

Otto-von-Guericke-Universität Magdeburg



Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme

Diplomarbeit

titel

Verfasser:

Vorname Nachname

dd.mm.yyyy

Betreuer:

Prof. xxx,

xxx,

xxx

Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D-39016 Magdeburg

Nachname, Vorname:

title

Diplomarbeit, Otto-von-Guericke-Universität
Magdeburg, 2008.

Zusammenfassung

Inhaltsverzeichnis

Abbildungsverzeichnis	v
Tabellenverzeichnis	vii
Abkürzungsverzeichnis	ix
1 Einleitung	1
1.1 Motivation	1
1.2 Zielstellung	2
1.3 Gliederung der Arbeit	3
2 Grundlagen	5
2.1 Eingebettete Systeme	5
2.2 Atmel AVR Hardware-Produktlinie	7
2.3 Programmierung eingebetteter Systeme	7
2.3.1 Assembler	8
Literaturverzeichnis	11

Abbildungsverzeichnis

2.1	Wirkungskette System/Umgebung [Sch05]	6
2.2	Zwei Varianten der eingesetzten Hardware-Produktlinie	8

Tabellenverzeichnis

2.1	Programmiersprachen und deren Eignung zum Einsatz in Software-Projekten [VDI94]	9
-----	---	---

Abkürzungsverzeichnis

AOP	Aspekt-Orientierte Programmierung
CAN	Controller Area Network
DBMS	Datenbank Management System
FOP	Feature-Orientierte Programmierung
HPL	Hardware-Produktlinie
OOP	Objekt-Orientierte Programmierung
SPL	Software-Produktlinie
USART	Universal Synchronous and Asynchronous Receiver Transmitter
VDI	Verein Deutscher Ingenieure

KAPITEL 1

Einleitung

1.1 Motivation

Mehr als 98 % der im Jahr 2000 weltweit produzierten Prozessoren waren Bestandteil eingebetteter Systeme [Ten00]. Der hohe Prozentsatz ist ein Anzeichen für die große Vielfalt eingebetteter Systeme, die in nahezu jedem technischen Gerät zu finden sind. Ihre Anwendungsgebiete liegen in den Bereichen Unterhaltungs- und Haushaltselektronik, Telekommunikation, Automotive, Industrieautomatisierung u. a. Jährliche Wachstumsraten im zweistelligen Bereich sorgten bis heute dafür, dass sich der Anteil an Prozessoren in eingebetteten Systemen gleichbleibend auf dem Niveau des Jahres 2000 hielt [Tur02, Kri05, Sch05, Tur08]. Weitergehend zeigen entsprechende Veröffentlichungen zu diesem Thema, dass eingebettete Systeme in Zukunft wahrscheinlich maßgeblich zur Wertschöpfung von Produkten beitragen [BKR⁺05, HKM⁺05].

*Stellenwert
eingebette-
ter
Systeme*

Obwohl die Kosten für Hardware-Komponenten gering ausfallen, führt der vermehrte Einsatz eingebetteter Systeme zu einem erheblichen Anteil von ca. 50 % an den Produktionskosten in den zuvor genannten Anwendungsgebieten [HKM⁺05]. Einen entscheidenden Beitrag leistet hierbei die Software der Systeme. Laut [BHK02] und [Kri05] nimmt ihr Umfang und ihre Komplexität fortwährend zu. Der Grund liegt in den zusätzlichen Anforderungen, die eingebettete Systeme mittlerweile erfüllen müssen und deren Umsetzung vermehrt in Software erfolgt. Bestehende Ressourcenbeschränkungen der eingesetzten Hardware führen in diesem Zusammenhang häufig zu aufwendigen Neuentwicklungen oder Anpassungen, die die Kosten zusätzlich steigern.

