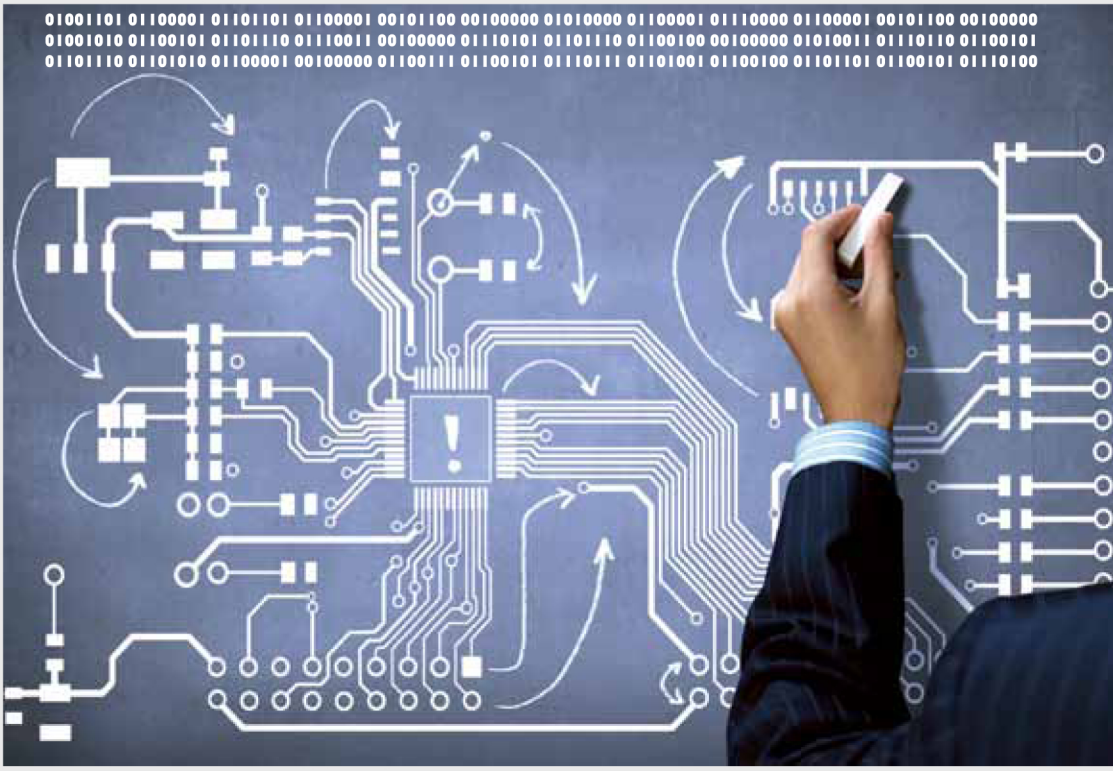
Steffen Jaschke

Informatikdidaktische Diskussion über das Design eingebetteter Systeme



universi
UNIVERSITÄTSVERLAG SIEGEN

Informatikdidaktische Diskussion über das Design eingebetteter Systeme

GENEHMIGTE DISSERTATION

zur Erlangung des Grades eines Doktors
der Naturwissenschaften

vorgelegt von
Dipl.-Inform. Steffen Jaschke

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen

Steffen Jaschke
Universität Siegen
Naturwissenschaftlich-Technische Fakultät
Department Elektrotechnik-Informatik
Breite Straße 11
D-57076 Siegen
steffen.jaschke@uni-siegen.de

Genehmigte Dissertation
Erste Gutachterin: Prof. Dr. Sigrid Schubert
Zweiter Gutachter: Prof. Dr. Rainer Brück
Tag der mündlichen Prüfung: 20.02.2015

ISBN: 978-3-936533-57-6

universi – Universitätsverlag Siegen
Am Eichenhang 50
D-57076 Siegen
info@universi.uni-siegen.de
www.uni-siegen.de/universi

Kurzfassung

Die Ausbildung künftiger Entwickler eingebetteter Systeme ist heute geprägt von einer subjektiven, kulturspezifischen Gestaltung von Lehr-Lernprozessen, welche die Ergebnisse der Kompetenzforschung zumeist nicht berücksichtigen. Es besteht Konsens, dass in Kompetenzmodellen strukturierte Kompetenzen – kognitive Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen – notwendig sind, um zwischen abstrakten Bildungszielen und konkreten Lehr-Lernprozessen zu vermitteln. Damit stellen die Erforschung von Kompetenzmodellen und Pfaden der Kompetenzaneignung grundlegende Forschungsbedarfe zur Hochschuldidaktik der technischen Informatik dar. In dieser Arbeit werden Konzepte zur theoretischen Fundierung von Laborpraktika der technischen Informatik entwickelt, welche auf Ergebnissen des von der *Deutschen Forschungsgemeinschaft* geförderten Projektes *Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen (KOMINA)* aufbauen.

In der vorliegenden Arbeit wird das Verständnis von eingebetteten Systemen als Teil informatischer Curricula insofern erweitert, als dass sich diese Systeme als Lerngegenstand eignen, um Kompetenzen verschiedener Informatikdisziplinen zu fördern und damit nicht auf die technische Informatik beschränkt sind. Eine Taxonomie zur Vergleichbarkeit fachdidaktischer Publikationen zu eingebetteten Systemen wird weiterentwickelt und angewandt, um institutionelle Besonderheiten sowie die Vielseitigkeit des Praxisfeldes zu erfassen. Forschungsgegenstand ist das im Rahmen von KOMINA entwickelte Entwurfs- und Anwendungspraktikum für eingebettete Systeme. Zielgruppe sind Studierende der Informatik. Es werden typische Lernhürden identifiziert, wodurch neue Erkenntnisse, über die formative Evaluation des unter Beteiligung des Autors entwickelten und durchgeführten Praktikums hinaus, gewonnen werden. Diese Erkenntnisse begründen die Notwendigkeit neuer didaktischer Konzepte und lernförderlicher Software unter Berücksichtigung institutioneller Besonderheiten sowie zielgruppenspezifischer Vorkenntnisse.

Kognitive Strukturen als Komponente didaktischer Systeme werden in diesem Forschungsprojekt erforscht. Sie dienen als Basis für die informatikdidaktische Verfeinerung des in KOMINA empirisch evaluierten Kompetenzstrukturmodells für das Entwickeln eingebetteter Mikro- und Nanosysteme. Bislang wurden Erarbeitungsreihenfolgen informatorischer Fachkonzepte in didaktischen Systemen betrachtet, welche drei Funktionen besitzen. Die Orientierung der Lernenden im Fachgebiet, die Organisation zur Planung von Lehr-Lernprozessen sowie die Diskussion didaktischer Entscheidungen. In diesem Beitrag zur Grundlagenforschung zur Hochschuldidaktik der technischen Informatik steht die Diskussion didaktischer Entscheidungen bei der Gestaltung von Lehr-Lernprozessen und Pfaden der Kompetenzaneignung im Vordergrund. Deshalb werden die Anforderungen an die Darstellung kognitiver Strukturen – Ausdrucksstärke, Übersichtlichkeit und Nachvollziehbarkeit – zugunsten der Diskussion didaktischer Entscheidungen angepasst.

Der Autor stellt Forschern zur Hochschuldidaktik der technischen Informatik Konzepte bereit, die es ermöglichen Fachkonzepte und Lehr-Lernprozesse zu analysieren sowie durch Anpassung an institutionelle Besonderheiten theoretisch fundiert zu gestalten. Dies sind insbesondere eine Taxonomie zur Identifikation von Lernhürden, die Methodik zur Ausdifferenzierung von Kompetenzen mit Bezug zu den identifizierten Lernhürden, die Visualisierung kognitiver Strukturen mit diesen Kompetenzen im Zentrum sowie die in der Hauptverantwortung des Autors entwickelte lernunterstützende Software *Explorative Learning and Visualization Environment*. Diese dient als Beispiel für den Einsatz von Simulationen in Laborpraktika der technischen Informatik. Es wird damit exemplarisch gezeigt, wie die informatikdidaktische Verfeinerung des Kompetenzstrukturmodells in Verbindung mit kognitiven Strukturen und lernförderlicher Software zur Überwindung der mithilfe der entwickelten Taxonomie identifizierten Lernhürden eingesetzt werden können.

Abstract

Today, the education of future developers of embedded systems is characterized by a subjective, culture-specific design of teaching and learning processes. This design, mostly, does not take the results of research on competences into account. There is a consensus that competences – cognitive abilities and skills used to solve specific problems – and competence models are needed to mediate between abstract and concrete educational goals of teaching and learning processes. Therewith, the exploration of competence models and paths of competence acquisition are fundamental research needs for didactics of computer engineering at university. Within this work, based on the results of the project *competence development with embedded micro- and nanosystems (KOMINA)* funded by the *German Research Foundation*, approaches for a theoretical foundation of laboratory courses of computer engineering have been developed.

Within this thesis, the understanding of embedded systems as a part of computer science curricula has been broadened since it is not limited to computer engineering. While these systems are also suitable as learning objects to promote competences within various computer science disciplines. A taxonomy to foster the comparability of research on didactics of computer engineering is further developed and applied in order to conceive institutional particularities and the versatility of the practice field. The *design and application laboratory for embedded systems*, developed in the context of KOMINA, is the object of research. Students of computer science are the target group. In addition to the formative evaluation of the developed laboratory the taxonomy enables the identification of typical learning barriers. These findings justify the need for new educational concepts as well as learning software, taking institutional particularities and target group specific knowledge into account.

Cognitive structures as a component of Didactic Systems are investigated. As they serve as a basis for the refinement of the empirically evaluated competence structure model for the development of embedded micro- and nanosystems. So far, the sequences of teaching units have been considered in Didactic Systems, which posses three functions. The orientation of the learner in the topic, the organization of the planning of teaching and learning processes as well as the discussion of didactic decisions. In this work, regarding the contribution to basic research on didactics of computer engineering at university, the discussion of didactic decisions in the design of educational processes and paths of competence acquisition has priority. Therefore, the demands on the representation of cognitive structures – expressiveness, clarity and comprehensibility – are adjusted in favor of the discussion of didactic decisions.

The author provides concepts which allow researchers to analyze technical concepts as well as to develop educational processes in a theoretically established manner. These are, in particular, a taxonomy to identify learning barriers, the

methodology for the differentiation of competences related to the identified learning hurdles, and the visualization of cognitive structures with these competences at center. Additionally, the learning software *Exploratory Learning and Visualization Environment*, that has been developed with the author's responsibility, is presented. This software serves as an example for the use of simulations in laboratory courses of computer engineering. Thus, the combination of the refinement of the competence structure model in conjunction with cognitive structures in addition to learning software, which can be applied to overcome the identified learning hurdles, is shown as an example.

Vorwort

Die vorliegende Dissertationsschrift entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter im Projekt „Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen (KOMINA)“, welches von der Deutschen Forschungsgemeinschaft gefördert wurde. Eine ganze Reihe von Personen, die ich hier nicht alle namentlich nennen kann, haben dazu beigetragen, dass ich die Promotion erfolgreich abschließen konnte.

Besonderer Dank geht an Frau Prof. Dr. Sigrid Schubert für die Unterstützung und Förderung während meiner Dienstzeit am Lehrstuhl für „Didaktik der Informatik und E-Learning“ sowie Herrn Prof. Dr. Rainer Brück für die Übernahme des Koreferats und die hilfreichen Anmerkungen, die zu einer Präzisierung der Arbeit beigetragen haben.

Des Weiteren bedanke ich mich bei meinem Bürokollegen Steffen Büchner für die zahlreichen fachlich konstruktiven Diskussionen und die sehr angenehme Zeit am Lehrstuhl. Herrn Gerd Müller danke ich, wie auch dem weiteren Lehrstuhlteam, für die technische Unterstützung und motivierenden Gespräche. Herrn André Schäfer danke ich für die gute Zusammenarbeit im Projekt KOMINA. Auch danken möchte ich den betreuten studentischen Projektgruppen und Hilfskräften, die Teile der Implementierungen übernahmen, sowie den Probanden im untersuchten Laborpraktikum. Herrn Prof. Dr. Ralph Dreher und dem ganzen Team danke ich für die Freiräume am Lehrstuhl „Technikdidaktik am Berufskolleg“.

Mein ganz besonderer Dank gilt meiner Familie für die ausdauernde, über das Lektorat hinausgehende Unterstützung in den vergangenen Jahren; speziell meiner Frau Svenja für viel Geduld, Empathie, Toleranz und Freiräume; meinen Eltern Bernd und Brigitte für die Wegbereitung und andauernde Motivation; meinem Bruder Jens für seine kritischen Anmerkungen, die zu einem stärkeren Bewusstsein für den Forschungsbedarf zur Hochschuldidaktik der technischen Informatik beitrugen – AWG.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACM	Association for Computing Machinery
ADC	Analog-Digital Converter
AIS	Association for Information Systems
ASIC	Application-specific integrated circuit
ARTIST	Advanced Real-Time Systems Education Group
BMBF	Bundesministerium für Bildung und Forschung
CS	Computer Science
CE	Computer Engineering
CPS	Cyber-physische Systeme
DFG	Deutsche Forschungsgemeinschaft
DMA	Direct Memory Access
DQR	Deutscher Qualifikationsrahmen für lebenslanges Lernen
EAP	Entwurfs- und Anwendungspraktium für EMNS
EKSM	Empirisch verfeinertes Kompetenzstrukturmodell
ELVE	Explorative Learning and Visualization Environment
EMNS	Eingebettetes Mikro- und Nanosystem
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GI	Gesellschaft für Informatik
HaPra	Hardwarepraktikum
HdTI	Hochschuldidaktik der technischen Informatik
HTTP	Hypertext Transport Protocol
IEEE	Institute of Electrical and Electronics Engineers
IFIP	International Federation for Information Processing
JTAG	Joint Test Action Group
KMK	Kultusministerkonferenz
KOMINA	Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen
LED	Leuchtdiode
MoKoM	Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lernprozessen für Modellierung und Systemverständnis in der Informatik
NKSM	Normatives Kompetenzstrukturmodell
OECD	Organisation for Economic Co-operation and Development
SDdI	Systemorientierte Didaktik der Informatik
TI	Technische Informatik
UART	Universal Asynchronous Receiver Transmitter
VHDL	Very High Speed Integrated Circuit Hardware Description Language

1. Einleitung

> „Embedded-Technologien sind zentraler Baustein wichtiger Industriezweige, in denen Deutschland weltweit eine führende Position einnimmt – etwa im Automobilbau, in der Automatisierungstechnik, im Maschinen- und Anlagenbau oder in der Umwelt- und Energietechnik. Eingebettete Systeme sind zudem eine Basistechnologie zur Bewältigung großer gesellschaftspolitischer Herausforderungen des 21. Jahrhunderts, wie die demographische Entwicklung, die Sicherung von Mobilität und Energieversorgung oder die Steigerung der Ressourceneffizienz“ [BITKOM, 2010, S. 3].

1.1. Motivation

Die Relevanz eingebetteter Systeme für die Gesellschaft, Wirtschaft und Forschung ist unbestritten und Deutschland – in weltweit führender Position – ist auf gut ausgebildete Fachkräfte in Forschung und Entwicklung, insbesondere in den Querschnittstechnologien, angewiesen. Trotz dieser Erkenntnis existiert ein Forschungsmangel zur kompetenzorientierten Hochschuldidaktik der technischen Informatik, welche eine adäquate Ausbildung künftiger Entwickler[1] eingebetteter Systeme unterstützt.

Forschungsarbeiten zur Didaktik der Informatik zeigen, dass mithilfe didaktischer Modelle, theoretisch fundierter Lehr-Lernkonzepte einschließlich lernförderlicher Software und Kompetenzmodellierung/-messung die informatorische Bildung vergleichbar wird und damit einem Verbesserungsprozess zugeführt werden kann. Qualitätssichernde und vielmehr -verbessernde Maßnahmen zur hochschulischen Ausbildung von Informatikern erfordern eine Orientierung an den Ergebnissen einer informatikdidaktischen Grundlagenforschung. Dazu gehört auch die Strukturierung der Studiengänge auf curricularer Ebene, was eine Vorstellung über die Struktur von Kompetenzen und die Möglichkeiten der Förderung von Fähigkeiten und Fertigkeiten in Lehr-Lernprozessen, wie Vorlesungen, Übungen, Projekten und Laborpraktika, einschließt. Über qualitätsverbessernde Maßnahmen hinaus, ist die Anzahl der Studienabsolventen zu erhöhen, was unter anderem einhergeht mit der Steigerung der Attraktivität von Studiengängen, Modulen und Lehrveranstaltungen, um einen Studienabbruch zu vermeiden.

[1] Alle Personenbezeichnungen gelten gleichermaßen für die männliche und weibliche Form.

Eine Anpassung universitärer Curricula an theoretisch fundierte Erkenntnisse zur Kompetenzforschung ist weiterhin notwendig, um der Volatilität des Fachgebietes Rechnung zu tragen und von subjektiv gestalteten Lehr-Lernprozessen zu intersubjektiv vergleichbaren Pfaden der Kompetenzförderung zu gelangen, die über die Inhaltsebene, also das zu erlangende Wissen, hinausgehen. Empfehlungen zu Curricula der Gesellschaft für Informatik (GI), des Institute of Electrical and Electronics Engineers (IEEE), der Association for Computing Machinery (ACM) und der Advanced Real-Time Systems Education Group (ARTIST) beschreiben Fachkonzepte sowie Technologien der technischen Informatik ohne die Verknüpfungen zwischen den Wissens- und Anwendungskomponenten zu explizieren. Lehrbücher zur Einführung in die Entwicklung von eingebetteten Systemen beschränken sich oft auf relativ niedrige kognitive Stufen, wie der ausschließlichen Programmierung von Mikrocontrollersteuerungen, welche allein nicht zur ganzheitlichen Entwicklung dieser Systeme befähigt, aber dennoch eine Möglichkeit des Einstiegs bietet. Weitere Lehrbücher abstrahieren von der Implementierungsebene und betrachten die Systemmodellierung als universellen Einstieg, unabhängig von konkreten Technologien, Architekturen und Problemstellungen. Hier werden höhere kognitive Leistungen gefordert und gefördert. Publizierte Praxisberichte und Forschungsbeiträge sind geprägt von subjektiven, auf konkrete Lehrveranstaltungen beschränkten, Eindrücken der Dozierenden und sind immer im Kontext der institutionellen und personellen Besonderheiten zu betrachten. Eine Antwort auf die Frage nach einem generischen Zugang und Lernpfad für die Studierenden soll und kann in dieser Arbeit nicht gegeben werden. Vielmehr stellt sich die Frage, wie Kompetenzen für ein ganzheitliches Systemdesign, das die niederen Schichten von Informatiksystemen – die Systemhardware – mit einschließt, gefördert werden können.

Eines der Hauptprobleme bei der Gestaltung von Lehr-Lernprozessen und damit auch die Wahl eines geeigneten Zugangs ist, dass sich Empfehlungen und Praxisberichte zu konkreten Lehrveranstaltungen nur unzureichend an der aktuellen Kompetenzdiskussion orientieren und meist Lehrinhalte beschreiben – nicht aber die anwendbaren Kompetenzen (vgl. Abschnitt 2.1). Die Modellierung von Kompetenzen ist jedoch notwendig, um transparente Lehr-Lernprozesse zu gestalten und qualitätsverbessernde Maßnahmen ergreifen zu können. Kompetenzbeschreibungen dienen den Studierenden und Lehrenden gleichermaßen als Orientierung im Lehr-Lernprozess. Damit einhergehend ist eine Strukturierung von Vorwissensbeziehungen im Kontext der Entwicklung eingebetteter Systeme und dem interdisziplinären Umfeld vorzunehmen. Eine wissenschaftliche Arbeit zur Explikation von Kompetenzen für das Entwickeln eingebetteter Systeme und deren Vorwissensbeziehungen existiert bislang nicht.

Durch die sich verändernden Entwicklungsprozesse, bedingt durch eine steigende Systemkomplexität, sowie die industrielle gesellschaftliche Relevanz ist also eine Grundlagenforschung zur Hochschuldidaktik der Didaktik der technischen Informatik im Allgemeinen und der eingebetteten Systeme im Besonderen zur Qua-

litätssicherung der Ausbildungsprozesse künftiger Entwickler unerlässlich. Dieser Forderung wird mit dem Projekt Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen (KOMINA), gefördert von der Deutschen Forschungsgemeinschaft (DFG), entsprochen.

Durch die interdisziplinär arbeitenden Projektpartner –

- Mikrosysteme: Universität Siegen, Lehrstuhl für Mikrosystementwurf (Prof. Dr. Rainer Brück)
- Nanosysteme: Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 3, Rechnerarchitektur (Prof. Dr. Dietmar Fey)
- Informatikdidaktik: Universität Siegen, Lehrstuhl für Didaktik der Informatik und E-Learning (Prof. Dr. Sigrid Schubert)

– wird eine das Forschungsfeld umfassende Expertise geboten. Geleistet wird ein Beitrag, welcher auf der informatikdidaktischen Kompetenzforschung aufbaut, die bislang weder die Hochschulausbildung noch die technische Informatik fokussierte.

Diese Arbeit leistet einen Forschungsbeitrag zu KOMINA und der Hochschuldidaktik der technischen Informatik (HdTI). Zielgruppe in diesem Forschungsprojekt sind Studierende der Informatik und damit das Berufsfeld von Systementwicklern mit Hochschulausbildung [Schubert et al., 2010, S. 1]. Insbesondere die Strukturierung von Laborpraktika und der Einsatz von lernunterstützender Hard- und Software werden erforscht.

1.2. Forschungsziele und Forschungsverlauf

Dem beschriebenen Forschungsbedarf wird mit dem Erreichen der Forschungsziele (F1-F3) entgegnet. Als geeignete Konzepte zur Strukturierung von Lehr-Lernprozessen im Bereich Internetworking und objektorientierte Modellierung wurden didaktische Systeme erforscht, deren Komponenten als informatikdidaktische Grundlage für die Forschung zu oben genannten Problemen dienen, also einer fundierten Vorstellung über die zu erlangenden Kompetenzen, Erforschung von (alternativen) Wegen der Kompetenzförderung und zu Lernhilfen in der Lehr-Lernpraxis. Hier ist zu zeigen, inwiefern das Konzept bezüglich Zielgruppe und Fachgebiet angepasst werden muss, um zu neuen Empfehlungen zur Gestaltung von Laborpraktika zu kommen. Teilaspekte wurden bereits in der Forschungsarbeit von KOMINA erforscht, bedürfen jedoch nach einer Reflexion der Forschungsmethodik und den Ergebnissen einer weiteren Verfeinerung und Erweiterung.

Forschungsziel F1: Es wird belegt, dass das in KOMINA erforschte Kompetenzstrukturmodell sehr grobgranular ist, was einerseits eine allgemeine Anwendbarkeit erlaubt, andererseits keine Ableitung konkreter Handlungsempfeh-

lungen für die Praxis ermöglicht. Daher ist exemplarisch zu zeigen, wie die grobe Struktur des Modells in eine feine, für die Lehre nutzbare – und damit über die abstrakte Vorstellung der erforschten Kompetenzen hinausgehende – Strukturierungskomponente als Vorarbeit für ein didaktisches System für das Design eingebetteter Systeme zu überführen ist.

Forschungsziel F2: Als wirksamste Veranstaltungsform zur Lehre mit eingebetteten Mikro- und Nanosystemen (EMNS) gelten Praktika [Schubert et al., 2010, S. 3]. Theoretische Konzepte für Experimente zur Kompetenzförderung wurden in KOMINA hinsichtlich ihrer Machbarkeit untersucht. Dabei spielten die Faktoren technische Machbarkeit, Kosten, Umsetzungs- und Durchführungszeit, also institutionelle Besonderheiten, eine Rolle. Durch nichtteilnehmende Beobachtungen innerhalb des in KOMINA entwickelten Praktikums soll ein detaillierter Einblick in die Struktur der Kompetenzen von Studierenden ermöglicht werden. Zu dieser Struktur gehören die Kompetenzen als Voraussetzung sowie das implizite Mikrosystemverständnis.

Forschungsziel F3: Simulationen und webbasierte Experimente sind Alternativen zu realen Laborexperimenten. Simulationen eignen sich für praktisch nicht durchführbare Experimente (fehlende Laborausstattung, zu hohe Kosten etc.). Darüber hinaus wird erforscht, mit welchen Mitteln die in F2 identifizierten Lernhürden durch lernunterstützende Hard- und Software überwunden werden können.

Forschungsverlauf

Das Forschungsprojekt des Autors war Teil des in den Jahren 2011 bis 2014 von der DFG geförderten Projektes KOMINA, dessen Erfolg sich in zahlreichen Publikationen manifestierte. Die Forschungsmethodik und Ergebnisse von KOMINA werden in dieser Arbeit diskutiert und dienen als Motivation sowie informatikdidaktische Grundlage. Die Arbeit orientiert sich daher an der Chronologie der Projektschritte und gliedert sich in vier Hauptphasen (I-IV). Zur Abgrenzung der Arbeitsschritte des Autors von gemeinschaftlichen Projektarbeiten werden in Abbildung 1.1 drei Farben verwendet. Dabei entstandene Publikationen sind als Raute den jeweiligen Hauptphasen zugeordnet. Arbeitspakete in Zusammenarbeit mit allen Projektpartnern in KOMINA  . Forschungsarbeiten am Lehrstuhl Didaktik der Informatik und E-Learning – gemeinsam mit Steffen Büchner konzipiert, umgesetzt und publiziert  . Alleinige Forschungsarbeiten des Autors  .

