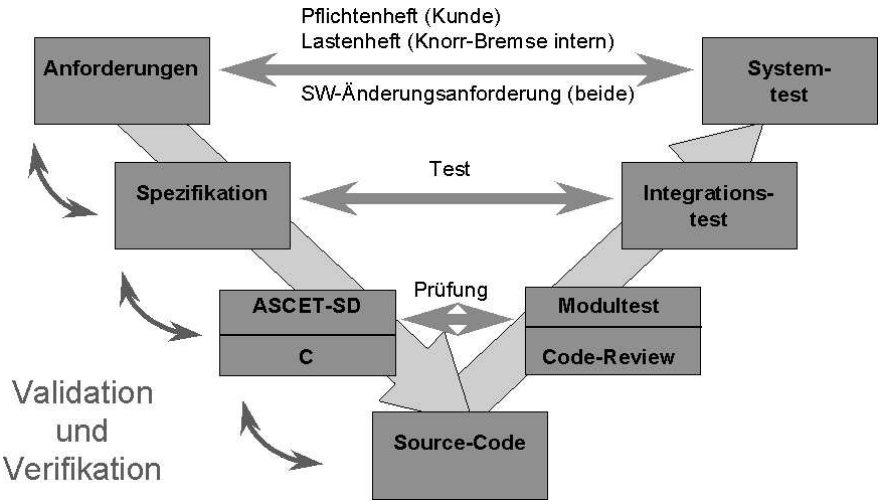

Abbildung 1: V-Modell

Die Funktionsspezifikation, sowie der Grob- und Feinentwurf des ASM wurden in ASCET-RSF, dem Vorgänger von ASCET-SD, erstellt. Es handelte sich dabei um ein CASE-Werkzeug das die graphische Spezifikation der zu implemtierenden Regelungsalgorithmen und deren Entwicklung erlaubte.
Es bestand keine Möglichkeit der automatischen Codegenerierung und erforderte somit für die Zielhardware eine manuelle Reimplementierung der Funktionen in C. Neben der zeitaufwendigen Umsetzung in C ergab sich als entscheidender Nachteil, dass das in ASCET-RSF vorliegende Modell, bedingt durch funktionale Änderungen im implementierten C-Code, binnen kürzester Zeit nicht mehr aktuell war, da die Nachführung der Änderungen im ASCET-RSF-Modell nicht zu erzwingen war. Der anfangs vorhandene Vorteil einer guten graphischen Spezifikation der Steuerungssoftware war somit nicht mehr gegeben. Als Fazit ergab sich die Notwendigkeit eines durchgängigen Entwicklungsprozesses von der Spezifikation bis zum Seriencode.

3 Pilotprojekt Elektronisches Stabilitäts-Programm (ESP)

Aufgrund der in den vorhergehenden Abschnitten dargestellten Problematik entschied sich Knorr-Bremse SfN im Rahmen eines Pilotprojektes für den Einsatz von ASCET-SD. Es handelt sich dabei um die Serienentwicklung eines, aus dem PKW-Sektor hinlänglich bekannten, Elektronischen Stabilitäts-Programms (ESP) für den Nutzfahrzeugbereich.

3.1 ESP für Nutzfahrzeuge

Die Grundaufgabe eines ESP besteht darin, rechnergesteuert ein durch den Fahrer oder die Umweltsituation induziertes kritisches Fahrverhalten des Fahrzeuges in einem kontrollierbaren Zustand zu halten. Dabei unterscheiden sich die Anforderungen an ein ESP für Nutzfahrzeuge von denen im PKW-Einsatz. Im Gegensatz zum PKW-ESP ergeben sich eine Reihe zusätzlicher Einflußgrößen auf die Fahrdynamik des Fahrzeuges, wie z. B.:

- zwei gekoppelte Fahrzeuge,
- veränderliche Gesamtmasse des Fahrzeuges in einem großen Wertebereich (bedingt durch Be- und Entladung mit unterschiedlichen Gütern),
- in drei Dimensionen örtlich variierender Masseschwerpunkt,