*Probleme
Software-
Entwicklung*

Aus der Warenproduktion ist seit längerem die Entwicklungsform der *Hardware-Produktlinie (HPL)* bekannt. Anhand standardisierter Bausteine können individuelle Ausprägungen je nach Kundenwunsch gefertigt werden. Ein Beispiel der erfolgreichen Umsetzung einer HPL sind die 8-Bit Mikroprozessoren (AVR) für eingebettete Systeme des Herstellers Atmel.¹ Die einzelnen Modelle dieser Produktlinie unterscheiden sich in der Ausstattung an Programm- und Arbeitsspeicher, IO-Anschlüssen und integrierten Komponenten.

*Lösungsansatz
Produktli-
nie*

Mit Programmfamilien und *Software-Produktlinie (SPL)* existieren zwei Konzepte, die die Methodik der Produktlinientechnologie auch in der Software-Entwicklung umsetzen. Ziel dieser Vorgehensweisen ist die Modularisierung und Wiederverwendung von Software-Komponenten, für deren Umsetzung unterschiedliche Programmierparadigmen existieren.

¹ <http://www.atmel.com/products/avr/>

OOP Innerhalb der Anwendungsprogrammierung für Personalcomputer ist *Objekt-Orientierte Programmierung (OOP)* zur Strukturierung von Aufgabenstellungen weit verbreitet. Wie [EAK⁺01, Pre97, TOHSMS99] zeigen, reicht OOP zur Modularisierung von Software nicht aus. Unterschiedliche Sichten und Anforderungen lassen sich durch einfache Dekompositionen, wie sie mit OOP möglich sind, nicht abbilden. Zwei Techniken die diese Problematik adressieren sind *Aspekt-Orientierte Programmierung (AOP)* und *Feature-Orientierte Programmierung (FOP)*. Beide Programmierparadigmen erlauben eine über die OOP hinausgehende Wiederverwendbarkeit von Software-Modulen.

Vorbehalte gegenüber modernen Programmierparadigmen

Die Anwendung moderner Programmierparadigmen zur Umsetzung von Software-Projekten für eingebettete Systeme wird in der Literatur kaum thematisiert. Dabei könnte die Software-Entwicklung eingebetteter Systeme von den Fortschritten in der Anwendungsprogrammierung profitieren. Aus Performancegründen wird häufig auf den Einsatz moderner Programmierparadigmen verzichtet. Vielfach wird diese Entscheidung mit dem Verweis auf Overhead (Programmgröße und -laufzeit) begründet, der im Zusammenhang mit neuen Software-Techniken stehen soll. Dies bezieht sich in erster Linie noch auf den Einsatz von OOP innerhalb von Software-Projekten.

Dennoch gibt es bereits einige Projekte in denen moderne Programmierparadigmen erfolgreich eingesetzt werden. In [SPLSS07] wird die Entwicklung einer SPL für Betriebssysteme auf der Basis von AOP beschrieben (Projekt CiAO).² Der familienbasierte Ansatz von CiAO führt zur Kapselung verschiedener Betriebssystemfunktionalitäten, wie z. B. Strategien zur Verarbeitung von Threads oder Interruptfunktionalität.

In [LSSP06] wird die Entwicklung einer eingebetteten Wetterstation mit Hilfe verschiedener Implementierungen — darunter C, C++ und AspectC++ — untersucht. Alle Implementierungen folgen dem Ansatz der Software-Produktlinienentwicklung mit dem Fokus auf den eingesetzten Hardware-Komponenten (HPL). Die Ergebnisse zeigen, dass der entstehende Overhead der AspectC++-Implementierung im Vergleich zu einer effizienten C-Implementierung weniger als 3 % ausmacht. Darüber hinaus konnten leichte Geschwindigkeitsvorteile bei der Umsetzung mittels AspectC++ im Vergleich zu den anderen Implementierungen festgestellt werden.

1.2 Zielstellung

Ziel dieser Arbeit ist es Empfehlungen für den Einsatz moderner Programmierparadigmen für die Umsetzung von Software-Projekten auf tief eingebetteten Systemen zu geben. Im Feld eingebetteter Systeme zeichnen sich tief eingebettete Systeme besonders durch ihre geringe Leistungsfähigkeit und Speicherausstattung aus. Neben den Empfehlungen werden zudem Interaktionspunkte zwischen HPL und SPL aufgezeigt.