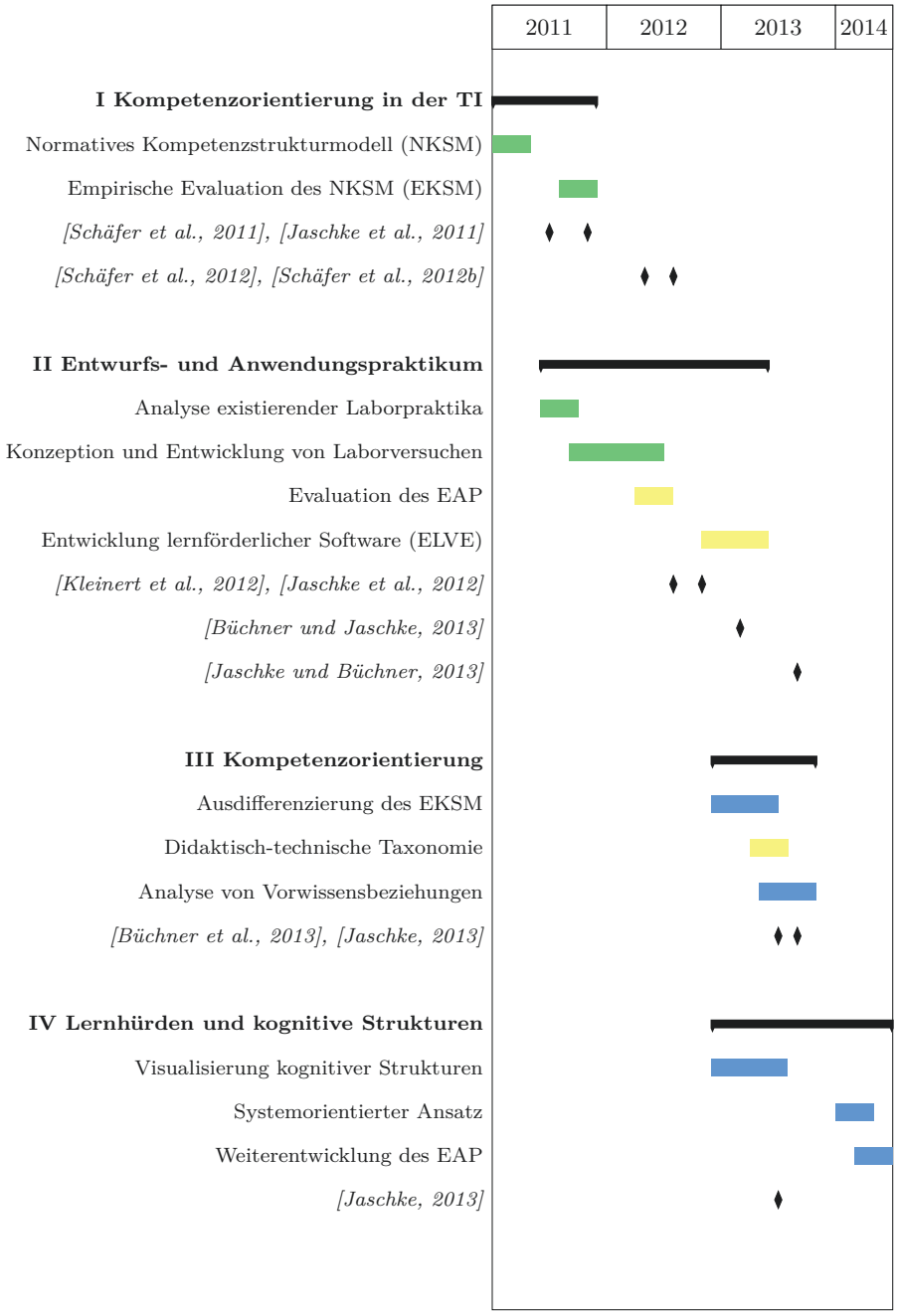


Abbildung 1.1.: Forschungsverlauf

1.3. Terminologie und Zielgruppendefinition

Eine informatikdidaktische Diskussion über das Design eingebetteter Systeme erfordert eine Einordnung in den Forschungsstand der Informatikdidaktik (siehe Kapitel 2). Zunächst ist der Forschungsgegenstand Informatiksysteme von den in der Literatur verwendeten Begriffen *eingebettete Systeme* beziehungsweise *cyber-physische Systeme* abzugrenzen. Außerdem ist eine Beschreibung der Zielgruppe, Studierende der Informatik als Entwickler eingebetteter Systeme, zur Klärung des Forschungsgegenstandes notwendig. Nur wenige internationale oder nationale Studiengänge fokussieren unmittelbar die Ausbildung von Entwicklern eingebetteter Systeme, sondern betrachten diese als Spezialisierung innerhalb der Studiengänge Informatik oder technische Informatik – seltener auch Elektrotechnik.

Um Kompetenzen in der technischen Informatik zu fördern, werden in dieser Arbeit Theorien, wie die systemorientierte Didaktik (SDdI) oder Konzepte didaktischer Systeme, auf die Lehre zu eingebetteten Systemen übertragen (siehe Kapitel 2). Die SDdI hatte Informatiksysteme als Schwerpunkt, welche daher als Ausgangspunkt dieser Arbeit betrachtet werden.

Informatiksysteme

In der Definition von Informatiksystemen nimmt die Hardware, also eine der technischen Informatik zuzuordnenden Disziplin, eine gleichberechtigte Stellung ein:

> „Als Informatiksystem bezeichnet man die spezifische Zusammenstellung von Hardware, Software und Netzverbindungen zur Lösung eines Anwendungsproblems. Eingeschlossen sind alle durch die Einbettung des Systems in den Anwendungsbereich beabsichtigten oder verursachten nicht-technischen Fragestellungen und ihre Lösungen, also Fragen der Gestaltung des Systems, der Qualifizierung der Nutzer, der Sicherheit sowie der Auswirkungen und Folgen des Einsatzes . . . “ [Claus und Schwill, 2006, S. 314].

Nicht-technische Fragestellungen implizieren Systemeigenschaften, welche über eine technische Betrachtung des Systems hinausgehen und einen Bezug zu sozialen Systemen herstellen. Man spricht dann von sozio-technischen Systemen (vgl. Kapitel 2), hier einem allgemeinen Informatiksystem, welches nicht durch eine Realisierung als digitales Computersystem oder hybrides (Analog/Digital gemischtes) System bestimmt wird. Stechert stellt fest, dass weitere, von der Sicht auf das System abhängige Definitionen zu Systemen, Computersystemen, Datenverarbeitungssystemen, Rechnersystemen etc. existieren, die teils widersprüchlich den Begriff System verwenden [Stechert, 2009, S. 35 ff] und charakterisiert drei Sichten auf Informatiksysteme, das *nach außen sichtbare Verhalten (Black-Box)*, die *innere Struktur (White-Box)* und *Implementierungsaspekte*, welche zur Kompetenzentwicklung beitragen sollen (vgl. Kapitel 2) [Stechert, 2009, S. 47]. Diese Sichten sind, bezugnehmend auf [Claus und Schwill, 2006], einerseits zur Förderung von

Kompetenzen von Systementwicklern und andererseits zur Qualifizierung von Systemnutzern zu erforschen.

Eingebettete Mikro- und Nanosysteme

Ein allgemeines Informatiksystem, wie der PC, zeichnet sich durch seine universelle Programmierbarkeit aus. Eingebettete Systeme hingegen zielen auf den konkreten Einsatz in technischen Umgebungen ab, an welche auch sämtliche Systemkomponenten wie die Software, Hardware und Schnittstellen angepasst werden müssen. Dazu ist die Umgebung – der physikalische Prozess – als Teil des Systems zu betrachten. Aus dem kontextspezifischen Einsatz in nahezu allen Lebens- und Arbeitsbereichen der Gesellschaft folgen die besonderen Anforderungen an eingebettete Systeme und deren Entwickler.

Eingebettete Systeme sind dedizierte Systeme, bestehend aus Software-, Hardware- und Vernetzungskomponenten zur reaktiven Steuerung, Regelung oder Überwachung peripherer technischer Prozesse, in die sie eingebettet werden [Claus und Schwill, 2006][Magenheim, 2003].

Mikrosysteme sind miniaturisierte technische Systeme, deren Bauteile und Komponenten typische Strukturgrößen beziehungsweise Toleranzen im Mikrometer- bis Nanometerbereich besitzen [Greiner et al., 2009, S. 7].

Nanosysteme sind miniaturisierte Systeme mit Halbleiterstrukturen im Größenbereich $\leq$100nm [ISO, 2010].

Je nach peripherem technischen Prozess und dem Zweck des Systems, werden Echtzeiteigenschaften, geringe Leistungsaufnahme, Störungssicherheit, Temperaturunabhängigkeit, Robustheit, Korrektheit, Zuverlässigkeit und weitere nichtfunktionale Anforderungen zugleich gefordert. Diese Faktoren führen wiederum zu einem schwierigen Spezifikationsprozess [Claus und Schwill, 2006, S. 224f.]. Dies wird durch die mit den eingebetteten Systemen in Wechselwirkung stehenden technischen Systeme und Prozesse noch bestärkt, weshalb die Entwicklung eingebetteter Systeme ein interdisziplinäres Arbeitsfeld ist. Des Weiteren sind eingebettete Systeme hybride Systeme, bestehend aus analogen und digitalen Komponenten, was Komponenten zur Umformung der Messgrößen des technischen Prozesses einschließt. Der sozio-technische Kontext hängt vom konkreten eingebetteten System ab und betrifft zum Beispiel Risiken und Auswirkungen durch den Einsatz, die Anwendung oder die Entwicklung des Systems, also über die Mensch-Maschineninteraktion hinausgehende, für den Benutzer verborgene, Prozesse mit gesellschaftlichen Auswirkungen.

In dieser Arbeit wird von eingebetteten Systemen bei der Betrachtung des Gesamtsystems gesprochen. Bei der Verwendung des Begriffs *eingebettetes Mikro- und Nanosystem* stehen die Struktur der Hardware oder die Herstellungsprozesse im Vordergrund.

Cyber-physische Systeme

> „Cyber–physical systems are physical, biological, and engineered systems whose operations are integrated, monitored, and/or controlled by a computational core. Components are networked at every scale. Computing is deeply embedded into every physical component, possibly even into materials. The computational core is an embedded system, usually demands real–time response, and is most often distributed. The behavior of a cyber–physical system is a fully–integrated hybridization of computational (logical), physical, and human action“ [Lee und Seshia, 2012].

Cyber-physisches System (CPS) hat sich in den vergangenen Jahren als Bezeichnung von vernetzten eingebetteten Systemen etabliert. Diese beschreibt die Verschmelzung der physikalischen Welt mit räumlich verteilten Informatiksystemen – dem Cyberraum – durch einen hohen Grad der Vernetzung [acatech, 2011]. Im Anwenderbereich sind Smartphones, Spielkonsolen, Wohngebäude etc. vernetzt zu einem persönlichen sozio-technischen CPS mit wesentlichem Einfluss auf die Lebenswelt der Anwender, also mit persönlichen und gesellschaftlichen Komponenten. Ferner ändert sich auch die Arbeitswelt, hin zu cyber-physischen Arbeitsprozessen der *Industrie 4.0*, in denen der Mensch überwiegend überwachende, reflektierende und optimierende Aufgaben übernimmt (vgl. [Spath, 2013], [Jaschke, 2014b], [Jaschke, 2014a]). CPS sind daher aktuell viel beachtet und stellen ein breites Forschungsgebiet der Informatik dar, das aufgrund besonderer Teilaspekte, wie dem Grad der Vernetzung, die Verbindung zu physikalischen Prozessen, Zuverlässigkeit und weiteren Randbedingungen, eine nur schwer zu bewältigende Komplexität hervorbringt. Bislang fehlen der Informatik ganzheitliche Lösungsansätze und formale Beschreibungssprachen, welche alle Teilaspekte vereinen [Lee, 2008], [Rajkumar et al., 2010]. Klassische Vorgehensweisen und Spezifikationstechniken zur Modellierung von eingebetteten Systemen stoßen bei der Entwicklung cyber-physischer Systeme an Grenzen und stellen eine wissenschaftliche Herausforderung zur Bewältigung der Prozesskomplexität dar [acatech, 2011].

Die besondere Stellung von CPS – die durchgängige Vernetzung zu einem Internet der Dinge – war nicht zentral im DFG-Projekt KOMINA und im Forschungsprojekt des Autors. Im Verlauf dieser Arbeit wird der Begriff eingebettete Systeme analog zu den Publikationen der an KOMINA beteiligten Wissenschaftler verwendet. Dieser Begriff beinhaltet explizit die Systemvernetzung über Domänengrenzen hinweg und schließt diese nicht aus. Im Bewusstsein der besonderen Anforderung an CPS sowie der wissenschaftlich-praktischen Aktualität umfassen die im Forschungsprojekt des Autors erforschten Lehr-Lernprozesse in Laborpraktika Vernetzungsaspekte. Durch die Kommunikation eines klassischen eingebetteten Systems über die Domänengrenze einer Hausautomation hinweg und einem Smartphone via Bluetooth werden CPS eingeführt, spielen jedoch eine eher untergeordnete Rolle, sodass die Verwendung des Begriffes eingebettete Systeme in dieser Arbeit angemessen ist. Vielmehr sollen Studierende durch einfache Beispiele für das Ge-

biet CPS motiviert werden, in denen die Systemkomplexität keine abschreckende Wirkung hat.

Zielgruppe

Lehrveranstaltungen zu eingebetteten Systemen sind Bestandteil verschiedener Studiengänge – beispielsweise Elektrotechnik, technische Informatik, Automatisierungstechnik oder Robotik. Die Entwicklung eingebetteter Systeme ist historisch betrachtet eine Kombination aus Informatik und Elektrotechnik und damit der technischen Informatik zugeordnet. Zur Klärung der Verortung in Empfehlungen (vgl. Kapitel 2) nimmt die ACM und das IEEE folgende, für die Beschreibung der Zielgruppe und Interpretation des Forschungsstandes wichtige, Unterteilung vor (siehe Abbildung 1.2) [ACM/IEEE, 2005]. Die Entwicklungsarbeit (B) beschreibt auf der horizontalen Achse das Verhältnis von (A) Theorie, Prinzipien und Innovation zu (C) Nutzung, Bereitstellung und Konfiguration. Auf der vertikalen Achse sind fünf Aspekte von Informatiksystemen aufgetragen, welche je nach Studiengang von besonderer Relevanz für die künftigen Tätigkeiten der Studierenden sind.

1 Computer Hardware und Architektur

2 Systeme und Infrastruktur

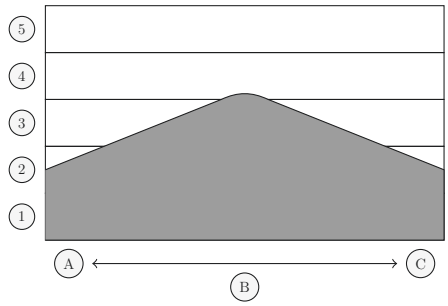
3 Softwaremethoden und -technologien

4 Anwendungstechnologien

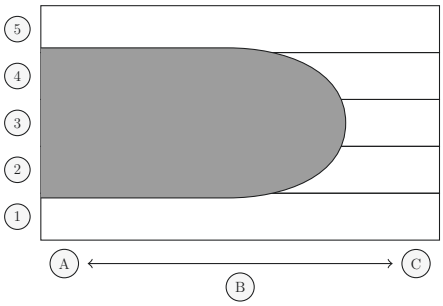
5 Betriebliche Aspekte und Informatiksysteme

Die Begriffe dieser Ordnung entstammen verschiedenen Blickwinkeln auf Informatiksysteme und umfassen die eher technischen Abstraktionsebenen Computer Hardware und Architektur (1), Softwaretechnologien (3) sowie Anwendungstechnologien (4). Nur scheinbar unterbrochen wird diese Ordnung durch Begriffe, die eher der Integration in (betriebliche) Prozessstrukturen dienen – Systeme und Infrastruktur (2) und Betriebliche Aspekte (5). Das spätere Berufsfeld eines Informatikers wird dadurch adäquat abgebildet, da diese Aspekte des gesamten Entwicklungsprozesses, welcher immer unter Berücksichtigung betriebswirtschaftlichen Randbedingungen erfolgt, nicht isoliert betrachtet werden können. Die Ebene System und Infrastruktur (2) ermöglicht eine Trennung von Tätigkeiten, welche zwar das System, seine Komponenten und die zugehörige Infrastruktur betrachten, die eigentliche Hardwareentwicklung und Integration aber außer Acht lassen. Mit dieser Ordnung lassen sich beide Sichtweisen bei einem unterschiedlich gewichteten Theorie/Praxis Verhältnis und damit das gesamte Berufsfeld von Informatikern auch ohne Bezug zur Systementwicklung beschreiben.

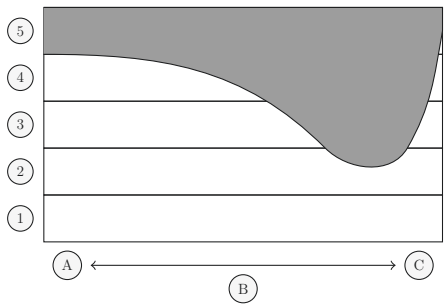
Betriebliche Aspekte stehen bei der Entwicklung eingebetteter Systeme nicht im Fokus. Den Studiengängen *Information Systems*, *Information Technology* und *Soft-*



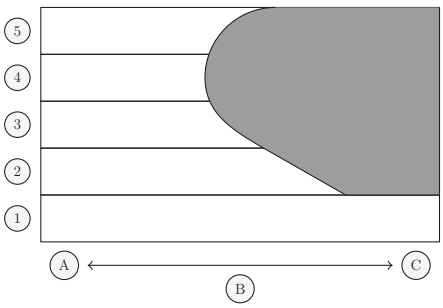
(a) Computer Engineering



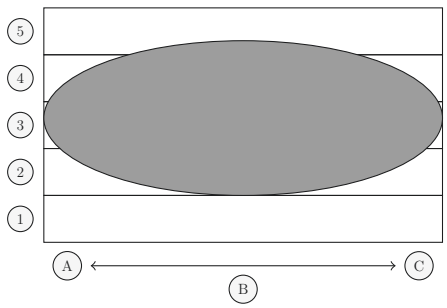
(b) Computer Science



(c) Information System



(d) Information Technology



(e) Software Engineering

Abbildung 1.2.: Vergleich der Informatikdisziplinen (vgl. [ACM/IEEE, 2005, S. 16ff])

ware Engineering wird bei der Gewichtung von Inhaltsbereichen, welche der Entwicklung eingebetteter Systeme zuzuordnen sind (*Intelligente Systeme, Digitale Logik, Eingebettete Systeme, Elektronik, Schaltungen und Systeme, Digitale Signalprozessoren, Very-Large-Scale Integration, Hardwaretest und Fehlertoleranz*), nur in geringem Umfang Relevanz beigemessen [ACM/IEEE, 2005, S. 24]. Es wird klar, dass dieses Gebiet den Studiengängen Computer Engineering (CE) und Computer Science (CS) vorbehalten ist, also eine wissenschaftliche Disziplin auf den unteren Ebenen *Computer Hardware und Architektur* bis *Anwendungstechnologien* darstellt. Die Elektrotechnik ist nicht Teil dieses Vergleiches und daher auch nicht ausgenommen. Die Zielgruppe bei Untersuchungen im KOMINA-Projekt waren Studierende der Informatik und technischen Informatik, deren Ausbildungsziel der Erwerb von Kompetenzen als Entwickler von Informatiksystemen und damit auch von eingebetteten Systemen ist [Schubert et al., 2010]. Die Probandengruppe in den qualitativen Studien ist jedoch sehr heterogen zusammengesetzt, da an der Universität Siegen das Fach Informatik mit einem Anwendungsfach mit einem Anteil von einem Drittel kombiniert studiert wird. Im Fall der Studierenden mit Anwendungsfach Elektrotechnik entspricht dies dem früheren, bis 2009 angebotenen, Studiengang Technische Informatik (Computer Engineering). Weitere Nebenfächer sind Medienwissenschaften, Mathematik – also Studiengänge mit weniger Affinität zur Systemhardware – sowie Automotive System Engineering. Der Fakultätentag Informatik, der die gemeinsamen Belange der Universitäten der Bundesrepublik Deutschland im Hinblick auf die Förderung der Zusammenarbeit in allen wissenschaftlichen Fragen und eine Koordinierung der Ausbildung im Bereich Informatik vertritt, benennt allgemeine vom Anwendungsfach unabhängige Ausbildungsziele und verortet eingebettete Systeme im Bachelorstudiengang Informatik.

> „**Eingebettete Systeme, Systemsoftware, Rechnernetze:** Die Studierenden sollen das Zusammenspiel zwischen der Hardware und Software auf unterschiedlichen Ebenen und die Funktionsweise von verteilten und vernetzten Systemen verstehen [...] Dazu müssen sie die hierfür notwendigen Grundlagen digitaler und kontinuierlicher Systeme beherrschen [Hervorhebung im Original]“ [Informatik, 2004, S. 10].

Betrachtet man nun die Studienfächer CE und CS, so erkennt man, dass die Schnittmenge bezogen auf das Gebiet der eingebetteten Systeme, bestehend aus Hardware, Software und Vernetzungskomponenten, relativ gering ist. Für die Forschungsarbeit des Autors ist somit aufgrund der heterogenen Zielgruppe mit ebenso heterogenen Vorkenntnissen, Erwartungshaltungen, Motivationen und Leistungen zu rechnen. Gleiches gilt für das Arbeitsgebiet künftiger Entwickler, da der Entwicklungsprozess, die verwendeten Technologien und damit auch die notwendigen Kompetenzen immer vom entwickelten Produkt abhängig sind.

Eingebettete Systeme als Informatikdisziplin

Unstrittig ist, dass die Entwicklung eingebetteter Systeme eine Informatikdisziplin ist und trotzdem in den Empfehlungen der ACM/IEEE noch nicht entsprechend repräsentiert wird. Teilaspekte eingebetteter Systeme finden sich in den Studiengängen CE und CS wieder (siehe Abbildung 1.2). Die Relevanz eingebetteter Systeme in Forschung und Lehre erfordert eine neue Sichtweise auf Studiengänge der Informatik, in denen die Entwicklung eingebetteter Systeme als eigenständige Disziplin verstanden und nicht als Teilmenge einem der Studiengänge nach ACM/IEEE zugeordnet wird. In Anbetracht der Breite des Fachgebietes und den spezifischen Konzepten, Technologien und Tools lassen sich offensichtlich nicht alle zur Entwicklung eingebetteter Systeme notwendigen Kompetenzen, beispielsweise in einem Informatikstudium, erreichen. Der Versuch, die Breite des Forschungs- und Praxisfeldes eingebetteter Systeme unter Berücksichtigung der anwendungsspezifischen Gewichtung von Hard- und Softwarekomponenten in diesem zweidimensionalen Diagramm angemessen darzustellen ist ziemlich sicher zum Scheitern verurteilt. Dennoch eignet sich eine solche Visualisierung, um die Unterschiede verschiedener Curricula vergleichend darzustellen und zu zeigen, dass Konzepte eingebetteter Systeme keiner der aufgeführten Disziplinen allein zugeordnet werden können. Die Forderungen nach einer Verortung von eingebetteten Systemen in allen beschriebenen Curricula mit einer Anpassung von Breite und Tiefe zeigt, dass ein geeignetes Verhältnis zwischen Theorie und Praxis individuell von jeder Universität gefunden werden muss.

> „Scientists and engineers who are properly trained in the fundamentals of computation, control, networking, and software engineering are critically needed. CPS basics need to be added to the lingua franca of all technical graduates. Creative trade-offs between depth and breadth may need to be adopted“ [Rajkumar et al., 2010].

Verzichtet man auf eine anwendungsspezifische Gewichtung und kombiniert die eher hardwareorientierten theoretischen Aspekte der technischen Informatik mit softwareorientierten Konzepten der allgemeinen Informatik (siehe Abbildung 1.3), entsteht, wie bereits erwähnt, ein sehr breites und tiefes Forschungs- sowie Praxisfeld zu eingebetteten Systemen.

Wie in den Begriffsdefinitionen ersichtlich, umfasst die Entwicklung eingebetteter Systeme auch die unteren Systemschichten (Ebene 1, Computer Hardware und Architektur). Durch ihren sozio-technischen Kontext sowie Mensch-Maschinenschnittstellen sind Anwendungstechnologien ebenfalls Teilmenge dieses Lehrgebietes und ergänzen daher auch die technische Informatik um Aspekte der Ebene (4). Entwickler müssen dabei einerseits die theoretischen Grundlagen der Domäne verstehen als auch in der Lage sein reale, praxisrelevante Systeme zu erschaffen. Die Lehre zu eingebetteten Systemen muss also entgegen der gegenwärtigen Verortung innerhalb der technischen Informatik als eigenständige Disziplin betrachtet werden,

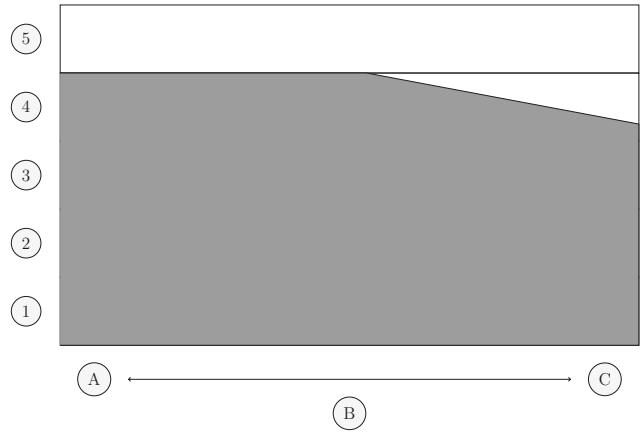


Abbildung 1.3.: Repräsentation eingebetteter Systeme

welche Teile weiterer Informatikdisziplinen, je nach institutioneller Ausrichtung, enthält. Es muss ein sinnvolles Verhältnis zwischen Theorie und Praxis sowie eine angemessene Repräsentation der Ebenen (1-4) gefunden werden. Dies gilt dementsprechend auch für die Betrachtung von Kompetenzen künftiger Entwickler, von denen nicht Tiefe und Breite über alle Ebenen des Systementwurfs zugleich gefordert werden darf.

1.4. Gliederung der Arbeit

Kapitel 1: In diesem Kapitel wurde die Motivation zur vorliegenden Arbeit mit der Relevanz eingebetteter Systemen und damit gleichzeitig existierenden Forschungsbedarfen zur Hochschuldidaktik der technischen Informatik begründet. Entgegnet werden soll diesem Mangel durch das Erreichen der drei beschriebenen Forschungsziele. Die weiteren Kapitel orientieren sich an dem Forschungsvorgehen (siehe Abbildung 1.4).

Kapitel 2: Arbeiten zur Kompetenzforschung in der technischen Informatik, die Analyse von Curriculaempfehlungen sowie die Klärung von Begrifflichkeiten zur Schaffung des Kontextes sind Teil dieses Kapitels. Dies umfasst auch Grundlagen des Entwurfs eingebetteter Systeme. Eine Taxonomie zur Vergleichbarkeit fachdidaktischer Publikationen zur technischen Informatik wird vorgestellt. Strukturierungsansätze werden diskutiert und auf ihre Anwendbarkeit hin untersucht.

Kapitel 3: Dieses Kapitel beinhaltet eine Reflexion des Projektes KOMINA und die Ergebnisdiskussion. Die Entwicklung eines Kompetenzstrukturmodells,

welches exemplarisch in einem Entwurfs- und Anwendungspraktikum umgesetzt wurde, wird diskutiert. Erkenntnisse aus der formativen Evaluation begründen den weiteren Forschungsbedarf.

Kapitel 4: Eine Verfeinerung des Praktikums nach einem systemorientierten Ansatz unter Weiterentwicklung der Komponenten didaktischer Systeme – Wissensstrukturen und lernförderliche Hard-/Software – wird begründet. Kognitive Strukturen werden zur weiteren Analyse von identifizierten Lernhürden abgeleitet und diskutiert.

Kapitel 5: Die Forschungsarbeiten werden zusammengefasst. Ein Fazit zu den Forschungszielen beschreibt die wesentlichen Aspekte und Ergebnisse zur Hochschuldidaktik der technischen Informatik. Abgeschlossen wird diese Arbeit mit offenen Fragen und einem Ausblick.