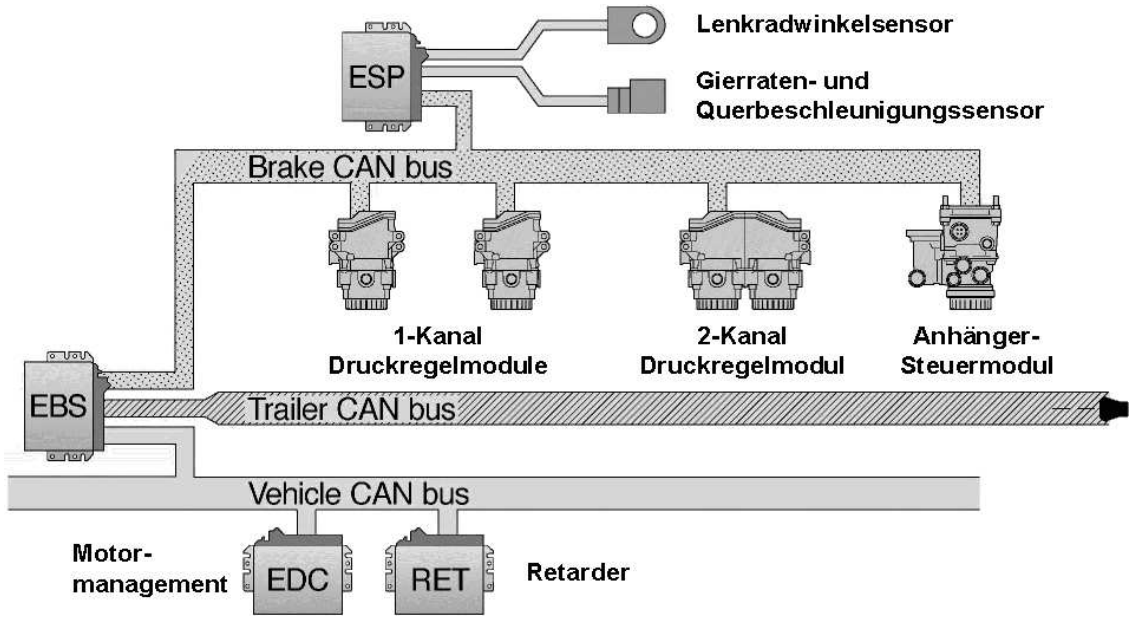

Abbildung 2: EBS-Systemübersicht.

- unterschiedliche Fahrzeugkonfigurationen, da Zugmaschine mit unterschiedlichen Auflieger bzw. Anhänger mit verschiedenen Achsen- (Liftachsen) und Bremskonfigurationen (z. B. 1- oder 2-Kanal Bremssysteme) betrieben werden,
- unbekannte Fahrzeugaufbauten und -einsatzgebiete.

Das ESP stellt eine Teilkomponente eines Elektronischen Brems-Systems (EBS) dar, das in Abbildung 2 dargestellt ist. Das ESP-Steuergerät kommuniziert über den Bremsen-CAN mit dem EBS-Steuergerät.
Auf die in der EBS-Zentrale integrierten Funktionen des Anti-Blockiersystems (ABS) und der Antriebs-Schlupf-Regelung (ASR) besitzt das ESP über entsprechende Schnittstellen Eingriffsmöglichkeiten.
Weiterhin besitzt das von Knorr-Bremse entwickelte ESP ein Dynamisches Stabilitäts-Programm (DSP). Die Aufgabe des DSP besteht darin, auf mittlerem und niedrigem Reibwert (μ)

- das Einknicken (Jack-Knifing) von Zugmaschine und Auflieger zu verhindern, sowie
- das Unter- und Übersteuern von Zugmaschinen bzw. Zugmaschinen mit Hängern innerhalb der physikalischen Möglichkeiten zu vermeiden.