Die Untersuchungen des Einsatzes erweiterter Programmierparadigmen erfolgt

² <http://www4.informatik.uni-erlangen.de/z/Research/CiAO/>

in dieser Arbeit an zwei Beispielen: Entwicklung von Gerätetreibern und Realisierung eines *Datenbank Management System (DBMS)* für die HPL eines tief eingebetteten Systems. Der Schwerpunkt wird auf die Umsetzung verschiedener Konfigurationen und Varianten mit Rücksicht auf das Zielsystem gelegt. Zur Realisierung werden die beiden Programmierparadigmen AOP und FOP herangezogen. Die Einschränkung auf diese beiden Paradigmen trägt dem Umstand Rechnung, dass in der vorliegenden Arbeit keine vollständige Untersuchung auf dem Gebiet moderner Programmierparadigmen erfolgen kann.³ Die konkrete Untersuchung erfolgt für AOP und FOP anhand der beiden Erweiterungen AspectC++⁴ und FeatureC++⁵ für C++.

1.3 Gliederung der Arbeit

Alle für das Verständnis dieser Arbeit notwendigen Grundlagen werden in Kapitel 2 vermittelt. Das schließt eingebettete Systeme und deren Programmierung sowie Fachwissen aus dem Bereich moderner Programmierparadigmen ein. In Kapitel 3 folgt eine Analyse zum Einsatz des Präprozessors `cpp` bei der Umsetzung von Software-Projekten für tief eingebettete Systeme. Kapitel 4 greift die Ergebnisse des vorhergehenden Kapitels auf und diskutiert den Einsatz erweiterter Programmierparadigmen am Beispiel der Entwicklung von Gerätetreibern. Im Anschluss folgt eine Domänenanalyse für Datenmanagement auf tief eingebetteten Systemen. In diesem Zusammenhang werden existierende DBMS vorgestellt und ihre Eignung für den Einsatz auf tief eingebetteten Systemen analysiert. In Kapitel 6 wird der Aufbau und die Implementierung eines DBMS für tief eingebettete Systeme diskutiert. Einen Schwerpunkt dabei bildet die Bewertung der eingesetzten Paradigmen. Kapitel 7 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf weitere Arbeiten.

³ Weitere Paradigmen sind Intentionale Programmierung, Subjekt-Orientierte Programmierung, Generative Programmierung u. a.

⁴ <http://www.aspectc.org/>

⁵ http://wwwiti.cs.uni-magdeburg.de/iti_db/forschung/fop/featurec/

KAPITEL 2

Grundlagen

Mit dem Mikroprozessor Modell 4004 von Intel aus dem Jahr 1971 kam die Entwicklung von eingebetteten Systemen auf [Bar99]. Die zum damaligen Zeitpunkt vorherrschenden Ressourcenbeschränkungen prägen in ähnlicher Form bis heute den Einsatz und die Programmierung dieser Systeme. Im folgenden Kapitel wird u. a. ein Einblick in die Thematik eingebetteter Systeme gegeben. Dazu gehört neben der Vorstellung der für die Arbeit eingesetzten eingebetteten Systeme auch der aktuelle Stand der Programmierung.

Auf die Erläuterungen zur Programmierung eingebetteter Systeme folgt eine Einführung in die Methode der Domänenanalyse am Beispiel der *Feature-Orientierte Domänenanalyse (FODA)*. Das geschilderte Vorgehen wird später zur Bestimmung der Anforderungen von Datenmanagementsystemen für tief eingebettete Systeme verwendet (Kapitel ??, S. ??). Die Ergebnisse der Analyse bilden die Grundlage zur Entwicklung der *Software-Produktlinie (SPL)* des Datenmanagementsystems *RobbyDBMS*. Zum besseren Verständnis wird der Begriff SPL im Abschnitt ?? erörtert. Zwei Konzepte zu deren Umsetzung werden mit den Programmierparadigmen AOP und FOP im Anschluss vorgestellt. Mit AspectC++ und FeatureC++ wird jeweils eine Programmiersprache als Vertreter eines Paradigmas eingeführt.