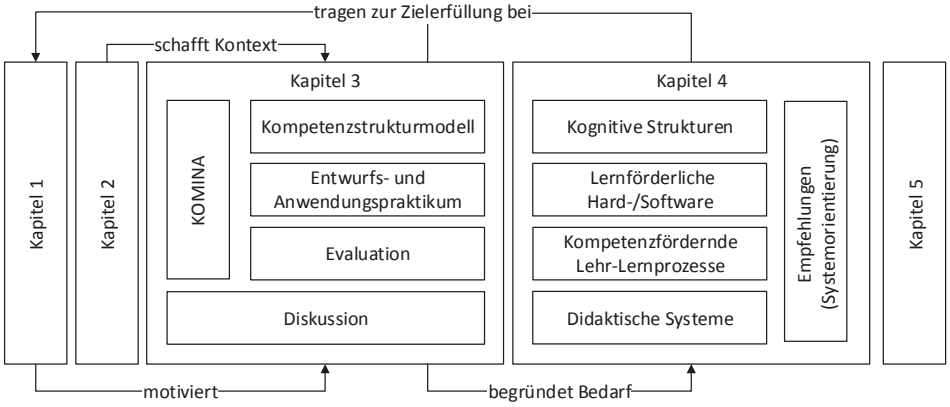


Abbildung 1.4.: Struktur der Arbeit

2. Grundlagen und Vorarbeiten

> „Es tritt ein Wandel im Lehrverständnis ein, wonach Lehren strikt aus der Perspektive des Lernens gedacht wird. Gegenstand der Lehre ist nicht mehr das zu lehrende Wissen, sondern die zu erlernende Kompetenz, die zwar das Wissen einschließt, jedoch den Schwerpunkt verlagert von der Vermittlung von Inhalten hin zum Erwerb von Kompetenzen. Es ist naheliegend, dass ein solcher Wandel neue Anforderungen an die Tätigkeit und die Rolle der Lehrenden heranträgt“ [Paetz et al., 2011, S. 36].

Die sich durch diesen Wandel ergebende neue Perspektive wird als Kompetenzorientierung bezeichnet und vereint eine Inhalts- und Anwendungskomponente zur Kompetenz, also über eine ausschließliche Wissensvermittlung hinausgehende didaktische Kategorie. In diesem Kapitel werden die theoretischen Grundlagen zur Kompetenzorientierung (Modellierung/Modelle) und ihre Bedeutung für die Ausbildung von Entwicklern eingebetteter Systeme beschrieben. Außerdem wird eine Taxonomie zur Beschreibung von didaktischen Beiträgen vorgestellt, welche im weiteren Verlauf der Arbeit zur Klassifizierung von Aufgaben sowie der Analyse von Barrieren des Kompetenzerwerbs verwendet wird.

2.1. Kompetenzorientierung

Ein Ziel im Forschungsprojekt des Autors ist es, eine theoretisch fundierte Strukturierung von Lehr-Lernprozessen der technischen Informatik – im Speziellen zu eingebetteten Systemen – zu ermöglichen. Dafür werden Kompetenzen hinsichtlich ihrer Niveaustufen und Vernetzung untersucht. Bevor eine Diskussion von domänenspezifischen Kompetenzen möglich ist, wird der Terminus Kompetenz innerhalb dieser Arbeit geklärt. Viele Lernzielformulierungen, in Modulbeschreibungen universitärer Lehrveranstaltungen, aber auch Empfehlungen zu Curricula, berücksichtigen die Ergebnisse der Kompetenzforschung (noch) nicht. Die in diesem Kapitel diskutierte Literatur verwendet dabei die Begriffe *learning outcome, outcome, output, objective, competence, competency und Kompetenz* teilweise synonym, was zu einer erschwerten Vergleichbarkeit von Forschungsergebnissen führt. Die Gewichtung einzelner Kompetenzen ist immer geprägt von institutionellen beziehungsweise kulturspezifischen Sichtweisen und muss im Gesamtkontext, der Quellenherkunft, betrachtet werden.

Die Organisation for Economic Co-operation and Development (OECD) gruppiert Schlüsselkompetenzen in drei Kategorien – interaktive Anwendung von Medien und Mitteln (Tools), interagieren in heterogenen Gruppen sowie eigenständiges Handeln:

> „Eine Kompetenz ist mehr als nur Wissen und kognitive Fähigkeiten. Es geht um die Fähigkeit der Bewältigung komplexer Anforderungen, indem in einem bestimmten Kontext psychosoziale Ressourcen (einschließlich kognitive Fähigkeiten, Einstellungen und Verhaltensweisen) herangezogen und eingesetzt werden“ [OECD, 2005, S. 6].

Zwischen den verschiedenen länderspezifischen Qualifikationssystemen und den Vorstellungen von zu erwerbenden Kompetenzen innerhalb der Europäischen Union vermittelt der Europäische Qualifikationsrahmen für lebenslanges Lernen, in dem acht Stufen von Kompetenzen unterschieden werden (Stufe 6 entspricht dem Bachelorniveau; vgl. [EU, 2008]). Der Metarahmenplan der Europäischen Union wurde vom Arbeitskreis Deutscher Qualifikationsrahmen unter der Führung des Bundesministeriums für Bildung und Forschung (BMBF) und der Kultusministerkonferenz (KMK) im Deutschen Qualifikationsrahmen für lebenslanges Lernen (DQR) umgesetzt:

> „**Kompetenz** bezeichnet im DQR die Fähigkeit und Bereitschaft des Einzelnen, Kenntnisse und → Fertigkeiten sowie persönliche, soziale und methodische Fähigkeiten zu nutzen und sich durchdacht sowie individuell und sozial verantwortlich zu verhalten. Kompetenz wird in diesem Sinne als umfassende Handlungskompetenz verstanden [Hervorhebung im Original]“ [AK DQR, 2013, S. 8].

In dieser Arbeit wird für den Begriff Kompetenz die Definition von [Weinert, 2001] beziehungsweise [Klieme et al., 2007] verwendet, welche in der nationalen Bildungsforschung mit ihrer Erfassung vielfältiger Einflussfaktoren auf ein erfolgreiches Bewältigen von Anforderungssituationen weit verbreitet ist (vgl. [Stechert, 2009, S. 17], [Klieme et al., 2007, S. 72 ff]):

> „In Übereinstimmung mit Weinert (2001, S. 27f.) verstehen wir unter Kompetenzen die bei Individuen verfügbaren oder von ihnen erlernbaren kognitiven Fähigkeiten und Fertigkeiten, bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“[Klieme et al., 2007, S. 27].

Bis zum Jahre 2001 beschränkte sich die Vorstellung von Kompetenzen zunächst auf kognitive Fähigkeiten und Fertigkeiten sowie auf einen begrenzten Sektor von Kontexten und Situationen. Obiges gilt heute als Referenzzitat der Kompetenzforschung in Deutschland. Die Definition wird der Komplexität einer umfassenden

Handlungsfähigkeit gerecht und hebt die Problemlösefähigkeit, welche in der Hochschulausbildung sowie im Hinblick auf das Forschungsgebiet als zentrales Element gesehen wird, hervor. Kompetenzen werden mit affektiven, nicht-kognitiven Fähigkeiten und Bereitschaften (motivational, volitional, sozial) definiert. Im Allgemeinen sind Kompetenzen bereichsspezifische (z. B. Kompetenzen zur Entwicklung von eingebetteten Systemen), aber dennoch begrenzt verallgemeinerbare Dispositionen (vgl. [EU, 2008],[OECD, 2005],[AK DQR, 2013]). Eine Transformation durch einen generischen Satz von Schlüsselkompetenzen kann jedoch keine fachspezifischen Kompetenzen ersetzen [Klieme, 2004]. In allen oben genannten Definitionen zu Kompetenzen nehmen die Einstellungen und Verhaltensweisen eine gleichberechtigte Rolle zu Fähigkeiten und Fertigkeiten ein. Ferner werden im Schwerpunktprojekt *Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen* Kompetenzen definiert als

> „. . . kontextspezifische kognitive Leistungsdispositionen, die sich funktional auf Situationen und Anforderungen in bestimmten Domänen beziehen“ [Klieme und Leutner, 2006, S. 879].

Daraus folgt, dass je Fachdisziplin ein spezifisches Kompetenzmodell zu erforschen ist. Diesem Sachverhalt wird unter anderem im Forschungsprogramm Kompetenzmodellierung und Kompetenzerfassung im Hochschulsektor des BMBF Rechnung getragen, in dem grundlagenorientierte Projekte zur Kompetenzforschung im tertiären Bildungssektor gefördert werden [Blömeke und Zlatkin-Troitschanskaia, 2013]. In dieser Arbeit werden Kompetenzen, welche zur Entwicklung von eingebetteten Systemen notwendig sind (Entwicklerkompetenzen), erforscht.

Die Entwicklung und Messung von Kompetenzen in diversen Ausprägungen (Niveaustufen) erfordert eine Operationalisierung der zugehörigen Kompetenzbeschreibungen. Sie ist nur dann möglich, wenn affektive Aspekte unberücksichtigt bleiben, da diese nur schwer zu messen sind. Aus diesen pragmatischen Gründen wird in diesem Forschungsprojekt, wie auch bei der Formulierung von Bildungsstandards der KMK, eine Einschränkung auf die kognitiven Leistungsbereiche zugunsten der Operationalisierbarkeit vorgenommen [Klieme, 2004]. Eine Diskussion der nichtkognitiven Leistungsbereiche wird dennoch durch strukturierte und intersubjektiv vergleichbare Beobachtungen ermöglicht.

Werden Erkenntnisse der Kompetenzforschung nicht berücksichtigt und Operatoren zur Lernzielbeschreibung verwendet, die keine Niveaubestimmung erlauben, da sie keine Referenz zu einer Taxonomie enthalten, bedeutet dies für die informatikdidaktische Diskussion, dass die Vergleichbarkeit verringert wird. Ferner wird dann selten Bezug auf Problemlösungen und variable Situationen genommen, was zu einer Sammlung von Wissensartefakten (Themenlisten) führt.

> „Kompetenz stellt die Verbindung zwischen Wissen und Können [. . .] her und ist als Befähigung zur Bewältigung von Situationen beziehungsweise von Auf-

> gaben zu sehen. Jede Illustration oder Operationalisierung einer Kompetenz muss sich daher auf konkrete Anforderungssituationen beziehen“ [Klieme et al., 2007, S. 73].

Für die Messbarkeit von Kompetenzen impliziert dies, dass neben der Beschränkung auf kognitive Leistungsbereiche zugunsten der Operationalisierbarkeit konkrete Anforderungssituationen aus dem fachspezifischen Kontext der Zielgruppe herangezogen werden müssen. Ferner seien Kompetenzen nicht durch einzelne isolierte Leistungen darstell- beziehungsweise erfassbar, was dazu führe, dass eine Anforderungssituation, in der eine bestimmte Kompetenz gefordert sei, ein Leistungssprektrum umfasse [Klieme et al., 2007, S. 73]. Dies ist außerdem mit der Ausprägung von Kompetenzen, den Kompetenzfacetten, begründet (Fähigkeit, Wissen, Verstehen, Können, Handeln, Erfahrung und Motivation). Daraus folgt, dass eine Kompetenzmessung durch einen Wissenstest allein nicht der Komplexität beziehungsweise Mehrdimensionalität der Kompetenzen mit ihren Facetten gerecht wird. Außerdem wird die Komponente *Können* durch einen Wissenstest nicht abgedeckt. Für die HdTI hat die Messung der Kompetenzaneignung durch praktische Versuche beziehungsweise Fragestellungen zu konkreten variablen Situationen – den Kontexten – zu erfolgen. Der Vergleich der Kompetenzen zweier Probandengruppen anhand von Tests ist nicht möglich. Angemessener ist hier eine Beobachtung von Probanden, um Unterschiede in der Kompetenzentwicklung zu erkennen und individuelle Lernhürden aufzudecken sowie zu klassifizieren. Eine quantitative Überprüfung von Maßnahmen zur Kompetenzförderung kann dadurch nach aktuellem Forschungsstand zur HdTI nicht im Rahmen von *Pretests* und *Posttests* erfolgen.

Die Forschungsergebnisse des DFG-Projektes Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lernprozessen für Modellierung und Systemverständnis in der Informatik (MoKoM), in dem Informatiker und Psychologen gemeinsam forschen, dienen im Projekt KOMINA und somit auch im Promotionsvorhaben des Autors als Orientierungshilfe in der Forschungsmethodik. Ebenso wurden verallgemeinernde nicht-kognitive Kompetenzen des auf Softwareentwicklung fokussierten Projektes in das Kompetenzmodell von KOMINA transformiert [Jaschke et al., 2011]. Grund dafür ist, dass bei der Entwicklung von eingebetteten Systemen, wie auch in der allgemeinen Softwareentwicklung, vergleichbare, projektunabhängige Prozessschritte aufeinander folgen. Die Anforderungsprofile von Absolventen der Informatik beinhalten dabei immer nicht-kognitive Kompetenzen. Aufgrund der ständigen Weiterentwicklung von Technologien und Entwicklungskonzepten von eingebetteten Systemen sind insbesondere Motivation und Offenheit für neue Ideen und Anforderungen als nicht-kognitive Kompetenz von Bedeutung [Schäfer et al., 2011].

Kompetenzmodelle

Kompetenzmodelle bieten eine theoretisch fundierte Vorstellung von Outputorientierung in Lehr-Lernprozessen, also der veränderten individuellen Disposition der Lernenden und damit dem Resultat von Bildungsprozessen, was gleichzeitig deren Evaluation ermöglicht.

> „Kompetenzmodelle stellen damit die Grundlage für Operationalisierungen von Bildungszielen dar, die den Output des Bildungssystems über das Erstellen von Testverfahren [...] empirisch zu überprüfen erlauben“ [Klieme et al., 2007, S. 71].

Durch eine einheitliche Verwendung von Operatoren zur Beschreibung der intendierten Kompetenzen wird zwischen abstrakten Bildungszielen (dem Output des Bildungssystems) und Aufgabensammlungen vermittelt [Klieme et al., 2007, S. 71]. Kompetenzmodelle beinhalten die Komponenten (Strukturmodell), die von der Zielgruppe erwartet werden, sowie eine wissenschaftlich begründete Vorstellung von dem Pfad der Kompetenzaneignung auf definierten Niveaustufen (Entwicklungsmodell/Niveaumodell) [Klieme et al., 2007, S. 74]. Eine Vorstellung über die Beschaffenheit von Kompetenzen bildet dabei die Grundlage einer Strukturierung von Lehr-Lernprozessen. Ein Kompetenzstrukturmodell wurde für die Zielgruppe im Rahmen von KOMINA entwickelt und empirisch verfeinert (siehe Kapitel 3).

Um diverse Ausprägungen oder Niveaustufen von Kompetenzen nach qualitativen Merkmalen differenzieren zu können und in der Diskussion einheitlich zu verwenden, ist zunächst die Festlegung einer Taxonomie notwendig. Dazu werden je nach Disziplin und Zielgruppe unterschiedliche Taxonomien vorgeschlagen (vgl. [Fuller et al., 2007]). Für eine Operationalisierbarkeit von Lernzielen in der Informatik erweisen sich die Taxonomie von [Anderson et al., 2000] beziehungsweise [Fuller et al., 2007, S. 163], welche auf der Bloom'schen Taxonomie aufbauen, als geeignet [Stechert, 2009, S. 25]. Differenziert werden dort sechs Kompetenzstufen: 1. Erinnern, 2. Verstehen, 3. Anwenden, 4. Analysieren, 5. Bewerten, 6. Gestalten. In der Lernzieltaxonomie nach [Fuller et al., 2007] findet eine weitere Unterteilung der Niveaustufen in den Bereichen *Interpretieren* und *Produzieren* statt.

> „We can also distinguish between disciplines in which there is an emphasis on learning through interpreting and those in which learning is predominantly achieved through doing. [...] Computing students are expected to do a lot of learning through doing, whether it is learning about software engineering by developing systems of increasing complexity, learning about networking by implementing protocols or learning about group dynamics by working in teams“ [Fuller et al., 2007, S. 163].

Diese Aussage gilt gleichermaßen für die Studierenden der Technische Informatik (TI), da sich lediglich die Schichten des Systementwurfs oder der verwendeten

Technologien unterscheiden, nicht jedoch die grundlegenden Prozesse *Interpretieren* und *Produzieren* (siehe Abbildung 2.1).

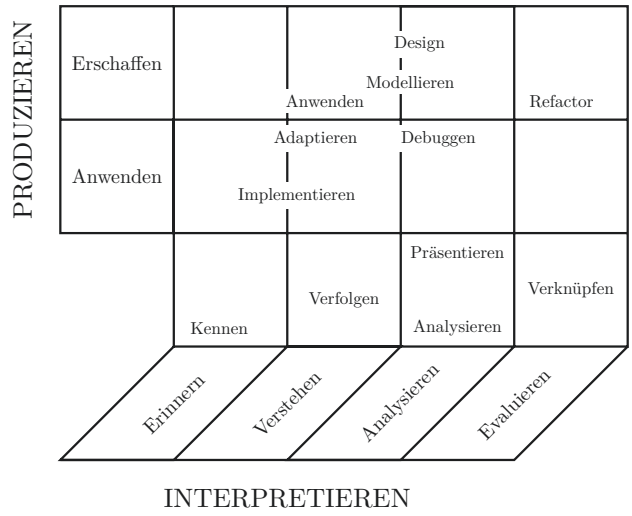


Abbildung 2.1.: Ebenen der Lernzieltaxonomie nach [Fuller et al., 2007, S. 164]

[Fuller et al., 2007] zeigen, dass die meisten von ihnen untersuchten Kurse der Ebene *Verstehen/Anwenden (Implementieren)* zuzuordnen sind. Komplexere Anwendungsprobleme erfordern Fähigkeiten zur Analyse, Synthese und Evaluation und werden als *higher application* bezeichnet. Dies beinhaltet die im Informatikstudium zu fördernde kognitive Fähigkeit zur Problemlösung. [Fuller et al., 2007] identifizierten zwei Studierendengruppen, den praktischen und den theoretischen Studierenden, die unterschiedliche Pfade eines möglichen Kompetenzzugewinnes präferieren. Letztere wählen einen Weg über die Achse Interpretieren bis zur Analyse von Musterlösungen bevor Fachkonzepte erstmalig angewandt werden. Erstere versuchen diese Musterlösungen anzuwenden ohne die Ebene Verstehen erreicht zu haben. Durch Anwenden einer Versuchs- und Irrtumsstragie kann dann Verständnis erworben werden, ebenso aber eine unüberwindbare Lernhürde existieren, wenn aufgrund der Lösungsvielfalt kein geeigneter Lösungsweg gefunden wird. Über die Planung von Lehr-Lernprozessen hinaus kann die Taxonomie eingesetzt werden, um Studierenden verschiedene Lernpfade und Irrwege aufzuzeigen, was einer Diskussion zwischen Lehrenden und Lernenden dienlich ist. Kritisiert haben [Fuller et al., 2007] unter anderem die nicht überlappungsfreien Stufen der Bloom'schen Taxonomie im Bereich schaffender/anwendender Fachdisziplinen. Zwar wurde diese Überlappung weitestgehend durch die zweidimensionale Aufteilung aufgelöst, dennoch bilden die Stufen keine trennscharfe Aufteilung. Die Tätigkeit beziehungsweise das geforderte Kompetenzniveau muss anhand von Beispielen im konkreten

Kontext, also an den Fachkonzepten, betrachtet werden. Hierbei wird deutlich, dass eine Stufe der Taxonomie keinen Hinweis über die notwendige kognitive Leistung zum Erreichen selbiger gibt, da diese immer von Fachkonzept und Studierendentypus abhängig ist.

> „Understand category may require a higher cognitive load than Executing in the Apply category in some contexts“ [Fuller et al., 2007, S. 154].

Aufgrund der höheren Präzision wird trotz der geringeren Verbreitung die Taxonomie gemäß Abbildung 2.1 verwendet, welche die sechs kognitiven Dimensionen nach [Anderson et al., 2000] enthält, allerdings die zweidimensionale Aufteilung des informatikspezifischen *Interpretierens* und *Produzierens* aufweist. [Fuller et al., 2007] illustrieren die Anwendung der Taxonomie mit typischen Tätigkeiten in der Softwareentwicklung, was gegebenenfalls um Tätigkeiten in der technischen Informatik ergänzt werden kann.

Adaptieren Modifizieren einer Lösung für andere Domänen/Bereiche.

Analysieren Testen der (Zeit-)Komplexität einer Lösung.

Anwenden Nutzen einer Lösung als Komponente in einem größeren Problem.

Debug Feststellen und Korrigieren von Fehlern in Designs.

Design Ableiten einer Lösungsstruktur.

Implementieren Codieren einer Lösung bei vollständig gegebenen Designs.

Modellieren Illustrieren oder Erstellen einer Abstraktion einer Lösung.

Präsentieren Erklären einer Lösung.

Erkennen Basiswissen, Vokabular einer Domäne.

Refactor Re-Design einer Lösung (z. B. zur Optimierung).

Verknüpfen Verstehen einer Lösung in anderen Kontexten.

Verfolgen Ablauf verfolgen.

2.2. Entwurf eingebetteter Mikrosysteme

Eingebettete Systeme werden in verschiedensten Bereichen angewandt, wie der Luft- und Raumfahrttechnik, Fahrzeugtechnik, Unterhaltungselektronik und Konsumgütern, Haushaltsgeräten, Gebäudeautomatisierung, Prozess- und Anlagensteuerungen in der Produktionstechnik und vielen weiteren, im Leben alltäglichen Handlungssituationen, in denen diese Systeme dem Benutzer verborgen bleiben (z. B. Steuerung einer Waschmaschine) oder durch ein User-Interface zugänglich

sind (z. B. KFZ-Bordcomputer) (siehe Abbildung 2.2). Unabhängig von ihrem Einsatzgebiet sind eingebettete Systeme durch das Zusammenspiel von Mechanik, Hardware und Software geprägt, welche erst die Vielzahl komplexer Funktionalitäten ermöglicht [Berns et al., 2010].

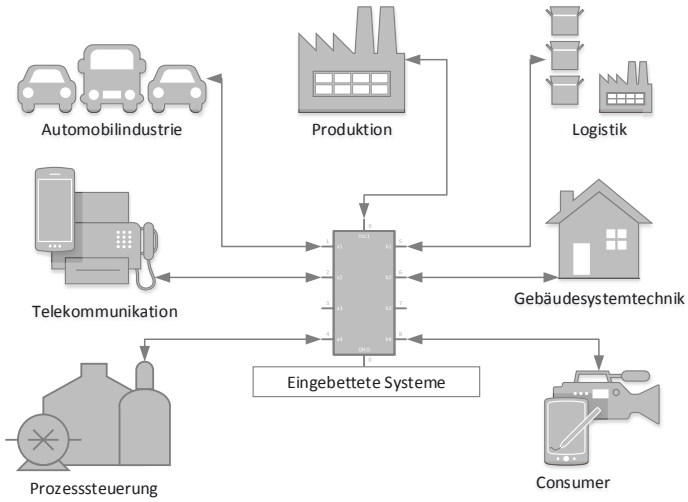


Abbildung 2.2.: Domänenspezifischer Einsatz eingebetteter Systeme

Die hohe Komplexität, die durch dieses Zusammenspiel entsteht, führt zu einem ebenso komplexen Entwurfsprozess eingebetteter (Mikro-)Systeme. Durch ein hierarchisches Entwurfsvorgehen auf verschiedenen Abstraktionsebenen sowie der Aufteilung in Komponenten (Software und Hardware), der Partitionierung, wird dieser beherrschbar. Für die Beschreibung von Entwicklungsvorgehen hat sich das Gajski-Kuhn-Diagramm[1] (siehe Abbildung 2.3) etabliert, welches fünf Stufen der Abstraktion (Systemebene, Algorithmische Ebene, Register-Transfer-Ebene, Logikebene und Schaltkreisebene) in drei Sichtweisen auf Systeme (Verhalten, Struktur und Geometrie) aufweist (vgl. [Bender, 2005]).

[Wagener, 2005] zeigt, dass das von einem Top-Down[2] Entwurf ausgehende Vorgehen im Y-Diagramm nicht geeignet ist beide in der Mikrosystemtechnik vorherrschenden Bereiche, verhaltensnaher und fertigungsnaher Entwurf, abzubilden. Bei der Entwicklung eingebetteter Systeme kommt auch dem Bottom-Up[3] Entwurfsstil

[1] Aufgrund seiner Darstellung in drei Dimensionen, ähnlich einem „Y“ wird das Gajski-Kuhn-Diagramm häufig auch als Y-Diagramm bezeichnet.

[2] Methode, bei der man schrittweise von allgemeinen, umfassenden Strukturen zu immer spezielleren Details übergeht [Bibliographisches Institut GmbH, 2014].

[3] Methode, bei der man von speziellen Details ausgeht und schrittweise über immer umfassendere Strukturen die Gesamtstruktur eines Systems errichtet [Bibliographisches Institut GmbH, 2014].

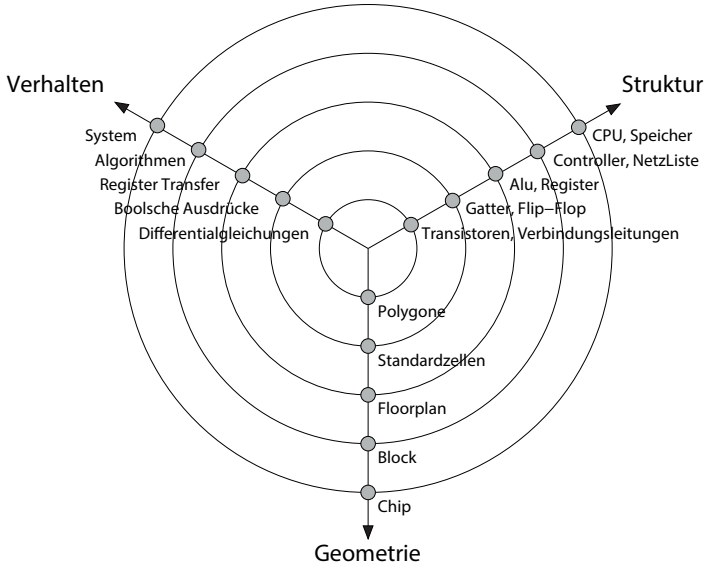


Abbildung 2.3.: Gajski-Kuhn-Diagramm [Gajski und Kuhn, 1983, Walker und Thomas, 1985, Schmidt, 2011]

eine Bedeutung zu, bei dem Komponenten ausgehend von technischen Möglichkeiten entwickelt werden. Im sogenannten Jojo-Vorgehen werden beide verbunden, sodass Bibliotheken aus Basisbauteilen Bottom-Up entwickelt werden und Top-Down zur Verfügung stehen. Als verhaltensnah werden die oberen drei Ebenen in Abbildung 2.4 bezeichnet, bei dem man von einer abstrakten Modellierung des Systems ausgeht und schrittweise zu einem physischen Modell des Mikrosystems synthetisiert (Top-Down) [Schmidt, 2011]. Der fertigungsnahe Entwurf umfasst die Restriktionen an das Mikrosystem, resultierend aus der Fertigungstechnologie. Bei einem ganzheitlichen Systementwurf in der Entwurfspraxis ist davon auszugehen, dass nicht selten aufgrund von Einschränkungen der Fertigungstechnologie ein vollständiger Neuentwurf der funktionalen Baugruppen stattfindet (Bottom-Up) [Schmidt, 2011].