Hierzu bremst das DSP selektiv die Räder an denen der Aufbau der längsgerichteten Bremskräfte und der Verlust an Seitenführungskräfte denselben Effekt auf das Drehmoment des Fahrzeuges besitzen.
Im Fall des Übersteuerns der Zugmaschine werden hauptsächlich die vordere, äußere Rad eingebremst. Zusätzlich wird in speziellen Situationen Situationen der Anhänger eingebremst, um das Einknicken des Fahrzeuges zu verhindern.
Im Fall des Untersteuerns wird hauptsächlich das hintere innere Rad eingebremst.
Die Roll-Over-Prevention (ROP) vermeidet im Rahmen der physikalischen Grenzen das Umkippen des Fahrzeuges. Dabei arbeitet die ROP in zwei Schritten:

1. Reduzierung der Fahrzeuggeschwindigkeit um die Querbeschleunigung des Fahrzeuges in einem unkritischen Bereich zu halten,
2. Erkennen des Radabhebens an der Zugmaschine bzw. am Anhänger (im Falle eines EBS-Anhängers).

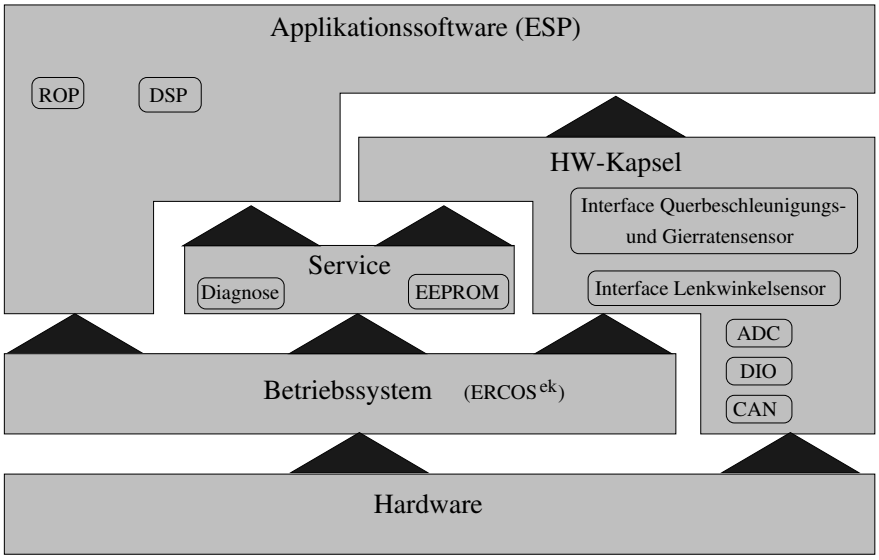

Abbildung 3: Softwarearchitektur: Übersicht über die unterschiedlichen SW-Ebenen.

Um das aktuelle dynamische Verhalten des Fahrzeuges bestimmen zu können, ist analog zum Anhängersteuermodul eine Masseschätzung realsisiert.
Das Vorhandensein mehrerer Regelungsfunktionen kann dazu führen, dass konkurrierende Regeleingriffe angefordert werden. Hieraus ergibt sich die Notwendigkeit für entsprechend angepasste Regelstrategien.
Da es sich bei einem elektronischen Bremssystem um eine sicherheitskritische Steuerungskomponente handelt, werden zu den zuvor erwähnten Funktionen zusätzliche Sicherheits- und Überwachungsfunktionen hinzugefügt. Deren Aufgabe bestehen unter anderem in der Aufdeckung hardwarebedingter Fehlfunktionen, sowohl auf Steuergeräteebene (CPU-Fehler, ROM- und RAM-Fehler, Kommunikationsüberwachung, etc.) als auch auf Sensor- und Aktuatorebene (Überwachung der Geschwindigkeits- und Drucksensoren, der Versorgungsspannung, etc.)