2.1 Eingebettete Systeme

Eingebettete Systeme werden nach [Sch05] wie folgt definiert:

Eingebettete Systeme sind Computersysteme, die aus Hardware und Software bestehen und die in komplexe technische Umgebungen eingebettet sind.

Tief eingebettete Systeme zeichnen sich darüber hinaus durch extreme Ressourcenbeschränkungen aus. Einschränkungen beziehen sich dabei auf die Leistungsfähigkeit des Mikroprozessors, den Speicher und den Stromverbrauch [SSPSS98]. In der Praxis bedeutet dies, dass der Prozessor mit einer geringen Taktfrequenz arbeitet und die Speicherausstattung bei nur wenigen Kilobyte liegt. Die Begrenzung auf minimale Ressourcen ist eine Folge von Kostenvorteilen, die sich bei sehr hohen Stückzahlen erzielen lassen. Gleichzeitig führt der Kostendruck zur Entkopplung der Entwicklung tief eingebetteter Systeme von der Speicher- und Mikroprozessorentwicklung im Computerbereich [ABP06]. In Abschnitt 2.2 (S. 7)

tief eingebettete Systeme

werden die für die Realisierung dieser Arbeit eingesetzten Systeme als Beispiel tief eingebetteter Systeme vorgestellt.

*schema-
tische
Arbeitswei-
se* Eingebettete Systeme finden sich in vielen Anwendungsbereichen wieder und müssen unterschiedliche Anforderungen erledigen. Grundlegend erfüllen sie Steuerungs- und Regelungsaufgaben z. B. in Kraftfahrzeugen, Konsumelektronik. Die Abbildung 2.1 stellt schematisch ihre Arbeitsweise dar.

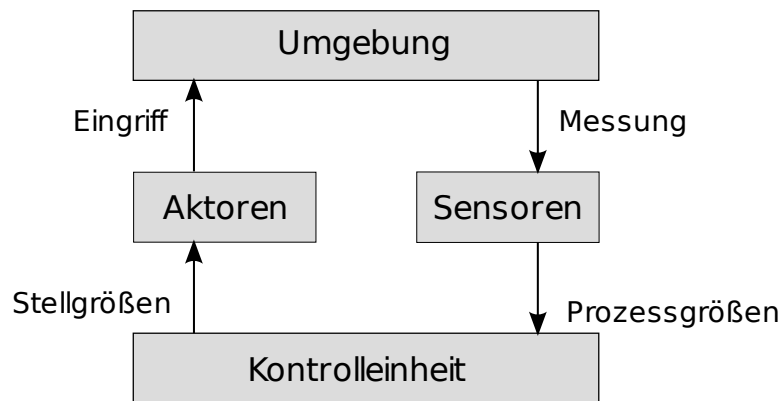


Abbildung 2.1: Wirkungskette System/Umgebung [Sch05]

Die Darstellung zeigt eingebettete Systeme als Sensor-Aktor Systeme. Zur Abarbeitung von Aufgabenstellungen stehen verschiedene Sensoren zur Verfügung, z. B. Druck, Temperatur, Beschleunigung. Anhand der Eingabewerte dieser Sensoren wird die Umgebung modellhaft erfasst und in der Kontrolleinheit verarbeitet. Diese generiert daraus Stellgrößen für Aktuatoren zur Rückwirkung auf die Umgebung. Je nach Nutzung oder Aufgabenbereich sind eingebettete Systeme noch mit weiteren Funktionen ausgestattet: Modul für drahtlose Kommunikation, Bussystem für drahtgebundene Übertragungen, LCD-Anzeige u. a.