Für die fachdidaktische Diskussion ist also festzuhalten, dass sowohl der fertigungsnahe Bottom-Up Ansatz als auch die verhaltensnahe Modellierung Top-Down in der Entwurfspraxis vorkommen und demnach auch Teil der Hochschulausbildung sein müssen. Durch die Vermischung der Vorgehensweisen je nach Mikrosystem ergeben sich die vier verschiedenen Entwurfsstile (siehe Abbildung 2.4), in denen die Ebenen (*System/Device/Physical*) verhaltensnah und (*Physical/Process*) als fertigungsnah bezeichnet werden – die physikalische Ebene also eine Schnittstelle bildet.

Das *Brezelmodell* illustriert das Zusammenspiel zwischen verhaltensnahem und fertigungsnahem Entwurf und unterscheidet fünf Zustände respektive Entwurfsdokumente [Hahn et al., 2004]. Zielgruppe im Forschungsprojekt des Autors sind Studierende der Informatik (siehe Kapitel 1), sodass der fertigungsnahe Entwurf nicht weiter betrachtet wird, da er als Spezialisierung im Masterstudiengang zu verorten wäre. Dennoch stellt sich die Frage, wieviel Technologiewissen notwendig ist, um auf höherer Abstraktionsebene zu agieren. Insbesondere im Hinblick auf Nanotechnologien hat die Fehlertoleranz Auswirkung auf die oberen Abstraktionsebenen durch die Notwendigkeit redundanter Systemkomponenten.

Die skizzierten Entwurfsstile dienen der Beschreibung von Entwurfsrichtungen innerhalb spezifischer Entwicklungsvorgehensmodelle und -ansätze. Die Wahl eines geeigneten Modells und Prozesses ist von verschiedenen Parametern abhängig und wird implizit im folgenden Abschnitt beschrieben.

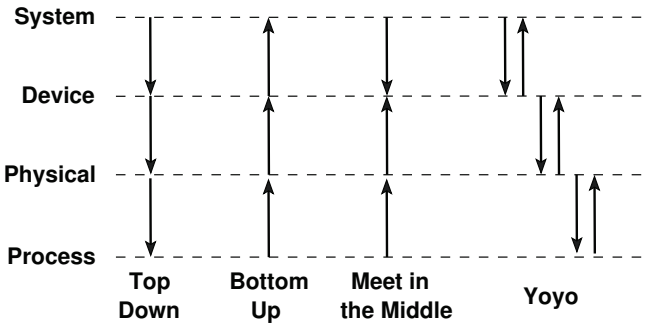


Abbildung 2.4.: Typische Entwurfsmethodiken [Schmidt, 2011, S. 28]

2.3. Vergleichbarkeit fachdidaktischer Publikationen zur Hochschuldidaktik der technischen Informatik

Auf international viel beachteten Fachtagungen und Workshops zur (Hochschul-) Didaktik der Informatik ITiCSE (Innovation and Technology in Computer Science Education), ICER (International Computing Education Research Workshop) und SIGCSE (Special Interest Group on Computer Science Education) der ACM nimmt die Softwareentwicklung eine vorherrschende Rolle ein. Beiträge aus der technischen Informatik betreffen meist die Programmierung von Robotern, seltener auch die Implementierung eingebetteter Systeme mittels Field Programmable Gate Array (FPGA) oder Mikrocontroller (µC). Die ganzheitliche Entwicklung eines eingebetteten Systems wird kaum vorgestellt. Die Konferenzen zur Ingenieursausbil-

dung der IEEE, EDUCON (Global Engineering Education Conference) und TALE (International Conference on Teaching, Assessment and Learning for Engineering) umfassen Beiträge zur technischen Informatik. Der Workshop WESE (Workshop on Embedded Systems Education) wird jährlich im Rahmen der ESWEEK (Embedded Systems Week) veranstaltet und beinhaltet daher ausschließlich Beiträge zu eingebetteten Systemen in der Hochschullehre.

Die Beiträge dieser Veranstaltungen in Form von Praxisberichten oder Forschungsprojekten zur HdTI sind insofern schwierig vergleichbar, als dass sich die Zielgruppe, das interdisziplinäre Umfeld, institutionelle und regionale Besonderheiten sowie die eingesetzte Hard- und Softwareumgebungen stark unterscheiden.

> „Given this fragmentation, the current state of education in embedded systems and software can be rather hard to investigate, since there are so many different actors and departments involved“ [Caspi et al., 2005, S. 591].

Die Unterschiede sind auf Anhieb nicht offensichtlich, da lediglich die betrachtete Abstraktionsebene - bei zunächst identisch wirkenden Inhalt - abweicht. [Caspi et al., 2005] nennen vier Hürden, welche überwunden werden müssen, um zu einem robusten didaktischen Ansatz im Bereich eingebetteter Systeme zu gelangen:

Vielfalt der Herkunft Wie eingangs in Kapitel 1 erläutert ist die Entwicklung eingebetteter Systeme geprägt von Interdisziplinarität, was eine individuelle Sicht auf eingebettete Systeme eines jeden Entwicklers mit sich bringt. Dadurch existieren zahlreiche, an die jeweilige Domäne angepasste Entwicklungsvorgehen und Technologien aus den Ursprüngen der Telekommunikation, Mechanik, Automobilindustrie, Luft- und Raumfahrt, Unterhaltungselektronik etc. Die Vorstellungen der Studierenden von eingebetteten Systemen hängen demnach von ihrem originären Anwendungsfach ab.

Vielfalt der Kulturen Die Unterschiede in der Herkunft – also den verschiedenen Domänen – führten zu zahlreichen Kulturen, was mit der Notation von Zeit exemplarisch gezeigt wird [Caspi et al., 2005]. Ein Ingenieur der Regelungstechnik oder Mechanik betrachtet Zeit als kontinuierliche Einheit auf einer Zeitachse. In der Informatik wird Zeit je nach Sicht als diskret (Sampling) oder als Folge asynchroner Ereignisse (Telekommunikation) betrachtet. Solche kulturellen Unterschiede führen in der Praxis zu Kommunikationsproblemen, wovon in dieser Arbeit jedoch nicht auszugehen ist. So sind die Studierenden der Zielgruppe (noch) nicht von ihrer jeweiligen Kultur geprägt und darüber hinaus werden keine Lehr-Lernprozesse betrachtet, welche potentiell zu unterschiedlichen Auffassungen führen können.

Vielfalt der Verfahren Die Entwicklung eingebetteter Systeme bietet eine sehr große Lösungsvielfalt, wie die Wahl zwischen einer (Teil-)Implementierung in Hardware oder Software, der Wahl zwischen verschiedenen Modellierungstechniken (z. B. *Matlab/Simulink/Stateflow* oder *UML*) und der Implemen-

tierung (z. B. in C oder Assembler) sowie der verwendeten Hardware (z. B. *FPGA/µC*), was demzufolge auch den gesamten Entwicklungsprozess bestimmt.

Vielfalt der Ausbildungen Das Problem bei der Vielzahl an Entwicklungsmethodiken und Entwicklungsumgebungen, Technologien sowie Hardwareplattformen liegt darin, dass es den Studierenden nicht möglich ist die Breite des Gebietes zu verstehen oder anzuwenden. Auch hier ist entscheidend welcher Kultur die Studierenden und Lehrenden entstammen, denn aus den jeweiligen Kulturen und Domänen hat sich die Vielfalt der Verfahren historisch entwickelt. Resultierend aus den ersten drei Punkten besitzt folglich jede der Kulturen auch ihre eigene Ausbildungskultur, in der ein eingeschränkter Blick auf eingebettete Systeme vorherrscht und damit eher domänenspezifische Techniken fokussiert werden, als eine Wissenschaft für sich zu bilden [Caspi et al., 2005].

Um die Vergleichbarkeit zwischen den Publikationen zu vielfältigen Ausbildungsaspekten herzustellen, wurde eine Taxonomie zur Vergleichbarkeit entwickelt. [Büchner et al., 2013] verglichen drei Publikationen zu didaktischen Konzepten im Bereich *Hardware/Software Codesign* miteinander, um ein gemeinsames Klassifikationsschema für hochschuldidaktische Publikationen zur technischen Informatik abzuleiten.

- [Mitsui et al., 2009] untersuchten verschiedene Hard- und Softwaredesign Experimente, darunter eins, welches Hardware/Software Codesign betrifft. In diesem Projekt sollte die Funktionalität eines JPEG[4] Encoders in Hardware auf einem FPGA implementiert werden.
- [Kassner und Ricks, 2005] evaluierten eine Projektveranstaltung im Bachelorstudiengang mit dem Thema Hardware/Software Codesign. Entwickelt werden sollte ein Spektrumanalysator[5] mit Echtzeitcharakteristik.
- [Schaumont, 2008] schlugen vor, Hardware/Software Codesign als Einstieg in den Bereich eingebetteter Systeme zu nutzen und verwendeten C und Very High Speed Integrated Circuit Hardware Description Language (VHDL) in einer gemeinsamen Entwicklungsumgebung.

Alle beschriebenen Lehrveranstaltungen führen in Konzepte des Hardware/Software Codesigns im Bachelorstudium ein. Dennoch ergab die Analyse in [Büchner et al., 2013] deutliche Unterschiede, sodass einerseits gezeigt werden konnte, dass ein Bedarf an einer Taxonomie besteht, welche eine schnelle Prüfbarkeit der Relevanz und Übertragbarkeit der Arbeiten für die eigene Forschung ermöglicht und andererseits dem Leser dieser Arbeit eine Orientierungshilfe bietet, um bei der Beschreibung von Aufgaben den Kontext und die Intention des Lehrenden fass-

[4] Standard zur Bildkompression, Norm ISO/IEC 10918-1.
[5] Ein elektronisches Messgerät zur Darstellung der Frequenzen eines Signals.

bar zu machen. Es stellt sich bei der Beschreibung von Lehr-Lernprozessen beziehungsweise gegebenen Problemstellungen immer die Frage nach dem zu erreichenden Ziel unter der Verwendung bestimmter Vorgaben, also der Aufgabenstellung, Voraussetzungen der Studierenden und lernunterstützenden Hilfsmitteln. Da diese Aspekte von verschiedenen Faktoren abhängig sind, ist auch die Beschreibung einer Problemstellung mehrdimensional. In Abbildung 2.5 sind die vier Dimensionen *Abstraktion*, *Komplexität*, *Profil* und *Kognitivität* dargestellt. Für die Beschreibung der kognitiven Dimension, also Anforderung an die Studierenden, wurde die Taxonomie nach [Anderson et al., 2000] gewählt – 1. Erinnern, 2. Verstehen, 3. Anwenden, 4. Analysieren, 5. Bewerten, 6. Gestalten (vgl. Abschnitt 2.1).

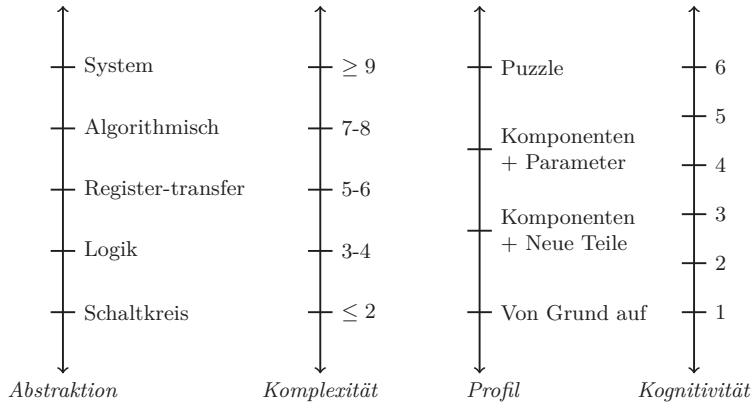


Abbildung 2.5.: Taxonomie – Vergleichbarkeit didaktischer Beiträge zur HdTI [Büchner et al., 2013, S. 2]

Obgleich sich das Y-Diagramm (vgl. Abschnitt 2.2) aufgrund seiner ursprünglichen Top-Down Sicht auf den Systementwurf nicht eignet, einen gemischt verhaltens- und fertigungsnahen Entwurfsstil abzubilden, so kann dennoch eine hinreichende Beschreibung der Abstraktionsstufen in Aufgaben gegeben werden. Es ist davon auszugehen, dass sehr wenige Publikationen ein Entwicklungsprojekt beschreiben, welches sich über alle Ebenen des Entwurfs inklusive der Fertigung eines Systems erstreckt, sodass Aufgrund der Freiheit und insbesondere der Einfachheit des Y-Diagrammes eine schnelle Kategorisierung von Aufgabenstellungen ermöglicht wird.

Abstraktionslevel

Der Abstraktionsgrad hängt von der zu verwendenden Sprache und Entwicklungsumgebung ab. In auf die Programmierung von eingebetteten Systemen beschränkten Lehrveranstaltungen werden die hardwarenahen und auf der physikalischen sowie strukturellen Dimension liegenden Konzepte von untergeordneter Relevanz

sein, wohingegen das Verhalten des Systems durch die Programmierung bestimmt wird. Die Abstraktionsebene wird nun durch die zu verwendende Umgebung/ Sprache bestimmt, welche unter Umständen automatisch in ein Hardwarelayout synthetisiert wird. Wird beispielsweise ein System mit VHDL modelliert und vollständig synthetisiert, ist - auch wenn das System später in Hardware gefertigt wird - die Abstraktionsdimension Register-Transfer-Ebene zu wählen. Auf jeder Sicht auf das System (Verhalten, Struktur, Geometrie) kommen dabei verschiedene Beschreibungsmittel zum Einsatz und werden im Folgenden exemplarisch benannt (vgl. [Bender, 2005]).

Systemebene Die Systemebene umfasst Blöcke wie Speicher, Prozessoren und Schnittstellen, meist in natürlicher Sprache und Skizzen.

Algorithmische Ebene Auf der algorithmischen Ebene werden nebenläufige Algorithmen mit Prozeduren, Funktionen, Prozesse, Kontrollstrukturen, Signalverknüpfungen, Variablen und Operatoren beschrieben.

Register-Transfer-Ebene Diese Ebene spezifiziert die Eigenschaften durch Operationen (z.B. Addition), den Transfer zwischen Registern, der Elemente wie Register, Codierer, Multiplexer oder Addierer sowie endlichen Automaten und eine Einteilung der Chipfläche.

Logikebene Die Logikebene umfasst logische Verknüpfungen und deren zeitliche Eigenschaften, zusammengeschalteter Grundelemente wie AND-, OR-, XOR-Gatter oder Flip-Flops, Boolesche Gleichungen oder Funktionstabellen.

Schaltkreisebene Sie enthält die Netzliste mit den Bauteilen wie Transistoren, Kapazitäten und Widerstände. Außerdem umfasst sie Differentialgleichungen und beispielsweise die Dotierung auf einem Chip sowie Polygonzüge.

Komplexitätslevel

Der Begriff Komplexität wird in diesem Zusammenhang als Menge und Relation von Anforderungen an ein System gesehen. Statt einer Modellierung dieser Anforderungen wurde eine Methode angewandt, um die Komplexität grob zu erfassen. Ein Katalog von Systemeigenschaften, welche bekanntermaßen als komplex gelten, ist gegeben [Büchner et al., 2013]:

1. Regeln/Steuern (nicht nur Erfassen) von Umgebungseigenschaften,
2. physikalische, biologische und chemische Parameter,
3. sehr großer Zustandsraum oder hohe Anzahl von Teilen,
4. maximale Leistung oder Optimierung,
5. Echtzeitfähigkeit,
6. Parallelität,

7. Implementierung heterogener Komponenten,
8. Erfüllung eines Standards,
9. starke ergonomische oder nicht-funktionale Anforderungen.

Offensichtlich genügt dieser Katalog weder den Anforderungen an Industrieprojekte oder einer Kostenabschätzung, noch ist er vollständig. Für die Klassifizierung von didaktischen Publikationen gibt die Anzahl der Kriterien aber eine hinreichende Beschreibung, von der Entwicklung eines autonomen Roboters mit Echtzeitauswertung der mit Laserscannern erfassten dreidimensionalen Umwelt (Klasse 9), bis zur Anschaltung einer Leuchtdiode (LED) mittels Mikrocontroller (Klasse 1). Bei der Analyse einer Publikation wird das zu entwickelnde System hinsichtlich der Punkte des Katalogs analysiert. Je zutreffenden Eigenschaft wird der Grad der Komplexität inkrementiert (vgl. [Büchner und Jaschke, 2013]).

Beispiele

Komplexität 1: Das Ansteuern einer LED mittels Mikrocontroller. Auf dieses System trifft lediglich Punkt eins zu, das Ansteuern eines Aktors.

Komplexität 4: Die Steuerung einer Abfüllanlage, welche vernetzte Subsysteme (Aktoren/Sensoren) via Mikrocontroller steuert. In diesem Beispiel arbeiten mehrere Systeme Parallel (6). Jedes Teilsystem steuert Aktoren an (1). Verschiedenste Aktoren und Sensoren stellen dabei heterogene Komponenten dar (7). Zeitschranken müssen für einen reibungslosen Betrieb eingehalten werden (5).

Komplexität 9: Die Entwicklung eines autonomen Roboters inklusive Navigation, Sensorik, Aktorik, Echtzeitauswertung von Bildern, Notfallprogrammen etc. Dieses Beispiel beinhaltet nahezu alle Punkte des Kataloges und wird im Sinne einer Aufgabe für die Lehre zu eingebetteten Systemen als maximal komplex angesehen, woraus folgt, dass eine solche Aufgabe nicht für Studierende in den ersten Semestern geeignet ist.

Aufgabenprofillevel

Ferner wurden Aufgabenprofile in die Taxonomie aufgenommen, die eine Einschätzung des Umfangs einer Aufgabe ermöglichen. Aufgaben, welche zunächst identisch wirken (z. B. aufgrund der verwendeten Programmiersprache), unterscheiden sich teils deutlich, je nach dem Umfang des bereits vorgegebenen Teils. Vier Stufen werden unterschieden:

- Die Entwicklung eines Systems von Grund auf (ohne existierende Subsysteme, keine Vorarbeiten). Keine Unterstützung durch automatisch generierte Strukturen von Quelltextbausteinen (sog. Skeletons) oder Bibliotheken.
- Entwicklung eines Systems unter Verwendung bereits existierender und komplett neuer Teile/Funktionalitäten.

- Schaffung neuer Teile durch die Verbindung existierender Subsysteme und ihrer Parametrisierung zur Erfüllung bestimmter Aufgaben (Anpassungen).
- Verbindung von existierenden Subsystemen ohne jegliche Anpassung der Teile/Funktionalitäten (Puzzle).

Beispiel

Analog-/Digital Wandler (Analog-Digital Converter, ADC)
Auf der ersten Stufe existiert noch kein System und folglich noch keine Restriktionen für den Einsatz oder gar den Entwurf eines Analog/Digital-Wandlers. Stufe zwei erfordert den Einsatz/Entwurf eines Analog/Digital-Wandlers für ein bestehendes System, in dem z. B. die Schnittstellen zum Analog/Digital-Wandler bereits definiert sind. Auf Stufe drei ist der zu verwendende Analog/Digital-Wandler vorgegeben und muss entsprechend des Systemeinsatzes parametrisiert werden (klassischer Mikrocontroller mit integriertem Analog/Digital-Wandler) [Büchner et al., 2013, Jaschke, 2013].

Im Unterschied zu anderen Ansätzen (z. B. [Bower, 2008]) wird bei der Taxonomie nach [Büchner und Jaschke, 2013] zwischen verschiedenen Tätigkeiten auf verschiedenen kognitiven Niveaustufen innerhalb einer Aufgabenstellung differenziert, was dazu führt, dass diese Beschreibung nicht eindimensional ist und damit die Unterscheidung zwischen Aufgabenprofil und kognitiver Stufe begründet.

Fazit

Die Taxonomie nach [Büchner et al., 2013] ermöglicht eine Analyse fachdidaktischer Publikationen anhand wesentlicher Merkmale von Lehr-Lernprozessen mit eingebetteten Systemen in vier Dimensionen – Abstraktion, Komplexität, Profil und Kognitivität. Letztere wurde zuvor durch die Taxonomie nach [Fuller et al., 2007] beschrieben. Dem Anspruch, eine simple ad hoc Bewertung fachdidaktischer Artikel und Praxisberichte zu ermöglichen, wird diese jedoch nicht gerecht, da eine weitere Dimension, welche unter Umständen nicht intuitiv greifbar ist, existiert. Daher wurde in [Büchner et al., 2013] die eindimensionale Taxonomie nach [Anderson et al., 2000] verwendet. Dieser Anspruch existiert indes in dieser Arbeit nicht, sodass eine Erweiterung zur Präzisierung kognitiver Stufen und Pfaden der Kompetenzaneignung vorgenommen wird (vgl. Abschnitt 3.4).

2.4. Didaktik der technischen Informatik – Ziele und Empfehlungen

> „Der Hochschulsektor repräsentiert in Deutschland einen Bildungsbereich, der trotz zunehmender gesellschaftlicher Bedeutung in der nationalen und internationalen empirischen Bildungsforschung bislang nur wenig Aufmerk-

samkeit erfahren hat [Blömeke und Zlatkin-Troitschanskaia, 2013, S. 2]."

Was in diesem Zitat für den Hochschulsektor im Allgemeinen attestiert wurde gilt auch für die technische Informatik im Speziellen, welche bislang nicht im Fokus hochschuldidaktischer Forschung stand, obgleich affine Themengebiete derzeit den Einzug im Unterricht an allgemeinbildenden Schulen finden. Dazu gehören insbesondere programmierbare Roboter. Forschungsarbeiten zur Didaktik der Informatik an allgemeinbildenden Schulen, wie auch zur Hochschulinformatik, haben meist keinen Bezug zur technischen Informatik und beschränken sich überwiegend auf Softwareentwicklung, Netzwerktechnik, theoretische und praktische Informatik.

Ein Grund für den geringen Anteil an Forschungsarbeiten zur HdTI ist, dass aufgrund der kulturspezifischen Sichtweisen, geringen Studierendenzahlen sowie ungleicher Module und Studiengänge empirische Forschung nur schwerlich betrieben werden kann. Zur Sicherung qualifizierten Nachwuchses in der Informatik publizierten nationale und internationale Branchenverbände diverse Empfehlungen zu Informatikcurricula im Hochschulsektor. Am umfangreichsten sind die Arbeiten der ACM/IEEE, welche im *Overview Report* strukturiert werden. Dieser gibt einen Überblick über verschiedene Studienprogramme der Informatik und soll Lehrenden, institutionellen Administratoren und Studierenden als Orientierungshilfe dienen, da sich das breite Gebiet über Mathematik, Informatik als Wissenschaft, Ingenieurwesen, Betriebswirtschaft etc. erstreckt [ACM/IEEE, 2005].

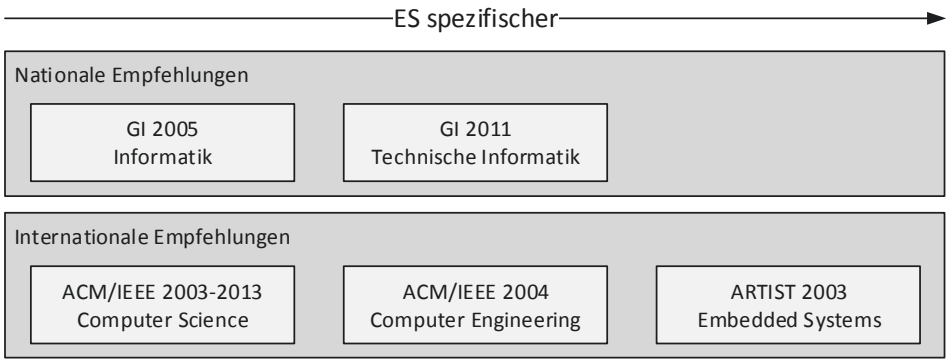


Abbildung 2.6.: Übersicht der relevanten Empfehlungen

Abbildung 2.6 zeigt die für diese Arbeit relevanten nationalen und internationalen Empfehlungen vom allgemeinen Informatikstudium hin zu Empfehlungen zu Studiengängen mit Fokus auf eingebettete Systeme. In KOMINA wurden die Empfehlungen [ACM/IEEE, 2008] sowie [GI, 2011] analysiert und bildeten die Grundlage für die Entwicklung eines Kompetenzstrukturmodells. In den darauf aufbauenden Arbeiten des Autors dienten zusätzlich die spezifischen Empfehlungen [ACM/IEEE, 2004a] und [Artist Education Group, 2003] als Grundlage für die Identifikation

von Vorwissensrelationen. Als nationales Pendant zum Computer Science Curriculum der ACM/IEEE wurde die Empfehlung der Gesellschaft für Informatik (GI) für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen analysiert. Inzwischen hat die ACM und IEEE das Computer Science Curriculum überarbeitet und in der Version 2013 veröffentlicht. Die Analyse innerhalb dieser Arbeit umfasst folgende Punkte:

Einordnung in den Kontext Die Empfehlungen sind an unterschiedliche Studierendengruppen ausgerichtet und enthalten verschiedene Vorgehensweisen, Inhaltsbereiche und Studienverlaufsempfehlungen.

Bestimmung der Granularität/Taxonomie Die Granularität der benannten Inhaltsbereiche sowie die stringente Verwendung von Operatoren einer Taxonomie ist entscheidend für die Verwertbarkeit innerhalb eines Kompetenzmodells und der Ableitung konkreter Lehrveranstaltungen.

Identifikation von impliziten/expliziten Vorwissensrelationen Vorwissensrelationen sind gegeben, wenn Module unterschiedlicher Inhaltsbereiche in Studienverlaufsempfehlungen aufeinander aufbauen.

Verortung von Modulen/Konzepten der technischen Informatik Es wird analysiert, in welchem Umfang und in welcher zeitlicher Abfolge TI-affine Module/Themen/Kurse in den Empfehlungen verortet sind, um zu einer übereinkommenden Vorstellung von zu erlangenden Kompetenzen auf bestimmten Niveaustufen zu gelangen, was auch Laborveranstaltungen (Projekte) umfasst.

2.4.1. GI: Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen (2005)

Die Gesellschaft für Informatik entwickelte in Zusammenarbeit mit dem Fakultätentag- und dem Fachbereichstag[6] Informatik eine Empfehlung zu Ausbildungszielen, organisatorischen und strukturellen Anforderungen sowie Kompetenzfeldern in Bachelorstudiengängen Informatik [GI, 2005]. Eine Unterscheidung von Studiengängen an Fachhochschulen und denen an Universitäten wird von der KMK durch die Reform im Bologna-Prozess[7] nicht vorgenommen, wenngleich eine Differenzierung zwischen grundlagen- und anwendungsorientiertem Studium beziehungsweise Forschung innerhalb der GI-Empfehlung stattfindet [NRW, 2014]. Um den Zugang zu einem konsekutiven Masterstudium zu ermöglichen, empfiehlt die GI das Studienprogramm entsprechend grundlagenorientiert zu gestalten. Ohne Anspruch auf Vollständigkeit gliedert die GI Kompetenzen in sechs Kompetenzfelder [GI, 2005] und beschreibt zusätzlich deren wichtigste Teilkompetenzen:

- Formale, algorithmische, mathematische Kompetenzen,

[6] Der Fachbereichstag vertritt seine Mitglieder in allen Aspekten des Informatikstudiums an Hochschulen für Angewandte Wissenschaften/Fachhochschulen.