3.2 SW-Architektur ESP

Ausgangspunkt bei der Realisierung des ESP war eine Strukturierung der SW-Architektur wie sie in Abbildung 3 wiedergegeben wird. Es ergab sich dabei eine natürliche Aufteilung der SW-Komponenten in applikationssnahe Module (z. B. Regelungsalgorithmen), die über wohldefinierte Schnittstellen Zugriff auf hardwarenah programmierte Module, der sogenannten SW-Plattform, besitzen. Letztere dient der Kapselung der Zielhardware und bietet dünne Schnittstellen für Zugriffe auf die Zielhardware, um ein hohes Maß an Wiederverwendbarkeit der Applikationsfunktionen und eine einfachere und schnellere Übertragung der SW-Plattform auf andere Zielrechner zu gewährleisten.
Es wurde nicht die gesamte Steuergerätesoftware in ASCET-SD modelliert. Da die Vorteile dieses Entwicklungswerkzeuges im Bereich der Applikationsfunktionen liegt, wurde die SW-Plattform manuell in C codiert. Von besonderer Bedeutung war dabei eine durchdachte Strukturierung und Modullarisierung der Plattform-SW, so dass eine Sammlung von voneinander unabhängigen C-Modulen entstand. Diese sind jeweils einzelnen funktionalen Komponenten (z. B. Digital-I/O, A/D-Converter, SSC-Kommunikation, CAN-Kommunikation, EEPROM-Service-Routinen, etc.) der verwendeten CPU und der sie umgebenenden Hardware zugeordnet.
Die in der SW-Plattform generierten Objekt-Dateien werden in ASCET-SD als externer Objekt-Code eingebunden und die einzelnen Routinen über die in ASCET-SD verwaltete Taskliste aufgerufen. Als Echtzeitbetriebssystem kommt dabei ERCOS^{EK} zum Einsatz.

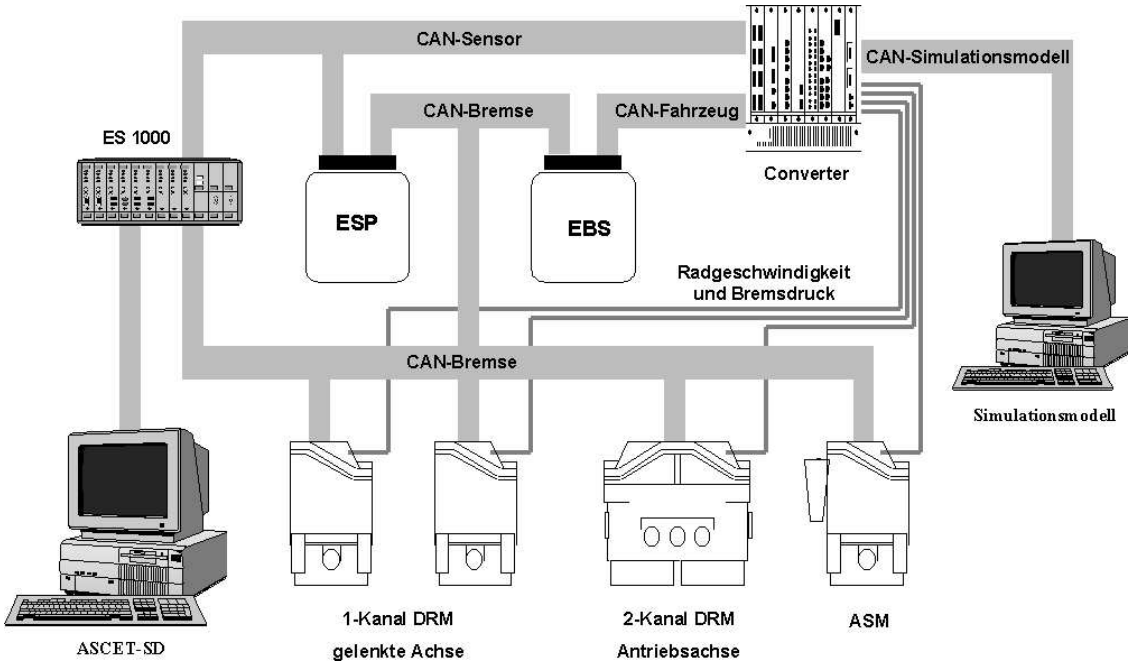

Abbildung 4: Aufbau des EBS/ESP-Prüfstandes.