Für verschiedene Aufgabenstellungen ist es notwendig, die mit den Sensoren gemessenen Daten für einen längeren Zeitraum vorzuhalten. Ein paar Beispiele für anfallende Daten sind: aufgetretene Fehler- oder Systemzustände, Konfigurationsdaten und Analysedaten, die während der Systementwicklung anfallen. Der Einsatz von Datenmanagementsystemen für die Datenhaltung ist bereits auf dieser Ebene sinnvoll. Eine genauere Erläuterung dazu erfolgt in den Abschnitten ?? und ?? (S. ?? bzw. ??).

Die Einbettung in reale Systeme und die damit verbundene Interaktion mit der Umwelt erfordert ein hohes Maß an Zuverlässigkeit. Im Einzelnen gehören dazu (Attribute der Zuverlässigkeit [Sch79]): Verfügbarkeit, Betriebsbereitschaft, Betriebssicherheit, Wartungsfreundlichkeit, Vertraulichkeit, Integrität und Ausfallsicherheit. Diese Attribute vergrößern die Anzahl der Anforderungen an ein eingebettetes System. Da nicht alle Attribute aufgrund von Zielkonflikten gleichermaßen erfüllt werden können, sind Konfigurationen zur Festlegung des Funktionsumfangs eines Zielsystems erforderlich.

*Hardware-
Produktlinie* Die in technischen Produkten weit verbreiteten Produktlinien kommen auch hier zum Einsatz. Eine Reihe von eingebetteten Systemen unterscheidet sich mit-

unter nur in einzelnen Merkmalen oder in der Ausstattung vorhandener Komponenten. Dazu gehören die zuvor genannten Sensoren, aber auch die Leistungsfähigkeit und Größe des Arbeits-, Daten- und Programmspeichers. Beispiele verschiedener eingebetteter Systeme werden nachfolgend vorgestellt.

2.2 Atmel AVR Hardware-Produktlinie

In dieser Arbeit werden verschiedene eingebettete Systeme auf der Basis der Prozessorfamilie AVR der Firma Atmel eingesetzt. Es handelt sich dabei um 8-Bit Prozessoren mit integriertem Arbeits-, Daten- und Programmspeicher. Die Prozessoren selbst bilden eine Produktlinie. Durch vorhandene Komponentenausstattungen und optionale Schnittstellen entstehen verschiedene Variationen. Zu den grundlegenden Ausstattungsmerkmalen gehören:

- 1 - 8 MHz Taktrate des Prozessors,¹
- 32 oder 128 KB Programmspeicher,
- 1 oder 4 KB Arbeitsspeicher und
- 1 oder 4 KB Datenspeicher (EEPROM).

Optionale Schnittstellen sind bspw. *Controller Area Network (CAN)* und *Universal Synchronous and Asynchronous Receiver Transmitter (USART)*. Eingebettete Systeme, deren Stromversorgung auf Batterien oder Akkumulatoren basiert, werden zusätzlich noch als autonom bezeichnet.

Auf der Basis der Prozessorfamilie AVR lassen sich unterschiedliche eingebettete Systeme kreieren. Zur Anzahl unterschiedlicher Varianten (Hardware-Konfigurationen) in der Produktlinie tragen vor allem externe Erweiterungen wie z. B. Sensoren bei. Darüber hinaus lassen sich weitere Varianten durch ein LCD-Display und Motoren aufbauen. Zwei Beispiele unterschiedlicher autonomer eingebetteter Systeme zeigt die Abbildung 2.2.

Produktlinie
eingebettetes
System

2.3 Programmierung eingebetteter Systeme

Im folgenden Abschnitt wird ein Überblick über Sprachen und Methoden zur Programmierung eingebetteter Systeme gegeben. Dazu werden einzelne Programmiersprachen erläutert, die u. a. in der vorliegenden Arbeit eingesetzt werden.² Diese Arbeit gibt keine Einführung in die genannten Programmiersprachen, sondern stellt deren Bedeutung für die Programmierung eingebetteter Systeme heraus. Beispiele zu den vorgestellten Programmiersprachen und eine Diskussion ihres Einsatzes zur Programmierung tief eingebetteter Systeme folgen in Kapitel ?? (S. ??).