[7] Ziel des Bologna-Prozesses ist die Errichtung des Europäischen Hochschulraums.

- Analyse-, Design-, Realisierungs- und Projekt-Management-Kompetenzen,
- technologische Kompetenzen,
- fachübergreifende Kompetenzen,
- Methodenkompetenzen,
- soziale Kompetenzen und Selbstkompetenz.

Die Systemhardware wird in den Beschreibungen der Design-Kompetenzen und technologischen Kompetenzen explizit aufgeführt.

> „Design-Kompetenzen umfassen die Fähigkeit zur Konstruktion von Systemen aus Hard- und Software, welche die Anforderungen vollständig erfüllen. [...] Diese umfassen ein breites und sehr unterschiedliches Spektrum von Fachkenntnissen. Informatikerinnen und Informatiker müssen Architektur, Konzepte und Funktionsweise moderner Betriebssysteme ebenso verstehen wie das Zusammenspiel von Hard- und Software. [...] Im Bereich Echtzeitsysteme müssen sie ein Verständnis der Hard- und Software-Konzepte für die Wechselwirkung eines Rechners mit seiner Umgebung haben und Kenntnisse über nebenläufige Systeme und ihre systemnahe Implementierung besitzen“ [GI, 2005, S. 8 ff].

In den weiteren Kompetenzfeldern ist die technische Informatik implizit in der Nutzung von Entwicklungsumgebungen, Algorithmen, Automaten, formalen Sprachen etc. enthalten. Teilkompetenzen wurden mit Operatoren der Taxonomie nach [Anderson et al., 2000] beschrieben, wenngleich diese zu grobgranular für die Ableitung konkreter Lehrveranstaltungen ist.

Im Hinblick auf die Zielgruppe im Forschungsprojekt des Autors sind zwei der drei betrachteten Studiengangstypen relevant - Studierende in einem Studiengang Informatik ohne Anwendungsfach (Typ 1) und mit Anwendungsfach, z. B. Elektrotechnik (Typ 2) – letzterer insbesondere. Typ 3 bilden interdisziplinäre Studiengänge mit einem Informatikanteil von $\leq 50\%$. Im exemplarischen Musterplan II des grundlagenorientierten Studiengangs vom Typ 1 - Studiengang der Informatik ohne Anwendungsfach - sind Module der technischen Informatik verortet (siehe Tabelle 2.1). Es wird klar, dass die Systemhardware und die technische Informatik einen großen Stellenwert in allen Studiengängen der Informatik haben und nicht Studiengängen mit Anwendungsfach Elektrotechnik vorbehalten sind.

Für die Studierenden der Zielgruppe, die auch nicht TI-affine Anwendungsfächer studieren (Typ 2 ohne Elektrotechnik), sind, verglichen mit einem Studiengang nach Typ 1, 10-35 ECTS[8] der informatikspezifischen Studieninhalte sowie bis zu 17 ECTS in den *Mathematisch-Naturwissenschaftlich-Technischen Grundlagen* weniger zu belegen. Je nach Umsetzung und Profilbildung der Hochschule stehen also

[8] Leistungspunkte nach dem *European Credit Transfer and Accumulation System*.

Tabelle 2.1.: Module der technischen Informatik (vgl. [GI, 2005])

TI-affine Module	**Umfang**
Rechnerarchitektur	5 ECTS
Hardwarepraktikum	5 ECTS
Systemsoftware	5 ECTS
Eingebettete Systeme Wahlpflicht	6 ECTS
Grundlagen der Technischen Informatik	5 ECTS

zwischen 10 und 45 ECTS weniger Zeit für ein grundlagenorientiertes Studium mit TI-Affinität zur Verfügung. Eine Empfehlung bezüglich der Umsetzung auf Modulebene sowie die zu verwendenden Technologien wird nicht gegeben.

2.4.2. ACM/IEEE: Computer Science Curriculum (2001-2013)

Die Empfehlungen der gemeinsamen Arbeitsgruppe der ACM und IEEE haben einen großen Einfluss auf die Gestaltung von Curricula und die internationale informatikdidaktische Forschung. Für normative Studien des Autors sowie im Projekt KOMINA diente das *Computer Science Curriculum 2008* [ACM/IEEE, 2008] als Quelle, welches eine Übergangsempfehlung zwischen dem im Jahr 2001 veröffentlichten *Computing Curricula 2001 – Computer Science* [ACM/IEEE, 2001] und dem 2013 veröffentlichten *Computer Science Curricula 2013* [ACM/IEEE, 2013] darstellt.

Begründet wird die Notwendigkeit für eine Zwischenversion unter anderem mit der sich ändernden beruflichen Praxis der Absolventen sowie dadurch benötigten Fähigkeiten und Fertigkeiten. Des Weiteren bedinge das volatile Fach eine ständige Anpassung der Inhaltsbasis, welche in 14 Bereiche gegliedert ist [ACM/IEEE, 2008]. Innerhalb der Bereiche intendierte Kompetenzen werden mit Operatoren nach [Anderson et al., 2000] beschrieben.

> „1. Describe what makes a system a real-time system.
> 2. Explain the presence of and describe the characteristics of latency in real-time systems.
> 3. Summarize special concerns that real-time systems present and how these concerns are addressed“ [ACM/IEEE, 2008, S. 57].

Einen eigenen Inhaltsbereich zu eingebetteten Systemen gibt es nicht. Es werden allerdings einige Themen der technischen Informatik und eingebetteten Systemen aufgeführt sowie eine Empfehlung zum Umfang in Informatikcurricula gegeben (siehe Tabelle 2.2). Darin enthalten ist die empfohlene Mindestanzahl an Vorlesungsstunden (280 Stunden), welche je nach Ausrichtung des Studienprogrammes erhöht werden können.

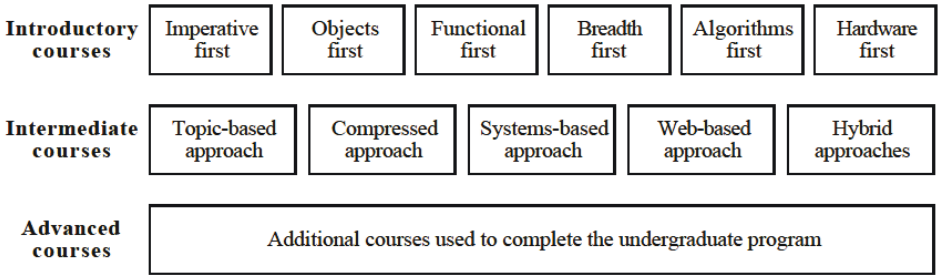


Abbildung 2.7.: Struktur der Empfehlungen und Ansätze [ACM/IEEE, 2001, S. 18]

In [ACM/IEEE, 2001] werden die Inhaltsbereiche Kursen zugeordnet, um eine direkt implementierbare Vorlage zu bieten, welche dann der institutionellen Curriculumentwicklung zugeführt werden kann. In der Kurzbeschreibung sind die Stundenanzahl und eine Zusammenfassung der Themen enthalten. Des Weiteren werden grobe Vorwissensrelationen durch die Angabe von Kursen als Voraussetzung gegeben, sowie verschiedene Ansätze für den Einstieg ins Studium, mittleres Niveau und Fortgeschrittene vorgestellt. Darunter auch ein Bottom-Up Ansatz, in dem die Informatik von der Hardware der Maschine aus betrachtet wird (*hardware first*, siehe Abbildung 2.7). Auf mittlerem Niveau wird ein technischer Verlauf, welcher die Entwicklung von Computersystemen fokussiert, beschrieben (*systems-based approach*). Zwar werden auf verschiedenen Ebenen die Einbindung von Labor- oder Projektmodulen vorgeschlagen, jedoch fehlt eine konkrete Empfehlung für ein hardwareorientiertes Praktikum respektive Labor.

> „Computing professionals should not regard the computer as just a black box that executes programs by magic. The knowledge area Architecture and Organization builds on Systems Fundamentals (SF) to develop a deeper understanding of the hardware environment upon which all computing is based, and the interface it provides to higher software layers“ [ACM/IEEE, 2013].

In der Überarbeitung des Curriculums [ACM/IEEE, 2013] wird eine weitere Hierarchieebene zu *core* und *elective* eingeführt deren Inhaltsbereiche mindestens zu 80 % in Informatikcurricula abgedeckt werden sollen. Insbesondere wird ein Wahlbereich (elective) gefordert, der den Studierenden eine Spezialisierung ermöglicht. Auf den ersten Blick scheint eine Reduzierung des Umfangs in den Bereichen Hardware, Architektur und Organisation zu erfolgen (von 31 auf 16 Stunden). Durch die Einführung eines weiteren Inhaltsbereichs (*systems fundamentals*) wird allerdings der Abbau einer Vorstellung des Computersystems als Black-Box hervorgehoben.

Tabelle 2.2.: Inhaltsbereiche – Computer Science Curriculum [ACM/IEEE, 2008]

Inhaltsbereich	**Stunden**
Discrete Structures	43
BasicLogic	10
Programming Fundamentals	47
EventDriveProgramming	6
Algorithms and Complexity	31
AutomataTheory	0
Architecture and Organization	36
DigitalLogicAndDataRepresentation	7
ComputerArchitectureAndOrganization	9
InterfacingAndI/OStrategies	3
Multiprocessing	0
PerformanceEnhancements	0
DistributedArchitectures	0
Devices	0
Operating Systems	18
RealTimeAndEmbeddedSystems	0
FaultTolerance	0
Net-Centric Computing	15
NetworkCommunication	7
MobileComputing	0
Programming Languages	21
Human-Computer Interaction	8
Graphics and Visual Computing	3
Intelligent Systems	10
Robotics	0
Information Management	11
Social and Professional Issues	16
Software Engineering	31
Computational Science	0
ModellingAndSimulation	0

2.4.3. ACM/IEEE: Computer Engineering (2004)

Die *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering* sind eine Fortführung der Arbeiten des IEEE, der ACM und der Association for Information Systems (AIS) (vgl. [ACM/IEEE, 2001, ACM/AIS/AITP, 2002, ACM/IEEE, 2004b]). Die Empfehlungen liefern Hintergrundinformationen zum Berufsfeld von Absolventen der technischen Informatik. Dabei wird die Relevanz von Kompetenzen für das Design, die Implementierung, die Konstruktion und Wartung von Systemen explizit hervorgehoben.

„Computer engineering is defined as the discipline that embodies the science

> and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment. Computer engineering has traditionally been viewed as a combination of both computer science (CS) and electrical engineering (EE)“ [ACM/IEEE, 2004a].

Der Fokus soll dabei auf dem Design von Hard- und Softwaresystemen liegen und damit auf einer höheren kognitiven Stufe als deren Aufbau und Konfiguration (siehe Abschnitt 2.1). Es umfasst neben dem Treffen von Entwurfsentscheidungen auch das Eingehen von Kompromissen bei gegebenen Randbedingungen.

Vorschläge für Veranstaltungsinhalte in einem Curriculum zur technischen Informatik werden analog zu den Empfehlungen in Abschnitt 2.4.2 in Inhaltsbereichen gegeben. Da hier das Gros der Themen TI-affin ist, wird auf eine explizite Aufführung verzichtet und Tabelle 2.3 auf die Oberkategorien beschränkt.

Tabelle 2.3.: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering [ACM/IEEE, 2004a]

Inhaltsbereich		**Stunden**
Computer Engineering		
	Algorithms	30
	Computer Architecture and Organization	63
	Computer Systems Engineering	18
	Circuits and Signals	43
	Database Systems	15
	Digital Logic	57
	Digital Signal Processing	17
	Electronics	40
	Embedded Systems	20
	Human-Computer Interaction	8
	Computer Networks	21
	Operating Systems	20
	Programming Fundamentals	39
	Social and Professional Issues	16
	Software Engineering	13
	VLSI Design and Fabrication	10
Mathematics		
	Discrete Structures	33
	Probability and Statistics	33

In der Empfehlung wird erneut die Schwierigkeit der Abgrenzung von mit eingebetteten Systemen verwandten Themengebieten erkennbar. Auch bei unterschiedlicher Benennung der Kurse überlappen die behandelten Themen, teilweise sind sie identisch. *Computer engineering* und *digital system design* sind Beispiele für

weitere, anders bezeichnete aber dennoch sehr stark mit eingebetteten Systemen verzahnte, Einheiten [ACM/IEEE, 2004a]. Laborerfahrungen werden als essentieller Teil des Curriculums gesehen und sollen den Studierenden die Möglichkeit bieten Geräte, Systeme und Prozesse zu beobachten, zu erkunden und zu verändern.

> „Laboratories should include some physical implementation of designs such as electronic and digital circuits, bread-boarding, microprocessor interfacing, prototyping, and implementation of hardware and software“ [ACM/IEEE, 2004a].

2.4.4. GI: Curriculum Technische Informatik in Bachelor- und Masterstudiengängen Informatik (2011)

Die Empfehlung der GI beinhaltet eine Strukturierung von Inhalten zusammengehöriger Themengebiete für Bachelor- und Masterstudiengänge der Informatik (Typ 1, Typ 2 und Typ 3, vgl. Abschnitt 2.4.1) sowie Lernziele ohne Gewichtung des Studienumfangs, die außerdem den Schwerpunkt in Studiengängen der technischen Informatik bilden sollen. Die Empfehlung beinhaltet also das Fundament einer technischen Informatikausbildung, die unabhängig vom konkreten Studiengang berücksichtigt werden soll.

Die Formulierung der Kompetenzen erfolgt nicht analog zu bisherigen Empfehlungen nach einer Lernzieltaxonomie, was eine Interpretation der Intentionen der Autoren notwendig macht.

> „Fähigkeiten der formalen und programmiersprachlichen Schaltungsbeschreibung, [...]“ [GI, 2011, S. 4].

Hier wird nicht klar, welche Fähigkeiten Studierende konkret erlangen sollen. Daraus könnte folgen, dass Studierende die Fähigkeit besitzen sollen, formale Beschreibungen von Schaltungen zu interpretieren. Aber auch eine selbständige Beschreibung von Schaltungen könnte gefordert sein. Es handelt sich dabei also nicht um eine Kompetenzbeschreibung im Sinne der in dieser Arbeit verwendeten Definition (vgl. Abschnitt 2.1). Auf der anderen Seite existieren eindeutige Beschreibungen durch die Nutzung operationalisierter Verben nach [Anderson et al., 2000].

> „[...] die Fähigkeit, unbekannte Schaltungen zu analysieren und zu verstehen, sowie eigene Schaltungen zu entwickeln, [...]“ [GI, 2011, S. 4].

Die Analyse von unbekannten Schaltungen stellt dabei eine Lösung in variablen Situationen dar. Zu jedem Inhaltsbereich der Empfehlung werden mögliche Übungen zum Erreichen der Lernziele angedeutet, jedoch keine konkrete Aufgabensammlung gestellt. Zu den Themengebieten Digitaltechnik und Rechnerorganisation werden

explizit Laborübungen zur Integration in Übungen beziehungsweise als eigenständiges Praktikum in Vertiefungsveranstaltungen empfohlen.

Tabelle 2.4.: Themengebiete – Curriculum Technische Informatik in Bachelor- und Masterstudiengängen Informatik [GI, 2011]

Pflichtbereich	**Wahlbereich**
Digitaltechnik	Parallelrechner/Parallelverarbeitung
Rechnerorganisation	Robotik
Laborübungen zu Digitaltechnik und Rechnerorganisation	Quantitative Modellbildung und -analyse
Betriebssysteme	Test von Hardwareschaltungen
Rechnernetze	Programmierung Eingebetteter Systeme
Rechnerarchitektur	Rechnertechnologie
Eingebettete Systeme	Automatisierung technischer Prozesse
	Rechnergestützter Schaltungsentwurf
	Technische Aspekte der IT-Sicherheit
	...

2.4.5. ARTIST: Guidelines for a graduate curriculum on embedded software and systems (2005)

Die *Guidelines for a Graduate Curriculum on Embedded Software and Systems* wurde in den Jahren 2002 und 2003 von Forschern zu eingebetteten Echtzeitsystemen entwickelt und 2005 vom *ARTIST European Network of Excellence on Embedded Systems Design* publiziert (vgl. [Artist Education Group, 2003], [Caspi et al., 2005]).

Die Empfehlung soll die Ausbildung von Studierenden mit Expertise in den Domänen Regelungstechnik, Informatik und Elektrotechnik unterstützen, da diese in allen drei Bereichen notwendig sei, um eingebettete Echtzeitsysteme entwickeln zu können [Artist Education Group, 2003]. Verglichen mit den zuvor betrachteten Empfehlungen erfolgt eine Spezialisierung auf die Entwicklung eingebetteter Systeme, mit der Zielgruppe Studierende im Masterstudium. Aufgeführt werden acht Inhaltsbereiche, welche induktiv oder deduktiv zu Inhaltsbereichen für ein Bachelorstudium transformiert werden können [Artist Education Group, 2003]:

- Foundations of Computer Science and Engineering,
- Basic Control and Signal Processing,
- Theory of Computing,
- Real-Time Computing,
- Distributed Computing,

- Evaluation and Optimization of Extrafunctional Properties,
- System Architecture and Engineering,
- Applications.

2.4.6. Zusammenfassung – Ziele und Empfehlungen

Die vorgestellten Empfehlungen unterscheiden sich in ihrem Aufbau, ihrem Detailgrad und den fokussierten Zielgruppen. Die umfangreichen Arbeiten der gemeinsamen Arbeitsgruppe der ACM/IEEE fokussieren auf Bachelorstudiengänge der Informatik und technischen Informatik und berücksichtigen institutionelle Profilbildungen. Die Empfehlung der GI zur technischen Informatik beschränkt sich auf eine Sammlung von Kernthemen, welche in jedem Informatikstudium verortet sein sollen. Für Informatikstudiengänge mit und ohne Vertiefungsfach sind Module mit Affinität zur technischen Informatik verortet. Die Curriculaempfehlung der ARTIST beschreibt die Professionalisierung in einem Masterstudiengang zu eingebetteten Systemen, aus welchen deduktiv und induktiv Themen für ein Bachelorstudium abgeleitet werden können. Für eine Analyse der intendierten Kompetenzen und Vorwissensbeziehungen sind aufgrund der nicht stringenten Verwendung einer Taxonomie Ausschnitte aus den verschiedenen Empfehlungen zu betrachten.

2.5. Strukturierung von Lehr-Lernprozessen und fachdidaktische Ansätze

Im Forschungsprojekt KOMINA wurden Laborpraktika untersucht, welche bestimmte, zuvor normativ ermittelte Kompetenzen fördern sollen. Im Folgenden werden Ansätze zur Strukturierung von einführenden Lehrveranstaltungen zu eingebetteten Systemen vorgestellt. In der Lehrpraxis und wissenschaftlichen Publikationen sind zumeist Mischformen bei der didaktischen Gestaltung der Lehr-Lernprozessen zu finden.

2.5.1. Systemorientierter Ansatz

Baumann führte als Erster den systemorientierten Ansatz in der Didaktik der Informatik ein (vgl. [Baumann, 1993, S. 13]) und definiert ihn wie folgt:

1. „Es geht nicht um die Entwicklung isolierter Programme, sondern um die von Systemen, deren Komponenten (Teilsysteme) geeignet zusammenwirken.
2. In einem Informatiksystem handelt es sich um das Zusammenwirken verschiedener Wissensformen (Formen mathematischen, sprachlichen, naturwissenschaftlichen, sozialwissenschaftlichen, geografischen usw. Wissens).
3. Systementwicklung besteht im Zusammenwirken von Auftraggeber und Auftrag-

nehmer.

4. Der Unterricht verläuft nicht als Tätigkeit von Einzelkämpfern, sondern im Zusammenwirken der Projektteilnehmer“ [Baumann, 1996, S. 114].

Dies bedeutet, dass die Lehre in Projekten stattfinden sollte. Zu Projekten gehört immer ein Auftraggeber (Lehrender) und ein Auftragnehmer (Lernender). In Projekten üblich und von Baumann bestätigt, findet die Systementwicklung und Projektbearbeitung im Team statt. Da sowohl bei Informatiksystemen als auch bei eingebetteten Systemen der Entwurfsprozess und der Systembegriff als Verbindung von Hard-, Software und Netzverbindung im Mittelpunkt stehen, sind auch hier keine isolierten Programme, sondern das Zusammenwirken der Komponenten zu betrachten. Komponenten eingebetteter Systeme sind immer in den Kontext, eine physikalische Umwelt integriert, sodass das Zusammenwirken von dieser beeinflusst wird. Dies wird in Abbildung 2.8 skizziert, in dem ein Eingangssignal aus dem physikalischen Prozess abgetastet (Sensorik), verarbeitet und wieder zurückgeführt (Aktorik) wird.

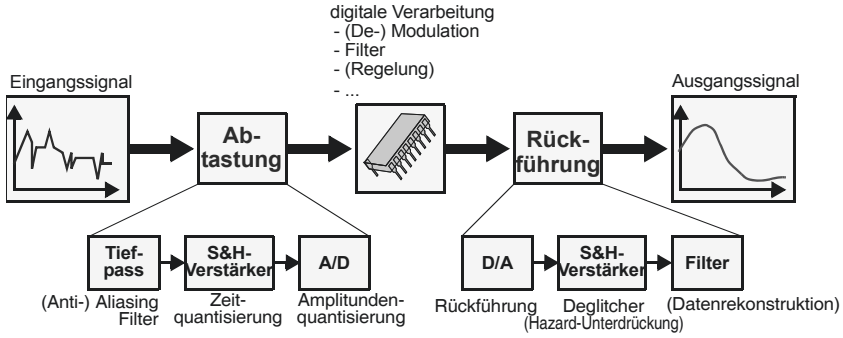


Abbildung 2.8.: Grundaufbau zur digitalen Verarbeitung analoger Signale [Berns et al., 2010, S. 78]

Baumann beschreibt den Lernprozess an allgemein bildenden Schulen. Eine theoretische Fundierung der systemorientierten Didaktik der Informatik hinsichtlich der Anwendbarkeit in der Hochschuldidaktik der technischen Informatik steht noch aus [Magenheim, 2003]. In weiteren Arbeiten wurden Beiträge zur SDdI geleistet (z. B. [Hampel et al., 1999]). So sieht Magenheim in [Magenheim, 2003, S. 17] das Konzept der Dekonstruktion als geeignete Alternative zum konstruktiven unterrichtsmethodischen Vorgehen an. Hierbei sollen sich Phasen des erkundenden und entdeckenden Lernens eines ausführlich dokumentierten Systems mit Phasen der Konstruktion eines eigenen Produkts abwechseln. Begründet wird dies mit lerntheoretischen Konzepten, wie dem exemplarischen Prinzip und fallbasiertem Lernen [Magenheim, 2003, S. 18]. Stechert analysiert die SDdI nach Baumann

und Magenheim und empfiehlt die Nutzung von Sichten [Stechert, 2009, S. 40ff]. Er differenziert drei charakterisierende Eigenschaften von Systemen nach [Claus und Schwill, 2006, S. 677] und begründet darauf eine grobe Strukturierung von Basiskompetenzen zu Informatiksystemen – das nach außen sichtbare Verhalten, die innere Struktur sowie Implementierungsaspekte von Informatiksystemen. [Claus und Schwill, 2006, S. 677f] betrachten Systeme der Informatik im Allgemeinen und schließen offene, das heißt in die Umgebung eingebettete und von ihr beeinflusste Systeme ein.

Stechert zeigt, dass Implementierungsdetails nicht für die Kompetenzentwicklung mit Informatiksystemen relevant und nur wenige ausgewählte Aspekte anhand von Programmiersprachen zu betrachten sind. Darüber hinaus seien Syntax- und Implementierungsdetails nur dann von gewissem Bildungswert, wenn die abstrakten Modelle veranschaulicht würden. Für die Zielgruppe im KOMINA Projekt – Entwickler von eingebetteten Systemen – gilt dies nicht, da die Implementierung des Systems eine wesentliche Aufgabe im Praxisalltag der Absolventen darstellt.

> „Computer engineers should be able to design and implement systems that involve the integration of software and hardware devices“ [ACM/IEEE, 2005, S. 27].

Verglichen mit der Implementierung von Anwendungssoftware für klassische Informatiksysteme unterscheiden sich die Konzepte vor allem in Abhängigkeit der gewählten oder aber durch die Rahmenbedingungen vorgegebene Hardwareumgebung. Klassische Informatiksysteme wie PCs oder Server abstrahieren von ihrer Hardwareumgebung mithilfe von Betriebssystemen, welche Ressourcen bereitstellen. Die Wahl der Implementierungssprache und -umgebung hängt von den volatilen State-of-the-Art Konzepten ab. In den vergangenen Jahren war hier das Konzept der Objektorientierung dominierend, sodass in der informatikdidaktischen Forschung im allgemeinbildenden Sektor – aber auch in Grundlagenveranstaltungen an Universitäten – diese Konzepte primär vermittelt wurden und werden. Für die Implementierung von eingebetteten Systemen gilt dies nicht, denn die Konzepte auf der programmiersprachlichen Ebene der Implementierung verändern sich hier weniger stark, wenngleich eine Tendenz zur verhaltensbeschreibenden modellierenden und anschließend synthetisierten Implementierung festgestellt werden kann (vgl. [Marwedel und Engel, 2011], [Lee und Seshia, 2012]). Ist zur Lösung des konkreten Anwendungsproblems eine dedizierte Hardware in Form eines Application-specific integrated circuit (ASIC) erforderlich, dann reicht eine ausschließliche Beschreibung des Verhaltens nicht aus und Arbeitsschritte auf weiteren Abstraktionsebenen und Dimensionen des Y-Diagramms bis zu fertigungsnahen Prozessschritten sind notwendig.

Da das Gros eingebetteter Systeme keine grafische Oberfläche besitzt, wird das Verhalten durch die Wirkung auf die physikalische Umgebung verdeutlicht. Die Wirkprinzipien und die innere Struktur bleiben dem Anwender in einer Art Black-

Box verborgen. Das Schließen auf die innere Struktur und Implementierungsaspekte durch die Betrachtung der Inputs und Outputs ist nur in Grenzen möglich. Um diese sichtbar zu machen ist die Vorstellung eines eingebetteten Systems als Black-Box aufzulösen.