3.3 Funktionsentwicklung unter ASCET-SD

Entscheidend für den Einsatz von ASCET-SD war die erstmals zur Verfügung stehende Durchgängigkeit des Entwicklungsprozesses, beginnnend bei der Spezifikation bis hin zum Seriencode. Bereits frühzeitig in der Prototypenphase können durch Simulation Funktionen auf ihr korrektes Verhalten hin überprüft werden, ohne dass bereits eine konkrete Zielhardware vorliegt. Mögliche Designfehler können somit in einem sehr frühem Stadium aufgedeckt und behoben werden. Dieser sogenannte Fullpass-Betrieb hat es uns erlaubt, Funktionen im Fahrzeug zu testen, ohne dass eine Ziel-Hardware vorhanden war.
Durch die ständige Fortentwicklung der Funktionen innerhalb dieses Entwicklungssystems ist kein Übergang zwischen Entwicklungswerkzeugen mehr notwendig, der zu einem Verlust vorhergehender Implementierungsinformation führen würde.
Die frühzeitige Simulation der Integer-Implementierung (ohne vorhandene Ziel-Hardware) erlaubt die Berücksichtigung von Quantisierungseffekten in frühen Entwicklungsphasen. Zeitintensive und fehleranfällige SW-Änderungen aufgrund nicht vorhersehbarer Quantisierungeffekte in späten Entwicklungsphasen werden somit vermieden.
Die zuvor geschilderten Schritte decken im V-Modell die Phasen bis hin zum Feinentwurf ab. Der abschließende Schritt der Implementierung im Steuergerätecode wird durch die automatische Codegenerierung von ASCET-SD realisiert. Die zeitaufwendige Reimplementierung, wie bei der Verwendung von ASCET-RSF, fällt nicht an und verkürzt dadurch die Entwicklungszeit (vgl. auch Abbildung 5). Die vorliegenden, abstrakteren Implementierungen in ASCET-SD bilden weiterhin die Basis für die Entwicklung und bleiben somit aktuell.

3.4 Prüfstand mit ASCET-SD

Eine besondere Bedeutung in der SW-Entwicklung nimmt die Testphase unter realitätsnahen Bedingungen ein. Da SW-Tests in Fahrzeugen zeitaufwendig, kostenintensive und zudem schlecht reproduzierbar sind, wurde bei Knorr-Bremse ein Prüfstand unter Verwendung von ASCET-SD aufgebaut.
Ausgehend von dem in Abbildung 2 gezeigtem Systemaufbau wurde der in Abbildung 4 gezeigte Prüfstand aufgebaut, der eine realistische Simulation von Fahrmanövern unter verschiedenen Bedingungen zulässt.