¹ Die angegebenen Taktraten gehören zur Werkskonfiguration, können jedoch angepasst werden.

² Weitere Sprachen sind Ada, Basic, Forth u. a.

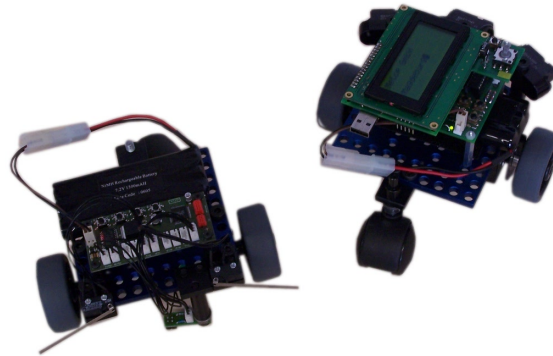


Abbildung 2.2: Zwei Varianten der eingesetzten Hardware-Produktlinie

2.3.1 Assembler

Assembler ist eine maschinennahe Programmiersprache, die sich auf eine konkrete Prozessorarchitektur bezieht. Assembler-Befehle werden häufig eins zu eins in Maschinenbefehle des Prozessors übersetzt und bilden damit die Grundlage für sehr effiziente Programme.

*Nachteile
Assembler*

Heutzutage wird häufig auf die Verwendung von Assembler bei der Programmierung von eingebetteten Systemen verzichtet. Das liegt zum einen an der Verbesserung der Compiler die Quelltext aus einer Hochsprache in Assembler transformieren und zum anderen daran, dass die Portabilität und Produktivität hinter Programmen, die in Hochsprachen geschrieben werden, zurückliegt [Bar99]. Bestätigt wird diese Einschätzung durch die VDI³-Richtlinie 2422 [VDI94] (Tabelle 2.1). In der Richtlinie wird beschrieben, dass Assembler zur Umsetzung komplexer Steuerungen ungeeignet ist. Dies wird mit dem Verweis auf die geringe Problemabstraktion begründet. Entsprechend der Angaben aus der Tabelle 2.1 eignen sich für komplexere Steuerungen nur Hochsprachen wie C, PL/M und Pascal. Dabei schafft es die Programmiersprache C als einzige unter den genannten Hochsprachen auch die Anforderung zeitkritischer Teilfunktionen zu erfüllen. Demnach kann C als Ersatz für Assembler betrachtet werden.

Obwohl die Prozessoren der AVR HPL einer Modellreihe angehören, unterscheiden sich ihre Assembler-Befehlssätze geringfügig. Je nach Modell fehlen einige Anweisungen, wodurch die Portierung von Programmen bereits innerhalb der Produktlinie erschwert wird.

³ Verein Deutscher Ingenieure (VDI)

Programmiersprache & Verwendung	einfache Steuerung	komplexe Steuerung	zeitkritische Teilfunktion
ASSEMBLER	x		x
BASIC	x		
FORTRAN	x		
C	x	x	x
PL/M	x	x	
PASCAL	x	x	

Tabelle 2.1: Programmiersprachen und deren Eignung zum Einsatz in Software-Projekten [VDI94]