Bei der Systementwicklung ist es notwendig, Teile eines Systems als Black-Box zu betrachten, was an der steigenden Komplexität des Gesamtsystems liegt. Modell- und Quelltextbibliotheken erlauben dem Entwickler Standardaufgaben wie Ein- und Ausgabe oder oft vorkommende Problemstellungen ohne erneute Implementierung der Algorithmen zu verwenden. Die zur Verfügung gestellten Schnittstellen der Teilsysteme bilden dann eine Grenze zur Black-Box, in der die Implementierungsdetails verborgen bleiben. Aufgrund der immer komplexer werdenden eingebetteten Systeme ist es einem einzelnen Entwickler nicht mehr möglich, alle Systemkomponenten eigenverantwortlich in endlicher Zeit zu entwerfen. Durch die Aufteilung in Teams und Teilverantwortlichkeiten je Systemkomponente ist die Definition von Schnittstellen und die Betrachtung der angrenzenden Komponenten als Black-Box erneut notwendig. In der Lehre kann mit dem Grad der Abstraktion und Wahl einer geeigneten didaktischen Reduktion durch die Bereitstellung von Teillösungen mit einem einzigen System verschiedenste Zielgruppen bedient werden – von Masterstudierenden bis zum Novizen in den ersten Semestern (vgl. Abschnitt 2.1). Ein Beispiel dafür sind Mikrocontrollerboards der Produktfamilie Arduino[9], welche einerseits durch die Verwendung von Bibliotheken eine einfache Implementierung durch Schüler der Sekundarstufe II ermöglicht (Programmierumgebung Scratch[10]), andererseits kann auch in nativem Assembler oder Standard C programmiert werden, was bei der Gestaltung von Lehr-Lernprozessen eine hohe Flexibilität erlaubt (z. B. [Brand et al., 2011]).

Außer Acht gelassen wird in Lehrveranstaltungen für Novizen oftmals, dass über die Hardware der ausführenden Einheit hinaus Hardware als Schnittstelle zur physikalischen Umwelt notwendig ist. Für ein ganzheitliches Systemdesign ist auch die Systemhardware an der Schnittstelle und damit die Grenze der Black-Box zu betrachten und in einführende Lehrveranstaltungen zu integrieren. In [ACM/IEEE, 2001] wird eine Implementierung eines Curriculums nach einem systemorientierten Ansatz vorgeschlagen und beinhaltet folgende Kurse, welche einen höheren praktischen aber dennoch ausreichenden theoretischen Anteil für ein grundlegendes Verständnis auch künftiger Technologien umfasst:

- Introduction to Computer Organization,
- Algorithm Design and Analysis,
- Computer Architecture,
- Operating Systems and Networking,

[9] Open-Source Mikrocontroller Plattform [http://www.arduino.cc/, Abruf 12/2014].
[10] Ein Projekt der *Lifelong-Kindergarten-Group* [http://scratch.mit.edu/, Abruf:12/2014].

- Programming Language Translation,
- Computer Graphics,
- Artificial Intelligence,
- Information Management,
- Software Development and Systems Programming,
- Capstone Project.

Es werden also Systeme als Ganzes betrachtet, von der Organisation der Komponenten über Schnittstellen bis hin zur Entwicklung von Systemsoftware.

2.5.2. Dekonstruktion

Dekonstruktion als didaktische Methode zur Einführung in den objektorientierten Systementwurf wurde erstmals im Forschungsprojekt objektorientiertes Modellieren im Informatikunterricht (OMI) an der Universität Paderborn erforscht [Hampel et al., 1999, S. 1]. Ein Konsens zur Einführung der Objektorientierung in den Informatikunterricht war zu dieser Zeit noch nicht gefunden und sollte durch diese Arbeiten von einem weiteren - an der Mode orientierten - Programmierparadigma abgegrenzt werden, was auch mit einer sich dynamisch entwickelnden Bezugswissenschaft begründet wird. Vielmehr sollen analytisches Denken, Abstraktionsvermögen, zergliedern von Problemen und Problemlösestrategien im Fokus eines an die systemorientierte Didaktik angelehnten Ansatzes sein [Hampel et al., 1999, S. 1]. Diese Begründung und damit verbundene Auswirkung auf die Lehr-Lernpraxis gilt ebenso für die technische Informatik. Analysephasen sollen sich, nach dem Ansatz der Dekonstruktion, mit Phasen der Dekonstruktion abwechseln, um so zu einer Vertiefung der Kenntnisse und einem Transfer in andere Problembereiche zu gelangen. Konzepte und Implementierungen anderer Entwicklerteams werden durch Erkundungsaufträge weitgehend selbständig herausgearbeitet. Erwartet wird, dass der Zusammenhang des gesamten Systems fassbar wird und somit die Implementierung der Einzelheiten später leichter fällt. Eine Gefahr sieht der Autor in der Erkundung der Teilsysteme, wenn durch das Reproduzieren einzelner Bausteine die Implementierung nicht mehr durch die Anwendung der erlernten Konzepte und Syntax geschieht, sondern durch das Kopieren von Lösungsansätzen und einem Versuch-Irrtum-Vorgehen. Zwar werden analytische Fähigkeiten gefördert, aber je nach Detailgrad der Dokumentation ist nicht das Verständnis für die Funktionsweise wichtig, sondern nur das Verständnis wofür einzelne Programmierkonstrukte benötigt werden und bei welchen Parameteränderung sich die eigene Lösung dem gewünschten Verhalten annähert.

Im Fokus der weiteren Arbeiten zur Dekonstruktion stand stets die Softwareentwicklung, obgleich Magenheim alle Teile einer systemorientierten Didaktik, also die Einheit von Software, Hardware, Netzwerk und assoziiertem sozialen Handlungs-

system von Personen im sozio-technischen Kontext in seine Betrachtung einschloss. Begründet wird dies mit einer nicht trennscharfen Abgrenzung der Komponenten eines Informatiksystems, da diese sowohl in Software als auch in Hardware realisiert werden könnten [Magenheim, 2003, S. 2].

In der technischen Informatik haben sich zwei wesentliche Ansätze etabliert – der projektorientierte und der praktikumsorientierte Ansatz beziehungsweise deren Kombination. In der systemorientierten Didaktik wird die besondere Eignung von Projekten bei der Vermittlung von Fach- und Schlüsselkompetenzen hervorgehoben. Teamarbeit fördert die nicht-kognitiven Kompetenzen der Studierenden sowie das Verständnis von Systemen als Komposition von Komponenten, welche durch Schnittstellen beschrieben und unabhängig voneinander entwickelt werden können. Praktika ermöglichen eine erstmalige Anwendung des zuvor erlangten Faktenwissens in dedizierten Lehr-Lernarrangements zumeist in Laboren, was sich durch eine Verstärkung der intrinsischen Motivation als förderlich erwiesen hat [Bouldin, 2004].

> „Capstone design projects often involve traditional engineering disciplines (e.g., circuit design and mechanism construction), as well as computing tasks (e.g., designing a small microcontroller that implements a sensor and/or control function). These projects provide an excellent opportunity for students to gain hands-on experience with embedded computing in a realistic, physical context“ [Sztipanovits et al., 2005, S. 555].

In dieser Arbeit wird diskutiert wie sich die als geeignet erwiesenen Ansätze der Dekonstruktion in der systemorientierten Didaktik in Laborpraktika strukturieren lassen. Erfolge bei der Förderung von Kompetenzen im Projekt KOMINA bilden dabei die Grundlage für weitere Strukturierungsmaßnahmen und der Bündelung informatikdidaktischer Forschung für didaktische Systeme.

2.5.3. Didaktische Systeme

Für die Strukturierung der Lehr-Lernprozesse durch Sequenzierung von Fachkonzepten sowie der Einführung von Abstraktionsebenen eingebetteter Systeme erwies sich das Konzept der didaktischen Systeme als geeignet. [Brinda, 2001], [Brinda, 2004] und [Freischlad, 2010] führten das Konzept der didaktischen Systeme als Strukturierungsinstrument ein. Während [Brinda, 2004] die objektorientierte Modellierung fokussierte, diskutierte und transferierte [Freischlad, 2010] das Konzept der didaktischen Systeme auf das Themengebiet Internetworking.

Ziel der didaktischen Systeme ist die Verknüpfung wesentlicher Bestandteile des Lehr-Lernprozesses zu einem verzahnten und in sich schlüssigen didaktischem Setting. Die Zerlegung von (Kompetenz-)Zielen in kleinere aufeinander aufbauende und sich ergänzende Teile, den Wissensstrukturen als Basis für den Erwerb von Problemlösefähigkeiten, bestimmt die Auswahl von Aufgaben und die dazugehörige

lernunterstützende Hard- und Software zur Exploration von Informatikkonzepten.

Wissensstrukturen

Wissensstrukturen – fachdidaktisches Metawissen – erfüllen drei didaktische Funktionen.

> „1. Orientierungsfunktion: Wissen wird in einen größeren Zusammenhang gestellt. Lernende werden dadurch insbesondere in Phasen der Reflexion und Lernende und Lehrende bei der systematischen Bestandsaufnahme des Lernfortschritts unterstützt. Lehrende können erwartete und erreichte Lernerfolge vergleichen.
>
> 2. Organisationsfunktion: Mögliche nächste Schritte im Lehr-Lernprozess werden dargestellt. Lehrende können dies zur flexiblen Unterrichtsplanung nutzen. Bekannte Voraussetzungen im Hinblick auf das Vorwissen der Lernenden werden berücksichtigt und unvorhergesehene Schwierigkeiten führen zur Verfeinerung der Wissensstruktur. Lernende können Wissensstrukturen für selbständiges Lernen verwenden.
>
> 3. Diskussionsfunktion: Didaktische Entscheidungen, die zu konkreten Lehr-Lernprozessen führen, werden sichtbar gemacht und damit für die Diskussion zugänglich. Durch eine geeignete Darstellung können Unterschiede und Gemeinsamkeiten verschiedener Zugänge erkennbar werden“ [Freischlad, 2010, S. 72f].

Zur Darstellung können Graphen verwendet werden, deren Knoten Wissenselemente beziehungsweise Lerneinheiten repräsentieren. Die Transitionen beschreiben die Art der Relation. [Brinda, 2004] nutzt Und-Oder-Graphen, bei denen Oder-Verknüpfungen *ist hilfreich für* und eine Und-Verknüpfungen *ist notwendig für* modellieren. Die Anforderungen an die Darstellungsform beschreibt [Freischlad, 2010, S. 75ff] mit Bezug auf [Brinda, 2004, S. 181f] mit den Kriterien Ausdrucksstärke, Übersichtlichkeit und Nachvollziehbarkeit.

> **„Ausdrucksstärke** ist die Voraussetzung dafür, dass Lehrende und Lernende sich anhand der Wissensstrukturen im Lehr-Lernprozess orientieren können. Das bedeutet insbesondere, dass mit der Darstellungsform die inhaltlichen Anforderungen ausgedrückt werden können. Dazu gehört auch, dass die Darstellungsform Erarbeitungs- und Vorwissensbeziehungen abbilden muss.
>
> **Übersichtlichkeit** beschreibt den Anspruch, dass der Grad der Detaillierung beziehungsweise der Abstraktion angemessen ist. [Brinda, 2004] empfiehlt, dazu möglichst wenige Darstellungsmittel zu verwenden: Knoten und Kanten.
>
> **Nachvollziehbarkeit** beschreibt das Ziel, eine leicht verständliche Darstellung zu benutzen. [Brinda, 2004] empfiehlt, dass eine standardisierte

> Darstellungsform verwendet wird, um einen intuitiven Zugang zu ermöglichen und existierende Werkzeuge zur digitalen Bearbeitung, Speicherung und Weitergabe verwenden zu können [...] “ [Freischlad, 2010, S. 75].

Unter Einbeziehung wissenschaftlicher Erkenntnisse der Lerntheorie (vgl. [Ausubel et al., 1980, Edelmann, 2000]) unterschied [Freischlad, 2010, S. 83f] fünf Arten von Relationen, welche im Folgenden auf die Entwicklung eingebetteter Systeme angepasst werden. Dazu werden Beispiele aus dem Entwurfs- und Anwendungspraktikum als repräsentatives Entwicklungsvorhaben verwendet.

Relation R1 Beobachtbare Objekte vor nicht sichtbaren Abläufen.

Interne Abläufe in Mikrocontrollersteuerungen bleiben dem Anwender verborgen. Beobachtbar beziehungsweise erfahrbar sind physikalische Größen wie Temperatur und Helligkeit sowie das Wirken auf den Prozess mit Aktoren, wie LEDs. Daher ist es sinnvoll, zunächst den Prozess und anschließend die Verarbeitung zu betrachten.

Relation R2 Zuerst das untergeordnete Fachkonzept und dann das übergeordnete Fachkonzept.

Resistive Temperatursensoren werden vielfach in eingebetteten Systemen eingesetzt, da der Spannungsabfall relativ einfach durch Analog/Digital-Wandler erfasst und in Mikrocontrollern verarbeitet werden kann (Temperatursensor „ist-ein“ resistiver Sensor). Bevor man resistive Sensoren einführt, kann das Verhalten am Beispiel des Temperatursensors gezeigt werden.

Relation R3 Einzelne Elemente vor zusammengesetzten Wissenselementen.

Ein eingebettetes System besteht aus Sensoren, Aktoren und einer digitalen Verarbeitungseinheit („ist Teil von“-Relation). Hier ist die schrittweise Einführung zunächst von Sensoren, danach Aktoren, dann die Verarbeitung und abschließend die Verbindung zu einem System sinnvoll.

Relation R4 Ein anschauliches Beispiel zur Illustration, als Kontext oder zur Schaffung eines Problembewusstseins.

Anschauliche Beispiele eigenen sich in verschiedenen Abschnitten im Lehr-Lernprozess. Ein Beispiel ist das Dimmen einer LED mittels Pulsweitenmodulation (PWM), in denen den Studierenden die Trägheit von Sensoren (das eigene Auge) und die Nichtlinearität verdeutlicht wird.

Relation R5 Deklaratives Wissen vor Handlungswissen.

Grundlagen der Elektrotechnik umfassen das Verhalten von Strom und Spannungen in gemischten Schaltungen. Um elektrische Größen zu messen, ist es einerseits wichtig zu wissen wie sich die Größen in Grundschaltungen verhalten (Kirchhoffsche Regeln), aber anderseits hilfreich, Kenntnisse über die

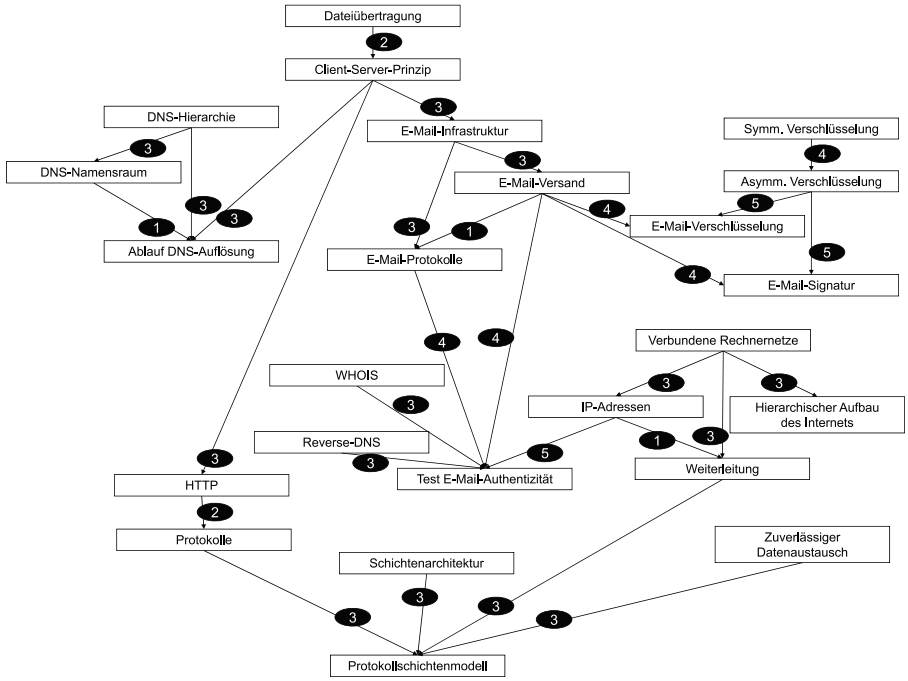


Abbildung 2.9.: Beziehungstypen der Wissensstruktur Internetworking [Freischlad, 2010, S. 84]

Funktionsweise von Messinstrumenten zu haben. Dadurch ist dann ein strukturierteres Vorgehen und eine Interpretation der Messwerte möglich.

[Schwidrowski, 2010] empfiehlt R5 nicht unreflektiert zu übernehmen, da der Lernende vom Handeln über Können zum Wissen geführt würde (vgl. [Jank und Meyer, 2005]). Bezogen auf komplexe Anforderungssituationen kann dem nicht zugestimmt werden. Erfahrungen aus der formativen Evaluation des Entwurfs- und Anwendungspraktikums (siehe Abschnitt 3.4) haben gezeigt, dass kein strukturiertes Vorgehen möglich ist solange kein deklaratives Wissen – bezogen auf die Terminologie und naturwissenschaftliche Grundlagen – vorhanden ist.

Jeder in Abbildung 2.9 dargestellte Knoten repräsentiert eine Unterrichtseinheit. Im Folgenden wird an zwei Beispielen gezeigt, dass das Kriterium der Nachvollziehbarkeit durch Angabe der Art der Relation nicht ausreicht.

1. *Symmetrische Verschlüsselung - asymmetrische Verschlüsselung (4)*: [Freischlad, 2010] verwendet die symmetrische Verschlüsselung als ein anschauli-

ches Beispiel für die asymmetrische Verschlüsselung. Dies kann bestätigt werden, sofern man die Aushandlung eines symmetrischen Schlüssels in einem asymmetrischen Verfahren betrachtet. Andererseits können beide Konzepte unabhängig voneinander behandelt und als untergeordnete Fachkonzepte der Verschlüsselung diskutiert werden. Im Hinblick auf die sich anschließenden Unterrichtseinheiten (E-Mail-Verschlüsselung und E-Mail-Signatur) ist die symmetrische Verschlüsselung nicht in jedem Fall als Voraussetzung anzusehen.

2. *HTTP - Protokolle (2)*: [Freischlad, 2010] betrachtet das Hypertext Transport Protocol (HTTP) als untergeordnetes Fachkonzept zu Protokollen. Die Verwendung der „ist ein“-Relation ist zutreffend. HTTP ist ein konkretes Produkt und könnte im Lehr-Lernprozess durch das File Transfer Protocol (FTP) oder jedes auf dem Client-Server-Prinzip basierende Protokoll ersetzt werden (wenngleich HTTP als lebensweltnahes Protokoll sinnvoll ist). Man könnte HTTP also auch als anschauliches Beispiel verwenden.

Daraus folgt, dass ohne eine ausführliche Beschreibung der Beziehungen die geforderte Ausdrucksstärke nicht immer erreicht wird und daher weitere beschreibende Elemente hinzugefügt werden müssen. Es ist ersichtlich, dass eine Differenzierung der Grob- beziehungsweise Feinziele notwendig ist, da die Strukturierung der Fachkonzepte allein keine hinreichende Orientierung im Lehr-Lernprozess ermöglicht. Im Rahmen dieser Arbeit dienen die Konzepte als informatikdidaktische Grundlage, welche hinsichtlich Zielgruppe und Themengebieten adaptiert werden, um zu einer didaktisch begründeten Strukturierung von kognitiven Strukturen sowie einer Verfeinerung eines Kompetenzstrukturmodells zu gelangen.

2.6. Zusammenfassung

Gegenstand der Lehre ist nicht mehr das zu lehrende Wissen, also der Inhalt, sondern die zu erlernende Kompetenz, für deren Erlangen dennoch Faktenwissen die Basis bildet. Durch die Beschreibung von Kompetenzen und deren Niveaustufen durch geeignete Operatoren einer Taxonomie (z. B. [Anderson et al., 2000] und [Fuller et al., 2007]) sowie deren Strukturierung in Kompetenzmodellen beschreiben sie den Output von Bildungsprozessen. Durch die Einführung einer Taxonomie zur Verbesserung der Vergleichbarkeit von Publikationen zur HdTI können kulturspezifische Praxis- und Forschungsbeiträge analysiert werden. Damit wird Forschern und Lehrenden die Möglichkeit gegeben eine ad hoc Einschätzung der Relevanz für die eigene Arbeit und die individuellen Ebenen der Abstraktion, kognitiven Anforderung, Komplexität der Problemstellung sowie deren Profil vorzunehmen. Nationale und internationale Empfehlungen zur Gestaltung von Studiengängen der (technischen) Informatik dienen als Quelle zur normativen Ableitung der Kompetenzen für ein Kompetenzstrukturmodell. In der vorliegenden Arbeit

werden die kognitiven Fähigkeiten und Fertigkeiten sowie motivationale Aspekte in einem, nach dem systemorientierten Ansatz strukturierten, Laborpraktikum empirisch erforscht. Das Konzept der didaktischen Systeme bietet einen Forschungsrahmen zur Strukturierung von Lehr-Lernprozessen mit den Komponenten Wissensstrukturen, Aufgabenklassen und Lehr-Lernsoftware. Es wurde gezeigt, dass die Ausdrucksstärke von Wissensstrukturen nicht für die HdTI ausreicht und erweitert werden muss.

3. Diskussion zur Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen

In diesem Kapitel wird die Forschung zur Didaktik der technischen Informatik, welche die Modellierung von Kompetenzen von EMNS-Entwicklern beinhaltet, sowie deren Einflüsse auf die Gestaltung von Lehrveranstaltungen diskutiert. Zentrale Themen sind die Forschungsschritte innerhalb des Projektes KOMINA, dessen Ziel das Leisten eines Beitrages zur technischen Informatik und ihrer Didaktik war und die Entwicklung kompetenzfördernder Experimente für Laborpraktika zu eingebetteten Systeme einschloss.

Eine den Forschungsprozess des Projektes reflektierende Arbeit existiert bislang nicht. Das Forschungsgebiet Hochschuldidaktik der technischen Informatik ist in besonderem Maße vielfältig und facettenreich (vgl. Kapitel 1), sodass Forschungsergebnisse zur Didaktik immer im Kontext der konkret untersuchten Lehr-Lernprozesse interpretiert werden müssen (vgl. Kapitel 2). Unter Berücksichtigung dieser Kontexte tragen die Ergebnisse aus KOMINA zu einer Verbesserung von Lehr-Lernprozessen bei. Darüber hinaus erlaubt die Reflexion des Forschungsprojektes eine neue Sichtweise auf die HdTI, indem von kulturspezifisch geprägten Praxisberichten zu vergleichbaren und theoretisch fundierten Konzepten einer Kompetenzorientierung bei der Entwicklung eingebetteter Systeme übergegangen wird.

Der erste Schwerpunkt des KOMINA-Projektes war die Erbringung eines theoretisch fundierten Beitrages zur systemorientierten Didaktik der technischen Informatik einschließlich lernförderlicher Experimente [Schubert et al., 2010]. Baumann führte als Erster den *systemorientierten Ansatz* in der Didaktik der Informatik ein (vgl. Kapitel 2). Da das Gros eingebetteter Systeme keine grafische Oberfläche besitzt (Steuerung technischer Prozesse), wird das Verhalten durch die Wirkung auf die physikalische Umgebung sichtbar. Die Wirkprinzipien und die innere Struktur bleiben dem Anwender in einer Art Black-Box verborgen. Das Schließen auf die innere Struktur und Implementierungsaspekte durch die Betrachtung der In- und Outputs ist nur in Grenzen möglich. Um diese sichtbar zu machen, ist die Vorstellung eines eingebetteten Systems als Black-Box aufzulösen. Bislang wurde nicht erforscht bis zu welchem Grad ein Betrachten von Teilsystemen als Black-Box ausreicht, um selbständig Lösungen zu entwickeln. Angenommen wird, dass

Bottom-Up strukturierte Bauteile zukünftig einen Einfluss auf höhere Abstraktionsebenen des Systementwurfs haben werden [Schubert und Schwill, 2011]. In Kapitel 2 wurde belegt, dass der ganzheitliche Systementwurf, welcher die Hardware und Schnittstellen eines Systems einschließt, in Curricula von Informatikstudiengängen zu verorten ist. Dazu werden im zweiten Schwerpunkt von KOMINA Kompetenzmodelle und deren exemplarische Erprobung in Laborpraktika, welche zum Auflösen des Back-Box-Denkens dienen sollen, erforscht:

> „Ein Kompetenzmodell (Dimensionen, Niveaustufen) wird eingeführt, um das Entwerfen von EMNS mit gemischt digital-analogen Funktionsprinzipien zu unterstützen. Dazu ist ein Verstehen grundlegender Hardwarefunktionalität durch Beschreibung physikalischer Phänomene auf einer qualitativen Ebene erforderlich“ [Schubert et al., 2010, S. 1].

Die Qualität des entwickelten Modells wird an zwei Kriterien bewertet – *Vollständigkeit* und *Angemessenheit* [Schubert et al., 2010].

Vollständigkeit bedeutet, dass alle wichtigen Kompetenzfacetten, welche zur Entwicklung von eingebetteten Systemen notwendig sind, im Kompetenzstrukturmodell aufgenommen wurden. Eine Aussage über den Grad der Granularität wird nicht vorgenommen. Offensichtlich ist, dass das breite Praxisfeld der Entwicklung eingebetteter Systeme mit allen Teilaspekten nicht im Rahmen eines einzelnen Forschungsprojektes erfassbar und erforschbar ist. Ziel in KOMINA und im Forschungsprojekt des Autors ist, im Rahmen der Grundlagenforschung zur HdTI eine erste Gliederung von Lehr-Lernprozessen anhand der erforschten Kompetenzstrukturen vorzunehmen und damit für weitere Forschungsprojekte nutzbar zu machen. Die Vollständigkeit hängt zum einen vom verwendeten Quellmaterial, zum anderen vom Detaillierungsgrad des Modells ab. Lassen sich einzelne Kompetenzen ohne Anpassung der Struktur des Modells nachträglich integrieren, bleibt das Kriterium der Vollständigkeit weiterhin erfüllt. Existieren wichtige Kompetenzen, welche sich nicht integrieren lassen, so ist eine Umstrukturierung des Modells notwendig. Das Kriterium Wichtigkeit von Kompetenzen und deren Beschreibungen wurde empirisch erforscht (vgl. Abschnitt 3.2). Das Kriterium Angemessenheit drückt aus, dass Kompetenzfacetten einen angemessenen Platz im Modell erhalten, was eine adäquate Niveaustufe einschließt. Die Ebene im Modell (Dimension, Subdimension etc.) ist dabei irrelevant.

Im dritten Forschungsschwerpunkt soll erforscht werden, inwiefern Techniken künftiger nanoskaliner eingebetteter Systeme in Lehr-Lernprozessen gefördert werden können.

> „Der Paradigmenwechsel hin zu Bottom-Up-Techniken und Redundanz beim Entwurf von EMNS wird nachvollzogen. Hier ist es bedeutsam zu lernen, durch Redundanz fehlerbehaftete Systeme nutzbar zu machen und den Weg von den klassischen zentral gesteuerten Systemen hin zu parallelen verteilten

> Systemen in Kombination mit einer Strukturbildung durch Selbstorganisation zu finden“ [Schubert et al., 2010, S. 1].

Erkenntnisse aus den Forschungsarbeiten zum Kompetenzmodell sollen dafür genutzt werden einzelne, das Thema Nanosysteme am stärksten tangierende Konzepte und Technologien zu identifizieren und damit Kompetenzen zu fördern, welche zukünftig benötigt werden. Die physikalischen Eigenschaften nanoskaliner Bauteile haben Auswirkungen auf das Verhalten und die Struktur eines Systems. Diese Auswirkungen sollen nachvollzogen und in lernförderlicher Software für Informatikstudierende aufbereitet werden.