Hierzu kommt ein eigenentwickeltes Fahrzeugsimulationsmodell auf einem separaten PC zum Einsatz. Dieses Modell erlaubt die Spezifikation unterschiedlicher Fahrzeugkonfigurationen, Umwelteinflüsse (z.B. unterschiedliche Reibwerte für μ) und Fahrmanöver, die in Echtzeit anhand eines sich bewegenden Fahrzeuges als 3D-Gittermodell visualisiert werden. Das Simulationsmodell liefert über den Simulation-Modell-CAN die Daten an den Konverter, die in einem realen Fahrzeug seitens der Sensoren zur Verfügung gestellt würden. Der Konverter realisiert neben der Umsetzung der Simulationsdaten auf den Sensor-CAN und Fahrzeug-CAN auch die Konvertierung der von den im Prüfstand verbauten Druckregelmodulen (PCM, Pressure Control Module) und Anhängersteuermodul (ASM, engl.: Trailer Control Module, TCM) gelieferten Analogsignale.
Im beschriebenen Prüfstand wird als Hardware-in-the-Loop ein ES1000-Systems mit ASCET-SD eingesetzt, das selektiv Teilfunktionen der ESP-Software außerhalb des ESP-Steuergerätes in der Experimental-Hardware ablaufen lassen kann (Bypass-Betrieb). Somit ist es möglich, ESP-Funktionalität z. B. in noch nicht-quantisierter Implementierung im Gesamtsystem zu testen. Die dafür notwendige Kommunikation zwischen dem ESP-Steuergerät und der ES1000-Hardware erfolgt dabei über den Sensor-CAN.
Analog zur Abbildung 4 werden ASCET-SD-Systeme über eine Hardware-in-the-Loop-Einbindung auch direkt im Fahrzeug eingesetzt. Die bei realen Fahrmanövern gewonnenen Messdaten dienen anschließend als Grundlage für die am Prüfstand durchgeführten Testfahrten.

3.5 Probleme mit ASCET-SD

Neben den beschriebenen Vorteilen die der Einsatz von ASCET-SD innerhalb des Entwicklungsprozesses mit sich bringt, besteht noch die Notwendigkeit für einige Verbesserungen an ASCET-SD.
Im Hinblick auf eine SW-Entwicklung, die in allen Belangen den Forderungen der QS-9000-Richtlinien genügt, ist der Einsatz eines Konfigurationsmanagements unabdingbar. Die derzeitige Konzentration auf einen KM-Anbieter ist der weiteren Verbreitung des Entwicklungswerkzeuges nicht förderlich.
In der Version 3.0 hat sich die Stabilität von ASCET-SD deutlich verbessert. Zudem wurde durch entsprechende Optimierungsschritte bei der Codegenerierung die Laufzeit des generierten Codes und die erzeugte Codegröße erheblich verringert. Dieses verursachte jedoch eine drastische Steigerung der Codeerzeugungszeiten. Kleine Modelländerungen, wie das Hinzufügen einer zusätzlichen Message, führen zu vollständigen Neuübersetzungen des Gesamtmodells und damit zu unerwünscht langwierigen Entwicklungszyklen. Ein Großteil der Übersetzungszeit wird dabei für die Modellanalyse benötigt.
Ein weiteres Problem ergab sich aus der unterschiedlichen Behandlung von Interruptroutinen in dem von ASCET-SD verwendeten Betriebssystem ERCOS^{EK} und den extern zugebundenen C-Modulen. ERCOS^{EK} nutzt eine eigene Stackverwaltung, die unter anderem sämtliche notwendigen Operationen zum Retten bzw. Restaurieren von Prozessorzuständen vor bzw. nach aufgerufenen Interruptroutinen realisisiert. Tritt bei der Abarbeitung eines Interrupts in den externen C-Modulen (diese Module verwenden einen anderen Stackmechanismus) an einer ungünstigen Stelle ein von ERCOS^{EK} verwalteter Interrupt auf, so kann dies zu einem unkontrollierten Verhalten des Steuergerätes führen. Dieses Problem kann derzeit nur dadurch behoben werden, dass beim Eintritt in bzw. beim Verlassen der Interruptroutinen innerhalb von externen C-Modulen die ERCOS^{EK}-spezifischen Interrupteinbzw. Interruptausgangsroutinen aufgerufen werden.