Literaturverzeichnis

- [ABP06] Nicolas Anciaux, Luc Bouganim, and Philippe Pucheral. Smart Card DBMS: where are we now? Technical Report 80840, Institut National de Recherche en Informatique et Automatique (INRIA), Le Chesnay Cedex, France, Juni 2006.
- [Bar99] Michael Barr. *Programming Embedded Systems in C and C++*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.
- [BHK02] Friedrich Bollow, Matthias Homann, and Klaus-Peter Köhn. *C und C++ für Embedded Systems*. mitp Verlag, Bonn, 2002.
- [BKR⁺05] Andreas Biagosch, Stefan Knupfer, Philipp Radtke, Ulrich Näher, and Andreas E. Zielke. Automotive Electronics - Managing Innovations on the Road. Technical report, McKinsey & Company, Inc., Düsseldorf, Juli 2005.
- [EAK⁺01] Tzilla Elrad, Mehmet Aksit, Gregor Kiczales, Karl J. Lieberherr, and Harold Ossher. Discussing Aspects Of AOP. *Communications of the ACM*, 44(10):33–38, 2001.
- [HKM⁺05] Alfred Helmerich, Nora Koch, Luis Mandel, Peter Braun, Peter Dornbusch, Alexander Gruler, Patrick Keil, Roland Leisibach, Jan Romberg, Bernhard Schätz, Thomas Wold, and Guido Wimmel. Study of Worldwide Trends and R&D Programmes in Embedded Systems in View of Maximising the Impact of a Technology Platform in the Area. Technical Report FAST 2005, F.A.S.T. Gesellschaft für angewandte Softwaretechnologie mbH, Technische Universität München, München, November 2005.
- [Kri05] Ravi Krishnan. Future of Embedded Systems Technology. Technical Report GB-IFT016B, BCC Research, Wellesley, MA, USA, Juni 2005.
- [LSSP06] Daniel Lohmann, Olaf Spinczyk, and Wolfgang Schröder-Preikschat. Lean and Efficient System Software Product Lines: Where Aspects Beat Objects. In *Transactions on Aspect-Oriented Software Development II (TAOSD)*, volume 4242 of *Lecture Notes in Computer Science*, pages 227–255. Springer-Verlag, 2006.
- [Pre97] Christian Prehofer. Feature-Oriented Programming: A Fresh Look at Objects. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, volume 1241 of *Lecture Notes in Computer Science*, pages 419–443. Springer-Verlag, 1997.

- [Sch79] Eugen Schaefer. *Zuverlässigkeit, Verfügbarkeit und Sicherheit in der Elektronik : eine Brücke von der Zuverlässigkeitstheorie zu den Aufgaben der Zuverlässigkeitspraxis*. Vogel Verlag, Würzburg, 1979.
- [Sch05] Peter Scholz. *Softwareentwicklung eingebetteter Systeme*. Springer-Verlag, Berlin, 2005.
- [SPLSS07] Wolfgang Schröder-Preikschat, Daniel Lohmann, Fabian Scheler, and Olaf Spinczyk. Dimensions of Variability in Embedded Operating Systems. *Informatik - Forschung und Entwicklung*, 22(1):5–22, 2007.
- [SSPSS98] Friedrich Schön, Wolfgang Schröder-Preikschat, Olaf Spinczyk, and Ute Spinczyk. Design Rationale of the PURE Object-Oriented Embedded Operation System. In *Proceedings of the IFIP International Workshop on Distributed and Parallel Embedded Systems (DIPES)*, pages 231–240. Kluwer Academic Publishers, 1998.
- [Ten00] David Tennenhouse. Proactive Computing. *Communications of the ACM*, 43(5):43–50, 2000.
- [TOHSMS99] Peri Tarr, Harold Ossher, William Harrison, and Jr. Stanley M. Sutton. N degrees of separation: multi-dimensional separation of concerns. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 107–119. IEEE Computer Society, 1999.
- [Tur02] Jim Turley. *The Essential Guide to Semiconductor Technology*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2002.
- [Tur08] Jim Turley. Intel Has Just 2% Market Share!, 2008. <http://www.jimturley.com/articles.htm#2percent>; [Online; Stand 29.08.2008].
- [VDI94] Entwicklungsmethodik f \ddot{A} $\frac{1}{4}$ r Geräte mit Steuerung durch Mikroelektronik, Februar 1994. VDI/VDE Richtlinie 2422.

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, 10. März 2009

Vorname Nachname