KOMINA (Forschungsdesign)

Das Arbeitsprogramm des Projektes war auf drei Jahre ausgelegt und wurde von der DFG zunächst für zwei Jahre bewilligt. Daraus ergibt sich ein von der ursprünglichen Projektskizze abweichendes Vorgehen, das sich auf die wesentlichen Arbeitspakete der drei beantragten Jahre beschränkt und insbesondere Auswirkungen auf die Methodik bei der Evaluation des im Rahmen von KOMINA entwickelten Praktikums zur exemplarischen Umsetzung des Kompetenzstrukturmodells hatte. Als wesentlich wurden die ersten beiden Forschungsschwerpunkte betrachtet, um zunächst Kompetenzen zu erforschen, welche dem aktuellen Bedarf an Forschung zur HdTI entsprechen. Der dritte Schwerpunkt beschränkt sich auf eine konzeptionelle Ebene sowie Empfehlungen für die Einführung nanoskaliner Bauteile und zugehöriger Konzepte. Eine praktische Erprobung eines E-Learning Kurses *NanoArch-Online* war im Projektzeitraum nicht möglich, was die zunächst geplante empirische Überprüfung der Kompetenzaneignung einschließt. Der Forschungsverlauf des Projektes ist in Abbildung 3.1 aufgeführt und umfasst den ersten, höherprior verfolgten Forschungszweig – Kompetenzen von Entwicklern eingebetteter Mikrosysteme. Die einzelnen Arbeitspakete sind als Rechtecke dargestellt, wesentliche Ergebnisse – das empirisch verfeinerte Kompetenzstrukturmodell sowie die entwickelte lernförderliche Hard- und Software – als Parallelogramme.

Normative Ableitung eines Kompetenzstrukturmodells: Um zu anwendbaren Konzepten der Kompetenzförderung in der technischen Informatik zu gelangen, ist zunächst die Struktur von Kompetenzen und deren Facetten zu erforschen. Die Herleitung einer Gliederung in einem Kompetenzstrukturmodell erfolgt zunächst normativ.

Empirische Überprüfung des Kompetenzstrukturmodells zum EKSM: Eine empirische Überprüfung des normativ abgeleiteten Kompetenzstrukturmodells war notwendig, um die zunächst intersubjektiv hergeleitete Struktur von Kompetenzen hinsichtlich Vollständigkeit und Wichtigkeit zu validieren. Experten der technischen Informatik bewerteten die identifizierten Kompetenzen in einer zweistufigen schriftlichen Befragung.

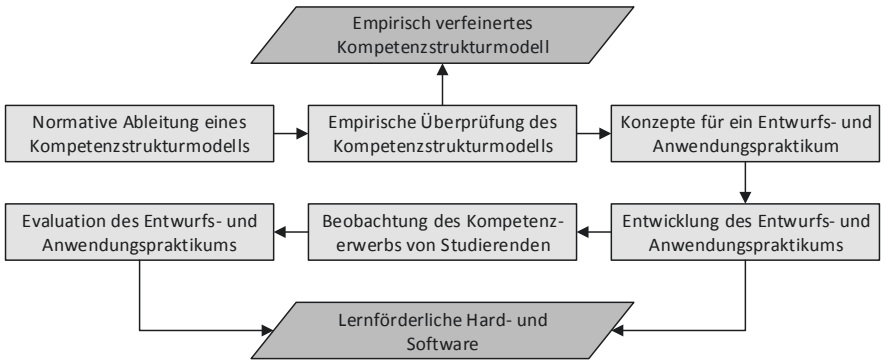


Abbildung 3.1.: Schwerpunkte im KOMINA Forschungsverlauf

Konzeption und Entwicklung des EAP: Es wird zu eingebetteten Mikrosystemen mit explizitem Hardwarebezug übergegangen, die eine hinreichende Nähe zu elektrisch-physikalischen Problemstellungen bieten. Die Automatisierung eines mit Sensoren und Aktoren ausgestatteten Modellhauses dient als Basis für die Entwicklung von Schaltungen und der Implementierung auf Mikrocontrollern und FPGAs.

Empirische Überprüfung des Kompetenzerwerbs und Evaluation: Um bereits in der ersten Erprobung des neuen Praktikums wertvolle Erkenntnisse zur Verbesserung der Laborversuche zu erhalten wurde eine Evaluation mittels nichtteilnehmender Beobachtung durchgeführt und typische Lernhürden identifiziert.

Entwicklung lernförderlicher Hard- und Software: Die zuvor identifizierten Lernhürden implizieren den Bedarf an weiterer lernunterstützender Hard- und Software im Entwurfs- und Anwendungspraktikum, welche an die konkreten Lehr-Lernprozesse, institutionelle Rahmenbedingungen und Zielgruppe angepasst werden.

Aufgrund des ergebnisreichen Projektverlaufs, der sich in zahlreichen Publikationen manifestierte, sowie den aufgedeckten neuen Forschungsbedarfen wurden weitere Forschungsarbeiten zur HdTI durchgeführt und veröffentlicht (vgl. [Büchner, 2014] [Büchner und Schubert, 2014] [Schäfer und Brück, 2013]). Im Folgenden werden die aufgeführten Arbeitspakete innerhalb des KOMINA Projektes diskutiert und um Ergebnisse aus weiteren Analysen des Autors ergänzt.

3.1. Normative Entwicklung des Kompetenzstrukturmodells für das Entwickeln von eingebetteten Mikro- und Nanosystemen (NKSM)

Kompetenzmodelle (siehe Kapitel 2) bieten eine theoretisch fundierte Vorstellung von Outputorientierung. In der HdTI ist zunächst die Strukturkomponente zu erforschen, welche Kompetenzen und deren Facetten zur Entwicklung eingebetteter Systeme in einem Modell gliedert (Kompetenzstrukturmodell).

Als Orientierung zur Entwicklung des NKSM dienten die Forschungsarbeiten in den Projekten MoKoM I und MoKoM II, in denen basierend auf dem Ansatz von [Klieme et al., 2007] internationale Empfehlungen und Standards von IEEE, ACM, International Federation for Information Processing (IFIP) und GI analysiert wurden (z. B. [Nelles et al., 2010], [Magenheim et al., 2010], [Magenheim et al., 2013], [Lehner et al., 2010]). Nicht-kognitive Kompetenzen der an der Softwareentwicklung orientierten Modellstruktur in MoKoM entsprechen denen von Entwicklern und Anwendern eingebetteter Systeme, weshalb diese in die vierte Dimension des NKSM überführt wurden.

Den Quellenkorpus für eine Strukturierung von Kompetenzen zur Entwicklung eingebetteter Systeme bildete zunächst die Empfehlungen der ACM, IEEE und GI (vgl. [ACM/IEEE, 2008], [GI, 2011], Kapitel 2), sowie Modulbeschreibungen universitärer Lehrveranstaltungen zur technischen Informatik und zu eingebetteten Systemen. Neben den an KOMINA beteiligten Universitäten – [Universität Siegen, 2011] und [FAU Erlangen-Nürnberg, 2011] – wurden die Modulbeschreibungen der drei Hochschulen RWTH Aachen, TH Karlsruhe und TU München analysiert (vgl. [RWTH Aachen, 2011], [TH Karlsruhe, 2011], [TU München, 2011]). Die Selektion von Kompetenzfacetten erfolgte von den Experten im KOMINA Projekt unabhängig voneinander und konzentrierte sich auf hardwarenahe Tätigkeitsbeschreibungen und damit sich von der Softwareentwicklung abgrenzende Themengebiete der technischen Informatik. Dazu wurden Kompetenzen, welche die Schnittstelle zwischen Soft- und Hardware betreffen, identifiziert. Ergebnis dieser Inhaltsanalyse ist eine Sammlung von Kompetenzbeschreibungen, welche in vier Dimensionen (*K1-K4*) gegliedert wurden (siehe Tabelle 3.1).

3.1.1. Methodik

Die Analyse des Quellmaterials orientierte sich am Vorgehen der qualitativen Inhaltsanalyse nach [Mayring, 2010], um zu intersubjektiv nachvollziehbaren Ergebnissen trotz einer subjektiv geprägten Sicht auf eingebettete Systeme zu gelangen [Bortz und Döring, 2005]. Die strukturierende Inhaltsanalyse verwendet Kategorien für die Kodierung von Sinneinheiten. Dieses Kategoriensystem wird in einem ersten Durchlauf erprobt und gegebenenfalls angepasst, bevor das Quellmaterial

Tabelle 3.1.: Normatives Kompetenzstrukturmodell [Jaschke et al., 2011]

K1: Kompetenzen als Voraussetzung	**K2: Entwicklungskompetenzen**
K1.1: Mathematik	K2.1: Org. des Entwicklungsprozesses
K1.2: Physik	K2.2: Anforderungsanalyse
K1.3: Informatik	K2.3: Systemdesign
K1.4: Elektrotechnik	K2.4: Implementierung
K1.5: Materialwissenschaften	K2.5: Optimierung und Test
K1.6: Englisch	
K1.7: Wissenschaftliches Arbeiten	
K1.8: Lernorganisation	
K3: Kompetenzen für ebenenübergreifendes Entwickeln	**K4: Nicht-Kognitive Kompetenzen**
K3.1: Top-Down	K4.1: Einstellungen
K3.2: Bottom-Up	K4.2: Sozial-kommunikative Komp.
K3.3: Jojo	K4.3: Motivationale- und volitionale Komp.

von den Experten unabhängig, anhand eines Kodierleitfadens, untersucht wird. In KOMINA erfolgte diese Kodierung implizit durch die Präzisierung des Schwerpunktes auf hardwarenahe Tätigkeitsbeschreibungen und damit sich von der Softwareentwicklung abgrenzende Themengebiete. Dass dabei auch niedere Abstraktionsebenen des Entwicklungsprozesses fokussiert werden, war gegeben. Gleichzeitig führte diese Schwerpunktlegung dazu, dass zentrale Konzepte der Entwicklung von eingebetteten Systemen keine eigene Ausprägung im NKSM fanden – beispielsweise das Echtzeitverhalten eingebetteter Systeme. In den Publikationen zum NKSM [Schäfer et al., 2011] und [Jaschke et al., 2012] wird eine Teilmenge der zugeordneten Kompetenzbeschreibungen aufgeführt. Ankerbeispiele beschreiben die Kategorien exemplarisch und dienen als Kodierleitfaden. Bei der Analyse wurden darüber hinaus jene Beschreibungen ausgewählt, welche eine Zuordnung zu Operatoren der Lernzieltaxonomie nach [Anderson et al., 2000] ermöglichen. Aufgrund von Interpretationsspielraum und Übersetzungsungenauigkeiten, wurde in nicht eindeutigen Fällen eine jeweils tiefere kognitive Stufe gewählt, um die Erwartungen an Studierenden nicht über die eigentliche Intention von Experten hinaus zu erhöhen und damit gegebenenfalls zu überfordern [Jaschke et al., 2012].

3.1.2. Quellenlage

Eine Beschränkung auf die nationale Empfehlung für die technische Informatik (vgl. [GI, 2011]) und die internationale Empfehlung für *Computer Science* (vgl. [ACM/IEEE, 2008]) war für die Erstellung des Kodierleitfadens geeignet, aber wenig ergebnisreich im Hinblick auf die Anzahl der Kompetenzbeschreibungen.

In KOMINA wird die Entwicklung eingebetteter Systeme als Teil der Ausbildung von Informatikern, als künftige Systementwickler, betrachtet. Klar ist, dass das gesamte Spektrum an Kompetenzen nicht in nur wenigen Modulen gefördert werden kann, um Spezialisten für die Entwicklung eingebetteter Systeme auszubilden. Dies würde der Existenz spezieller Studiengänge widersprechen und nicht einer grundlagenorientierten Ausbildung in Bachelorstudiengängen dienen. Des Weiteren herrscht in der Informatik eine softwareorientierte Denkweise vor, welche auf die besonderen Anforderungen eingebetteter Systeme angepasst wird, z. B. *RealTimeAndEmbeddedSystems* als wählbares Modul in [ACM/IEEE, 2008]. Der Autor schlägt für künftige Analysen vor, das Computer Engineering Curriculm [ACM/IEEE, 2004a] und die Arbeiten der ARTIST Education Group [Artist Education Group, 2003] entsprechend zu berücksichtigen, da sich in diesen differenziertere Kompetenzen zur Entwicklung von eingebetteten Systemen identifizieren lassen, was zu einer feineren Kompetenzstruktur führt. Aufgrund der Fokussierung der ARTIST auf Masterstudiengänge sind dann je nach institutioneller Ausrichtung Anforderungen an die kognitive Dimension bei der Vermittlung von Kompetenzen entsprechend zu verringern.

Die Auswahl der Modulbeschreibungen der an KOMINA beteiligten Universitäten ist sinnvoll, da das Kompetenzstrukturmodell im EAP exemplarisch umgesetzt wird und sich dort in gegebene institutionelle Strukturen einfügt. Insbesondere eine Vorstellung der *Kompetenzen als Voraussetzung (K1)* ist kontextspezifisch und durch vorhergehende Module vorgegeben. Für die Auswahl weiterer Modulbeschreibungen wurden Universitäten gewählt, welche im Förderprogramm *Exzellenzinitiative des Bundes und der Länder zur Förderung von Wissenschaft und Forschung an deutschen Hochschulen* (vgl. [DFG, 2006]) in einem von drei Programmen gefördert wurden:

1. Exzellenzinitiative des Bundes und der Länder zur Förderung von Wissenschaft und Forschung an deutschen Hochschulen,
2. Exzellenzcluster zur Förderung der Spitzenforschung,
3. Zukunftskonzepte zum projektbezogenen Ausbau der universitären Spitzenforschung.

Für die Lehre zu eingebetteten Systemen sind also die Universitäten zu betrachten, welche auch einen Forschungsschwerpunkt im Bereich Informatik beziehungsweise technische Informatik oder eingebettete Systeme haben. Dies ist für die Modulbeschreibungen der RWTH Aachen (Aachen Institute for Advanced Study in Computational Engineering Science), TU München (International Graduate School of Science and Engineering) und im Bereich Nanosysteme die TH Karlsruhe (Center for Functional Nanostructures) der Fall.

Lehrbücher zu eingebetteten Systemen enthalten implizites fachdidaktisches Wissen, welches in didaktischen Forschungen genutzt werden kann. [Brinda, 2004]

und [Freischlad, 2010] nutzten dieses Wissen für die Entwicklung von didaktischen Systemen. Ziele eines Buchkapitels werden in Form einer Kompetenzbeschreibung beziehungsweise einer Erklärung, in welchem Kontext der folgende Input angewendet werden kann, expliziert. In der Entwicklung des NKSM wurde weder implizites noch explizites fachdidaktisches Wissen aus Lehrbüchern integriert, da sich diese oftmals auf wenige fachspezifische Aspekte beschränken und für eine erste Gliederung in Kompetenzdimensionen keinen über die Empfehlungen hinausgehenden Mehrwert bieten.

> „When searching for a textbook in embedded systems, this becomes more difficult. Every available textbook on embedded systems emphasizes different aspects. It is equally difficult to agree universally on curricula and courses in embedded systems [...]“ [Grimheden und Torngren, 2005, S. 642].

Für eine weitere Ausdifferenzierung des Kompetenzstrukturmodells ist die Integration von, auf spezielle Aspekte fokussierende, Lehrbüchern zu empfehlen (vgl. Kapitel 4.2).

3.2. Empirische Verfeinerung des normativen Kompetenzstrukturmodells

Eine empirische Überprüfung des normativ abgeleiteten Kompetenzstrukturmodells war notwendig, um die intersubjektiv hergeleitete Struktur von Kompetenzen hinsichtlich Vollständigkeit und Wichtigkeit zu validieren. Experten der technischen Informatik bewerteten die identifizierten Kompetenzen in einer zweistufigen schriftlichen Befragung. Die erste Expertengruppe bildeten Professoren an Hochschulen, die Zweite Professoren an Fachhochschulen. In den Publikationen zum Kompetenzmodell (vgl. [Schäfer et al., 2012a], [Schäfer et al., 2012b]) wurden die Ergebnisse beider Befragungen zusammengefasst, nicht aber differenziert betrachtet.

Die Auswahl der Kompetenzbeschreibungen für die empirische Verfeinerung wurde von den Experten des KOMINA-Projektes vorgenommen und dabei Kompetenzbeschreibungen verwendet, welche die Subdimensionen adäquat repräsentieren und die Breite der Facetten widerspiegeln (Fragebogen siehe Anhang A.2). Außerdem stand den Befragten ein freies Feld für Kommentare zu den aufgeführten Kompetenzen beziehungsweise zur Ergänzung fehlender Kompetenzen zur Verfügung.

3.2.1. Erste empirische Überprüfung des NKSM

In der ersten empirischen Verfeinerung wurden 96 Professoren deutscher Universitäten befragt, von denen 38 Antworten ausgewertet werden konnten. Dies entspricht einer Ausschöpfungsquote von ca. 40 % und belegt damit den Bedarf an

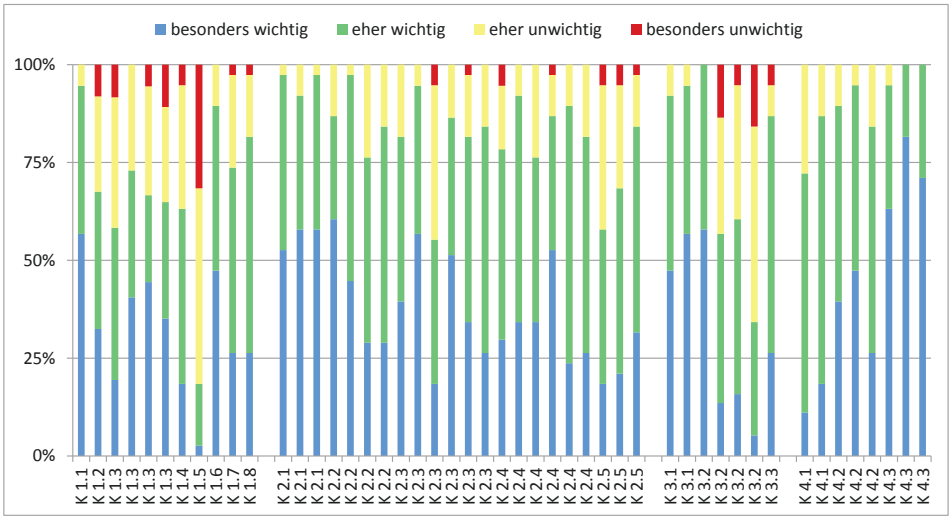


Abbildung 3.2.: Ergebnisse der ersten Expertenbefragung

Kompetenzforschung in der technischen Informatik und starkem Interesse an einer Mitwirkung.

In Abbildung 3.2 sind die Ergebnisse der ersten empirischen Überprüfung dargestellt. Auf der Y-Achse ist die prozentuale Verteilung auf die vier möglichen Antworten *besonders wichtig*, *eher wichtig*, *eher unwichtig* und *besonders unwichtig* aufgetragen. Auf der X-Achse sind die einzelnen Kompetenzbeschreibungen (Ankerbeispiele/Kompetenzfacetten) einer Subdimension, repräsentiert durch eine Beschreibung im Fragebogen (siehe Anhang A.2), aufgeführt.

Es ist zu erkennen, dass der überwiegende Teil der Kompetenzbeschreibungen als *besonders wichtig* und *eher wichtig* bestätigt wurde. Die Ausnahme bilden die Kompetenzsubdimensionen *(K1.5) Materialwissenschaften* und das vierte Ankerbeispiel zu *(K3.2) Bottom-Up Design* (die Studierenden verstehen die physikalischen Grundprinzipien neuer Basis-Nanobauelemente für die IT, wie Nanoröhrchen, Nanocrossbars oder Quantenzellularautomaten). Letzteres wurde von 5,3 % der Befragten als *besonders wichtig*, von 28,9 % als *eher wichtig*, von 50 % als *eher unwichtig* und von 15,8 % als *besonders unwichtig* bewertet. Ein Grund dafür ist, dass die hier genannten Technologien auf niedriger Abstraktionsebene heute (noch) eine geringe Relevanz für die Zielgruppe haben. Möglich ist aber auch, dass Kenntnisse in diesem Bereich auch in Zukunft nur den Experten anderer Fachrichtungen, wie der Chemie oder Physik, vorbehalten sind. Die Einschätzung der Befragten deckt sich in diesem Punkt also nicht mit den Experten des KOMINA Projektes.

(K1.5) Materialwissenschaften beurteilten 2,6 % der Befragten als *besonders wichtig*, 15,8 % als *eher wichtig*, 50 % als *eher unwichtig* und 31,6 % als *besonders unwichtig*. Es wird angenommen, dass der Oberbegriff einer Kompetenzsubdimension allein keine hinreichende Beschreibung für den Kontext – eingebettete Systeme – gibt. Intendiert war die Assoziation mit dem atomaren Aufbau der in Halbleiter gefertigten Hardware. Dieser Aufbau hat Auswirkungen auf die Eigenschaften und den fertigungsnahen Entwurf eingebetteter Systeme und wird insbesondere in Modulbeschreibungen beziehungsweise Curricula mit an Hardware orientierten Inhalten erklärt (vgl. [Universität Siegen, 2011]). Aufgrund dieser Ungenauigkeit wurde die Beschreibung im Fragebogen der zweiten Expertenbefragung um den Zusatz „insbesondere der atomare Aufbau“ erweitert (siehe Abbildung A.3).

Auffällig war außerdem eine sehr differenzierte Bewertung von *(K1.3) Informatik*. Verglichen mit den weiteren Subdimensionen aus *(K1) Kompetenzen als Voraussetzung*, welche mit dem Namen der Fachgebiete beschrieben wurden (Elektrotechnik, Englisch etc.), fiel die Bewertung einzelner Teilbereiche der Informatik unerwartet *geringer wichtig* aus. Im Hinblick darauf, dass informatische Grundlagen und Konzepte in jedem Bereich der Entwicklung eingebetteter Systeme von Relevanz sind, scheint die abweichende Formulierung anhand von Ankerbeispielen hier nicht sinnvoll. Um in den Folgearbeiten konsistent zu sein, wurde diese Subdimension in der zweiten Expertenbefragung auf den Namen der Fachdisziplin – Informatik – reduziert.

Das normativ entwickelte Modell wurde insofern bestätigt, als dass – von zwei Ausnahmen abgesehen – keine signifikant unwichtigen Kompetenzbeschreibungen enthalten sind. Teilnehmer kommentierten den Fragenkatalog wie folgt:

> „Eigentlich hätte ich bei allen Fragen ein a) geben können [...] Ein TI-ler muss doch alle genannten Fähigkeiten und Kompetenz möglichst optimal vereinen.“

> „Alle formulierten Kompetenzen werden als “notwendig“ angesehen. Es erfolgt eine tendenzielle Bewertung für die Punkte im Sinne von “besonders wichtig“ und “eher wichtig“ [Hervorhebungen im Original].“

Damit bestätigten sie das Kompetenzstrukturmodell im Ganzen. Sie schlagen implizit vor, lediglich eine Ordnung im Bereich *wichtig* vorzunehmen, da nach ihrer Auffassung alle genannten Kompetenzen wichtig seien. Weil aber eine Klassifizierung diverser Kompetenzbeschreibungen als *eher unwichtig* und *besonders unwichtig* erfolgte, wurde von diesem Vorgehen auch in der zweiten Befragung abgesehen.

Weitere Forschungsarbeiten zur Nanosystemtechnik und den damit verbundenen Kompetenzen zeigten, dass parallele Architekturen, welche in der Mikro- und Nanosystemtechnik vielfach Anwendung finden (Multiprozessortechnologien und damit parallele Prozesse), bislang keine ausreichende Ausprägung im Fragebogen fanden, wenngleich mehrere Kompetenzbeschreibungen bei der normativen Her-

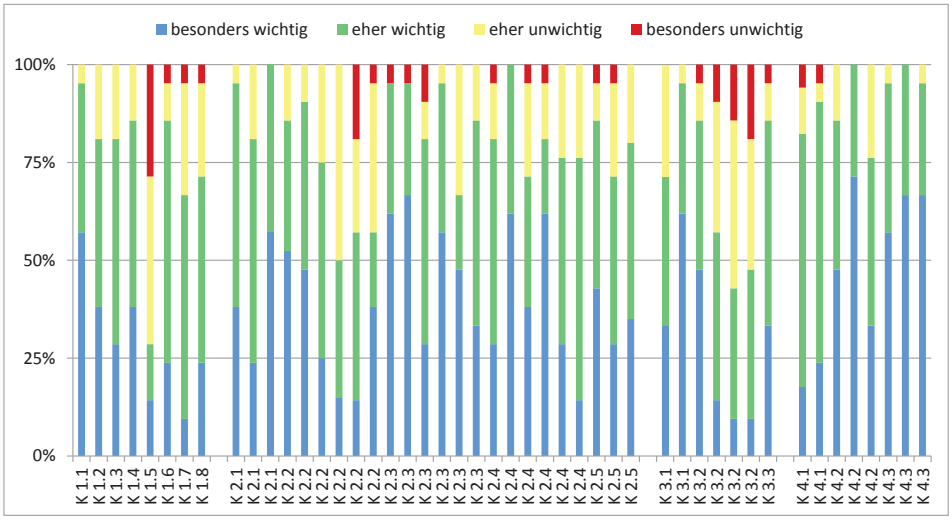


Abbildung 3.3.: Ergebnisse der zweiten Expertenbefragung

leitung identifiziert werden konnten. Damit fehlten im ersten Schritt wichtige, die Breite der Kompetenzsubdimension widerspiegelnde Aspekte, die in den Fragebogen aufgenommen wurden.

3.2.2. Zweite empirische Überprüfung des NKSM

In der zweiten empirischen Überprüfung wurde im Wesentlichen eine Bestätigung der bisherigen Ergebnisse erwartet. Der überarbeitete Fragebogen sollte in den zwei zuvor genannten Punkten zu einer insgesamt *positiveren* Bewertung führen. Es wurden 75 Professoren deutscher Fachhochschulen schriftlich befragt. Ausgewertet werden konnten 21 Antworten, was einer Rücklaufquote von 28 % entspricht und damit die Einschätzung zur Relevanz bestätigt.

In Abbildung 3.3 sind die Ergebnisse der zweiten empirischen Überprüfung analog zum Vorgehen in der ersten Expertenbefragung aufgeführt. Man erkennt, dass die Kompetenzbeschreibung *K1.3 Informatik* verglichen mit der ersten Expertenbefragung, nun lediglich durch den Namen repräsentiert, im Durchschnitt als *wichtiger* eingeschätzt wurde. Waren es zuvor ca. 65 % der Befragten, welche die Informatik als *eher wichtige* und *besonders wichtige* Voraussetzung für den Kompetenzerwerb zu eingebetteten Systemen sahen, liegt der Anteil in der zweiten Befragung bei 80 %. Keiner der Befragten bewertete die Vorkenntnisse in der Informatik als *besonders unwichtig*. Für weitere Arbeiten ist also festzuhalten, dass die Granularität von zu überprüfenden Kompetenzsubdimensionen konsistent innerhalb einer Dimension sein muss.