4 Erwartungen und Aussichten

Durch den Einsatz von ASCET-SD gehen wir von einer deutlichen Effizienz- und Qualitätssteigerung im Bereich der SW-Entwicklung für eingebettete Systeme aus, einhergehend mit einem entsprechenden Return-on-Investment (ROI). Abbildung 5 gibt dies anhand einer Gegenüberstellung zwischen den Entwicklungszeiten mit und ohne ASCET-SD wieder.
Die Durchgängigkeit des Entwicklungsprozesses „erzwingt“ einen strukturierten Entwurf, sowie eine stets aktuelle Dokumentation der entwickelten Funktionen. Dieses wird durch die graphische Strukturierung der Funktionsblöcke (Systemdokumentation), sowie der automatischen Dokumentationsmöglichkeiten erreicht.
Die Funktionsentwicklung kann sich weitestgehend auf den Entwurf der reinen Funktionalität konzentrieren und benötigt nicht mehr das detaillierte Wissen über die Zielhardware.
Als Ergebnis wird eine im hohen Maße wiederverwendbare, weil auch getestete Funktionssoftware zur Verfügung stehen. Durch die ebenfalls stark modularisierte, mit schmallen Schnittstellen versehene SW-Plattform

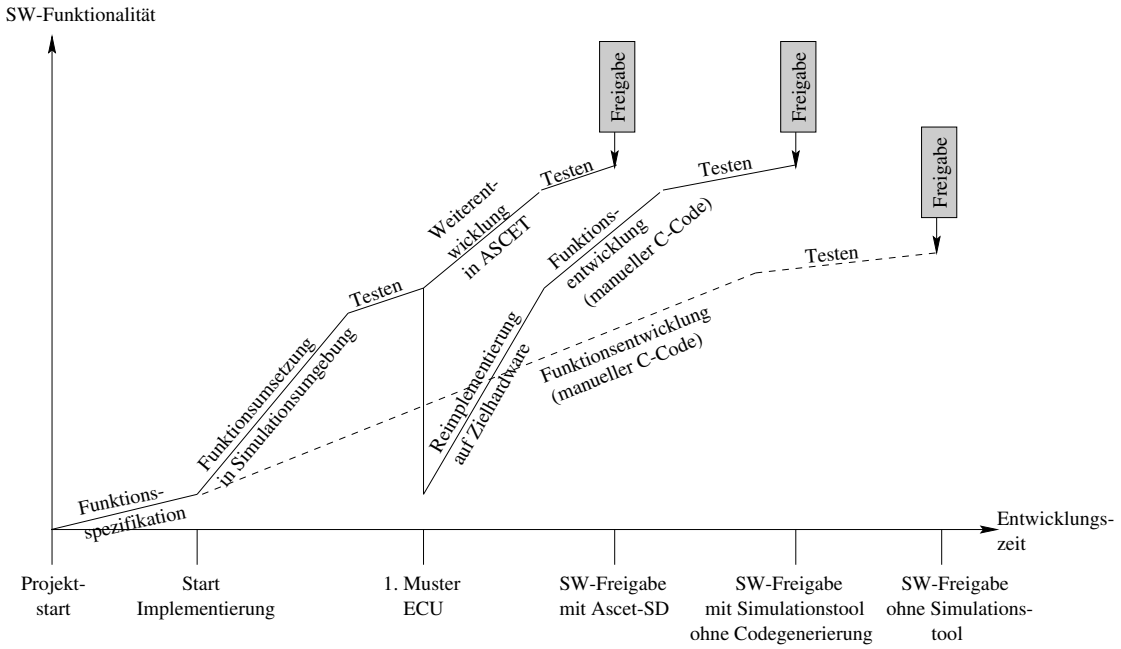

Abbildung 5: SW-Entwicklung mit und ohne ASCET-SD.

wird deren Wiederverwendbarkeit, bzw. schnelle Portierbarkeit auf andere Zielprozessoren ebenfalls gewährleistet.
Unter Ausnutzung der bei der ESP-Entwicklung gesammelten Erfahrungen muss sich die Leistungsfähigkeit des gesamten Entwicklungsprozesses zur Zeit im Rahmen unserer Entwicklung eines neuen EBS-Systems auf einer anderen Zielhardware beweisen.