Trotz der erweiterten Beschreibung wurde *K1.5 Materialwissenschaften* von 60 % der Experten als *eher unwichtig* und *besonders unwichtig* bewertet. Die Ergebnisse der ersten Befragung wurden also bestätigt und die Kompetenz infolge dessen aus dem Modell entfernt. Eine Zustimmung erfolgte auch in den Kompetenzbeschreibungen zu *K3.2(4) Bottom-Up-Vorgehen*. Diese Facette wurde von 50 % der Experten als *eher unwichtig* und *besonders unwichtig* bewertet, verglichen mit den anderen Kompetenzfacetten aus *K3.2* also signifikant *weniger wichtig*. Da diese Kompetenzbeschreibungen, bezogen auf weitere Forschungsarbeiten zur Lehre im Bereich Nanosystemtechnik, im Fokus stehen, wird im Modell eine eigene Kategorie *Nanoeffekte* eingeführt. Dies ermöglicht bei einer weiteren Ausdifferenzierung die Integration neuer Bottom-Up-Technologien ohne die Flexibilität des Modells zu reduzieren. Hinsichtlich einer auf Informatikstudierende zugeschnittenen Lehre mit Grundlagenveranstaltungen zu eingebetteten Systemen ist nach Expertenmeinung dieser Bereich zunächst nicht in Lehr-Lernprozesse zu integrieren. Die Kompetenzbeschreibungen mit Bezug zu parallelen Prozessen wurden von 60 % der Befragten als *wichtig* bewertet und damit in das Modell aufgenommen.

Analog zur ersten empirischen Evaluation kommentierten zwei Experten die Befragung. Es wird eine für die Praxis wichtige Kompetenz gefordert, die über die Korrektheit einer Lösung hinausgeht und Produktorientierung hervorhebt. Ferner wurde kommentiert, dass durch das Modell ein Sollzustand dargestellt ist, der derzeit nicht verfolgt wird.

> „Die Studierenden lernen eine Codierungsrichtlinie kennen und wissen um die Relevanz bei der Erstellung produktorientierten Codes“.
>
> „Ich habe die Fragen insofern beantwortet, als dass Kompetenzen erlernt werden sollten. Derzeit ist dies in vielen Punkten nicht der Fall [Hervorhebungen im Original]“.

Mit wenigen Ausnahmen deckten sich die Einschätzungen beider Expertengruppen. In Tabelle 3.2 werden die Kompetenzdimensionen aufgeführt, bei denen Unterschiede von $> 10\%$ auftreten (erste Befragung/zweite Befragung). Zumeist ist lediglich eine Verschiebung innerhalb *wichtig* und *unwichtig* zu erkennen. Deutlichere Unterschiede existieren bei *(K1.4) Elektrotechnik*, welche 36,8 % der Universitätsprofessoren als *unwichtig* bewerteten. Hier wird angenommen, dass die Forschung zu eingebetteten Systemen zumeist Konzepte auf höheren Abstraktionsebenen fokussiert und die Elektrotechnik eine eher untergeordnete Rolle spielt (von Aspekten des Fertigungsprozesses abgesehen). Die Kompetenzsubdimension *(K2.2) Anforderungsanalyse* bewerteten 30,8 % der Fachhochschulprofessoren als *unwichtig* verglichen mit 13.8 % der Universitätsprofessoren. Da hier in der zweiten empirischen Überprüfung weitere, die Themen Echtzeitsysteme und Multiprozessortechnologien betreffende Facetten eingeführt wurden, welche gegenüber den Übrigen als *geringer wichtig* bewertet wurden, ist hier kein Vergleich möglich.

Tabelle 3.2.: Analyse beider Expertenbefragungen

Kompetenzen	besonders wichtig	eher wichtig	eher unwichtig	besonders unwichtig
K1.4	18,4 %/38,1 %	44,8 %/47,6 %	31,6 %/14,3 %	5,2 %/0,0 %
Elektrotechnik				
K1.5	2,6 %/14,3 %	15,8 %/14,3 %	50 %/42,9 %	31,6 %/28,6 %
Materialwissenschaften				
K1.6	47,4 %/23,8 %	42,1 %/61,9 %	10,5 %/9,5 %	0 %/4,8 %
Englisch				
K1.7	26,3 %/9,5 %	47,4 %/57,1 %	23,7 %/28,6 %	2,6 %/4,8 %
Wissenschaftliches Arbeiten				
K2.2	40,8 %/32,0 %	45,4 %/37,2 %	13,8 %/26,8 %	0,0 %/4,0 %
Anforderungsanalyse				
K2.5	23,7 %/35,5 %	46,5 %/43,6 %	25,4 %/17,8 %	4,4 %/3,2 %
Optimierung und Test				
K4.1	14,8 %/20,7 %	64,8 %/65,7 %	20,5 %/8,3 %	0,0 %/5,3 %
Einstellungen				
K4.2	71,9 %/63,5 %	26,3 %/33,3 %	1,8 %/3,2 %	0,0 %/0,0 %
Sozial-kommunikative Kompetenzen				

Nach entfernen von *(K1.5)* und der Aufteilung von *(K3.2.1)* wurde eine Klassifizierung der Kompetenzen für eine bessere Unterscheidbarkeit vorgenommen. Die Klasse *wichtig* wurde in drei Bereiche *A-C* und die Klasse *unwichtig* in die Bereiche *D-F* aufgeteilt. Die Einteilung in diese sechs Klassen erfolgte nach folgendem Schema (vgl. [Schäfer et al., 2012a]):

Klasse A: Die überwiegende Anzahl von Experten bewertete diese Facetten als *besonders wichtig*.

Klasse B: Die überwiegende Anzahl von Experten bewertete diese Facetten als *wichtig*. Die Anzahl der Bewertungen *besonders wichtig* ist größer als die Anzahl der Bewertungen *unwichtig*.

Klasse C: Die überwiegende Anzahl von Experten bewertete diese Facetten als *wichtig*. Die Anzahl der Bewertungen *unwichtig* ist größer als die Anzahl der Bewertungen *besonders wichtig*.

Klasse D: 50 % oder mehr der Experten bewerteten diese Facetten als *unwichtig*. Die Anzahl der Bewertungen *wichtig* ist größer als die Anzahl der Bewertungen *besonders unwichtig*.

Klasse E: 50 % oder mehr der Experten bewerteten diese Facetten als *un-*

wichtig. Die Anzahl der Bewertungen *besonders unwichtig* ist größer als die Anzahl der Bewertungen *wichtig*.

Klasse F: Die überwiegende Anzahl der Experten bewertete diese Aspekte als *besonders unwichtig*.

Zusammenfassung

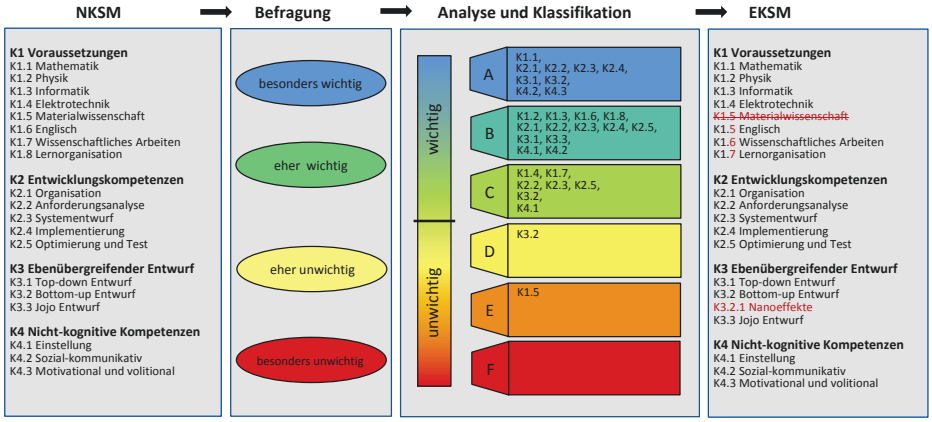


Abbildung 3.4.: Entwicklung des empirisch verfeinerten Kompetenzstrukturmodells [Schäfer et al., 2012a, S. 59]

In Abbildung 3.4 werden die Forschungsschritte vom normativen Kompetenzstrukturmodell zum empirisch verfeinerten Kompetenzstrukturmodell von links nach rechts zusammenfassend dargestellt, beginnend mit der normativ hergeleiteten Gliederung von Kompetenzdimensionen. In der folgenden zweistufigen empirischen Verfeinerung wurden die identifizierten Kompetenzen bewertet. Nach der Analyse der Ergebnisse wurde eine weitere Unterteilung der Facetten in sechs Kategorien vorgenommen (*A-F*). Die Kompetenzsubdimension *(K1.5) Materialwissenschaften* wurde aus dem Modell entfernt und die Nummerierung angepasst. Die Subdimension *(K3.2) Bottom-Up* wurde weiter differenziert und eine eigene Kategorie für Kompetenzen mit nanoskalinen Bauteilen erweitert.

> „[...] They understand the physical first principles of new basic nano assembly parts for the IT, like nano tubes, nano crossbars or quantum cellular automata“ [Schäfer et al., 2012a, S. 60].

Zur exemplarischen Umsetzung des Kompetenzstrukturmodells in einem Entwurfs- und Anwendungspraktikum wurden die Kompetenzen der *Klasse A* erforscht, welche dem Studienverlauf und der institutionellen Ausrichtung der Universität Siegen

entsprechen. Da die betrachteten Facetten nicht durchgängig einen Operator der Taxonomie nach [Anderson et al., 2000] enthalten, wird analog zum Vorgehen in Kapitel 2, in nicht eindeutigen Fällen, eine Beschreibung auf einer niedrigeren kognitiven Stufe vorgenommen (vgl. [Schäfer et al., 2012a]).

Tabelle 3.3.: Kompetenzen in der Klasse A des EKSM (vgl. [Schäfer et al., 2012a])

Kompetenzen	**besonders wichtig**	**eher wichtig**	**eher unwichtig**	**besonders unwichtig**
K1.1	57,9 %	38,6 %	3,5 %	0 %
Mathematik				
K2.1	56,9 %	41,4 %	1,7 %	0 %
Die Studierenden erlernen eine zielgerichtete und strukturierte Vorgehensweise beim Entwurf.				
K2.2	56,9 %	29,3 %	13,8 %	0 %
Die Studierenden kennen die besonderen Randbedingungen des Entwurfs eingebetteter Systeme.				
K2.3	61,4 %	35,1 %	3,5 %	0 %
Die Studierenden erwerben elementare Kenntnisse über die wichtigsten Technologien und Konzepte, die für den Entwurf und die Analyse rechnergestützter Systeme benötigt werden.				
K2.3	54,4 %	36,8 %	8,7 %	0 %
Die Studierenden verstehen den Aufbau und die Funktion aller wichtigen Grundschaltungen und Rechenwerke.				
K2.4	56,9 %	29,3 %	12,7 %	1,7 %
Die Studierenden können Schaltungen mittels einer Beschreibungssprache entwerfen.				
K3.1	59,7 %	36,9 %	3,5 %	0 %
Die Studierenden erlernen den Umgang mit der Programmiersprache C und deren Zusammenspiel mit der Hardware. Dabei werden grundlegende Funktionalitäten und das Zusammenspiel der Basiskomponenten eines Betriebssystems mit dem Schwerpunkt auf effiziente Ressourcenverwaltung vermittelt.				
K3.2	55,2 %	41,4 %	1,7 %	1,7 %
Die Studierenden sind in der Lage den Zusammenhang zwischen Hardware-Konzepten und den Auswirkungen auf die Software zu verstehen, um effiziente Programme erstellen zu können sowie einen Rechner aus Grundkomponenten aufbauen zu können.				
K4.2	56,9 %	39,7 %	3,4 %	0 %
Die Studierenden können im beruflichen Umfeld mit Ingenieuren und Elektrotechnikern als Informatikanwender kompetent kommunizieren.				
K4.3	62,1 %	32,8 %	5,2 %	0 %
Die Studierenden besitzen Offenheit für neue Ideen /Anforderungen				
K4.3	77,6 %	22,4 %	3,5 %	0 %
Die Studierenden besitzen Lernbereitschaft.				
K4.3	70,7 %	27,6 %	1,7 %	0 %
Die Studierenden besitzen Bereitschaft zum Engagement.				

3.3. Exemplarische Umsetzung des EKSM im Entwurfs- und Anwendungspraktikum

Das empirisch verfeinerte Kompetenzstrukturmodell bietet eine Vorstellung über die zu fördernden Kompetenzen künftiger Entwickler eingebetteter Systeme. Für die exemplarische Umsetzung des Modells wurde das an der Universität Siegen angebotene Modul Hardwarepraktikum (HaPra) neu gestaltet. Im Rahmen dieser Neugestaltung wurde geprüft, inwiefern die Ergebnisse aus den Arbeiten zur Kompetenzmodellierung genutzt werden können, um eingebettete Systeme als in der analog kontinuierlichen Realität verankerte Systeme erfahrbar zu machen [Schubert et al., 2010, S. 14]. Aufgrund sich verändernder institutioneller Randbedingungen, wodurch seit dem Jahr 2008 eine heterogenere Studierendengruppe im vierten Semester des Bachelorstudiums dieses Modul absolvierten, war sowohl der Zeitpunkt als auch die Zielgruppe für die Erforschung von Laborversuchen äußerst günstig.

Das HaPra bestand aus zwei Abschnitten (Versuche 1-3 und 4-10), welche nach zwei einführenden Vorlesungen in Laborpraxiseinheiten (Präsenz) durchgeführt wurden. Im ersten Veranstaltungsabschnitt wurden elektronische und physikalische Randbedingungen und Effekte mit Simulationen und Werkzeugen der elektrischen Messtechnik (Oszilloskop, Funktionsgenerator) untersucht. Widerstände und Kondensatoren als elektrotechnische Basisbauteile wurden in verschiedenen Schaltungen verknüpft, um real auftretende Effekte in Halbleiterbausteinen oder nachrichtentechnischen Anlagen in labortauglichen Maßstäben nachzubilden. Im zweiten Teil des HaPra wurde ein Prozessor, basierend auf den theoretischen Grundlagen des Moduls *Schaltwerke und Rechnerorganisation*, mit VHDL entwickelt und auf einem FPGA implementiert. Industrielle Werkzeuge wurden für den computerunterstützten Entwurf und die Synthese verwendet, um eine praxisnahe Lernumgebung zu bieten. Dieser in Teilen selbstentwickelte Prozessor wurde abschließend synthetisiert und in einer Assemblersprache programmiert [Brück et al., 2008].

3.3.1. Problemlage und Lösungsansätze

Die Lehrveranstaltung HaPra war seit 1997 Bestandteil des Curriculums an der Universität Siegen, zunächst im Diplomstudiengang technische Informatik, anschließend im Diplomstudiengang *angewandte Informatik mit Anwendungsfach Elektrotechnik*. Seit Einführung der Bachelorstudiengänge im Bologna-Prozess im Jahre 2008 ist das Praktikum nunmehr teil des Bachelorstudiengangs angewandte Informatik – unabhängig vom gewählten Anwendungsfach. Praktika dieser Art sind im Rahmen eines technisch orientierten Informatikstudiums üblich und werden als geeignet angesehen (vgl. [Feisel und Rosa, 2005], [Nooshabadi und Garside, 2006], [Bouldin, 2004]). Durch die Verortung in allen Bachelorstudiengängen der Informatik ist die einst hinsichtlich der Vorkenntnisse homogene Studieren-

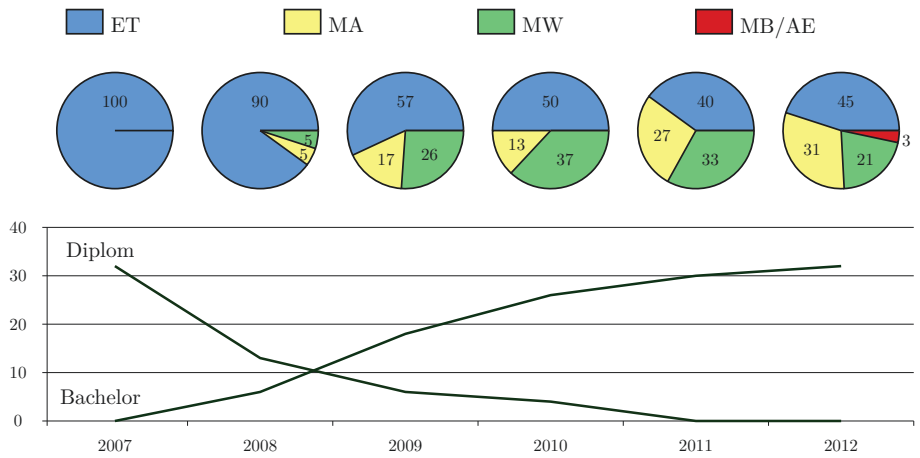


Abbildung 3.5.: Zielgruppe – Anwendungsfächer und Studiengänge [Jaschke et al., 2012, S. 2f]

dengruppe im Laufe der Jahre heterogener geworden (siehe Abbildung 3.5). Im oberen Teil ist die Verteilung der Nebenfächer der Studierenden aufgetragen, im unteren Bereich sind die absoluten Studierendenzahlen im Bachelor- und Diplomstudiengängen dargestellt. Zwar bildeten Studierende mit dem Nebenfach Elektrotechnik (ET) noch immer die größte Teilnehmergruppe (2012), dennoch belegten mehr als die Hälfte der Studierenden eines der Anwendungsfächer Mathematik (MA), Medienwissenschaften (MW), Maschinenbau (MB) oder Automotive Engineering (AE). Bei Betrachtung der unterschiedlichen Informatikdisziplinen (siehe Abbildung 1.2 in Kapitel 1) wird deutlich, dass die gemeinsame Schnittmenge der Studierenden durch die heterogenen Nebenfächer gering ist und die Hardwareebene zumeist nicht umfasst. In den Erfahrungsberichten der Lehrenden ließen sich drei Lernhürden identifizieren [Brück et al., 2008], welche miteinander in Verbindung stehen und zu einer geringen Studierendenmotivation zu Beginn und im Verlauf der Veranstaltung führen – der Einsatz professioneller Entwurfssoftware, die Bottom-Up-Vorgehensweise und die Heterogenität der Studierenden.

Einsatz professioneller Entwurfssoftware

Im Hardwarepraktikum wurde die Software *FPGA Advantage* eingesetzt, um einen einfachen Prozessor zu entwickeln und diesen für das *Xilinx Spartan 3 Board* zu synthetisieren. Dabei handelt es sich um eine sehr komplexe, da multifunktionale Software, deren Funktionsumfang auch den Tutoren nicht in vollem Umfang bekannt sein kann. Selbst Entwickler in der Industrie benötigten Monate zur effizienten Nutzung dieser Entwurfswerkzeuge [Brück et al., 2008, S. 176]. Durch

diese Funktionsvielfalt ist eine Orientierung bei der Navigation und Nutzung der verschiedenen Eingabe- und Konfigurationsmasken für Studierende schwierig. Zur Vorbereitung auf die jeweils nächste Laboreinheit und Nachbereitung der vergangenen Praxisphase war es den Studierenden nicht möglich ihre Erfahrungen durch Wiederholen, Testen und Erweitern der entwickelten Lösung zu manifestieren. So war ein erneutes Einarbeiten zu Beginn jedes Versuches nötig. Die Betreuer stellten im Verlauf fest, dass diejenigen Studierenden am erfolgreichsten waren, die sich strikt an die Aufgabenstellungen hielten und auf eigene Lösungsansätze verzichteten. Erfolgreich bedeutet in diesem Zusammenhang, dass die Aufgaben im vorgegebenen Zeitraum bewältigt wurden. Ziel einer Lehrveranstaltung ist aber nicht die Fertigstellung eines Produktes, sondern vielmehr der Kompetenzzugewinn. Die Entwicklung eigener Lösungsansätze, verbunden mit deren Überprüfung in praxisnahen Aufgabenstellungen, sind Ziele in projektorientierten Laborpraktika. Lässt sich also eine Aufgabe im vorgegebenen Zeitrahmen nur dann lösen, wenn nach einer Schritt-für-Schritt-Anleitung, einem Tutorial, Operationen durchgeführt werden und ein Abweichen von dieser strikten Reihenfolge zum Scheitern im Praktikum oder einem erheblich erhöhten Zeitaufwand führen, so ist eine Kompetenzentwicklung auf Basis von eigenen Erfahrungen und Erarbeiten von Problemlösestrategien nur in Grenzen möglich. Es kann insoweit aus didaktischen Gesichtspunkten nicht von Erfolg gesprochen werden.

Bottom-Up-Vorgehensweise

Studierende, die das Tutorial nach Anweisung durchführten, hatten die besten Chancen die Laborphasen in den vorgegebenen Zeiteinheiten zu absolvieren. Ein Grund dafür ist, dass sich durch das Bottom-Up-Vorgehen im zweiten Praktikumsabschnitt – Entwicklung des Rechenwerkes, des Prozessors, der Schnittstellen sowie Programmierung des Prozessors in Assembler – Fehler in frühen Phasen der Entwicklung in die folgenden Schritte fortpflanzen. Dies kann in der Praxis durch frühzeitige Simulations- und Testphasen vermieden werden. Im Praktikum aber kam es in den Durchführungsphasen zu Situationen, in denen erst in den letzten Versuchen – der Assemblerprogrammierung – Fehler im Prozessordesign entdeckt wurden. Diese späten Tests und die unter Umständen aufwändige Fehlerbehebung bedingen ein Vielfaches der vorgesehenen Präsenzzeit.

Die Ergebnisse einer Laborveranstaltung sind Voraussetzung für einen erfolgreichen Abschluss des nachfolgenden Abschnittes. Dies hatte zur Konsequenz, dass die Studierenden statt eigene zeitaufwändige Lösungsstrategien anzuwenden, sich strikt am Tutorial und an Musterlösungen der Kommilitonen orientierten.

Heterogenität der Studierenden

Insbesondere Studierende nicht-technischer Anwendungsfächer (Mathematik und Medienwissenschaften) erkannten nicht den „Sinn der Veranstaltung“, da diese ihr Studienziel in der Entwicklung von Anwendungssoftware sehen. Der Entwurf von

Prozessoren und die Programmierung in Assembler seien für die späteren Karrieren nicht relevant. [Brück et al., 2008] bestätigen dies, da die zugrunde liegende Hardwareplattform bei klassischen Informatiksystemen nahezu keine Rolle mehr spiele. Diesem Argument aber stehen die Entwicklungen populärer eingebetteter Systeme als Teilmenge der Informatiksysteme entgegen. Je nach Einsatzbereich, z. B. Anwendungssoftware von Smartphones, nehmen die Hardware und die Schnittstelle zu physikalischen Prozessen wieder an Bedeutung zu. Geringer Speicher, dedizierte Hardware und Betriebssysteme sowie eingeschränkte Energiekapazitäten, Sensoren und Aktoren erfordern auch bei Design und Implementierung auf höherer Abstraktionsebene ein Bewusstsein für die darunterliegenden Schichten.

3.3.2. Konzeptideen zu einem Entwurfs- und Anwendungspraktikum Mikrosysteme

Um den zuvor aufgeführten Problemen zu entgegnen, diskutierten [Brück et al., 2008] erste Konzeptideen für eine Umgestaltung des HaPra zu einem Entwurfs- und Anwendungspraktikum, welche auch im KOMINA-Projekt Anwendung fanden. Im Wesentlichen wird vorgeschlagen, zu eingebetteten Mikrosystemen mit explizitem Hardwarebezug überzugehen [Brück et al., 2008, S. 177]. Diese sollen mit einer hinreichenden Nähe zu elektrisch/physikalischen Problemstellungen die Motivation der Studierenden steigern. Nach dem zielorientierten Ansatz nach [Weicker, 2007], in dem fünf Aspekte miteinander vereint werden (Lernzielorientierung, Anknüpfung an Vorwissen, enge Verzahnung von Input und Aktivität, Motivierung und frühzeitiges Feedback bzgl. aller Lernbereiche) identifizierten [Brück et al., 2008] drei Lösungsansätze – Ausrichtung an Alltagserfahrungen, Top-Down-Vorgehen und Peer-Review in der Projektphase.

Ausrichtung an Alltagserfahrungen

Als motivierender Aspekt sollen eingebettete Mikrosysteme aus dem Alltag von Informatikern als Systementwickler aber auch als Bürger, also Anwender, statt des frei programmierbaren FPGA als Entwurfsgegenstand dienen. Einbezogen werden sollten Aktor- und Sensoreinheiten an Schnittstellen eines Mikrocontrollers sowie die zugehörigen Kommunikationseinheiten. Die Verwendung autonomer Roboter als Lehrgegenstand bietet sich für den Einsatz von Sensoren und Aktoren an, welche in einer Wettbewerbssituation motivieren könnten. Als weitere Lehrgegenstände werden Systeme für Umweltüberwachungen und -analysen sowie Steuerungs- und Automatisierungssysteme im Haushalt genannt [Brück et al., 2008].

Autonome Roboter erfreuen sich in universitären Lehrveranstaltungen zu eingebetteten Systemen sowie in der Sekundarstufe II an allgemeinbildenden Schulen großer Beliebtheit. Das liegt unter anderem daran, dass sich Steuerungs- und Automatisierungsaufgaben unterschiedlicher Komplexität stellen lassen und direkt eine haptische und beobachtbare Erfahrung mit der individuellen Lösung der Stu-

dierenden einhergeht. Vielfach eingesetzt werden Lego Mindstorms[1], welche sich durch das einfache Einstecken von Sensoren und Aktoren und der individuell zu gestaltenden äußeren Form vielfältig einsetzen lassen. Das Problem bei der Verwendung in Laborpraktika mit direktem Anwendungsbezug liegt darin, dass von den niederen Ebenen des Systementwurfs abstrahiert wird und die Hardware und Funktionsweise von Sensoren nicht im Zentrum stehen. Besser geeignet für ein Laborpraktika hinsichtlich der Lebenswelt der Studierenden scheinen daher einfache Steuerungs- und Automatisierungssysteme im Haushalt, welche einfach nachzuvollziehen sind, da sie aus wenigen Basisbauteilen aufgebaut werden können.

Als wesentlich für die Akzeptanz des Praktikums seitens der Studierenden wird die Integration von Smartphones zur Visualisierung, Steuerung und Regelung angesehen. Smartphones sind in unserer Gesellschaft allgegenwärtig und werden auch in Lehrveranstaltungen vielfach aufgrund ihres Lebensweltbezuges eingesetzt. Dabei muss zwischen dem Einsatz als Lerngegenstand zur Förderung von Kompetenzen zur Entwicklung eingebetteter Systeme oder zur Förderung von Kompetenzen in der Softwareentwicklung unterschieden werden – spezifische Konzepte und Funktionen von Android. [Muppala, 2011] vergleicht fachdidaktische Publikationen und stellt fest, dass die meisten Autoren letzteres Ziel verfolgen und nur wenige Konzepte eingebetteter Systeme, wie Echtzeitbetriebssysteme, Hardware/Software Codesign vermitteln. Daher schlägt [Muppala, 2011] ein Konzept vor, welches im Wesentlichen die Punkte *Embedded software development, RTOS and Timeliness Issues, Interaction with the real world, Memory Management und Testing sowie Debugging* beinhaltet. Bei genauerer Betrachtung des Inhaltes ist zu erkennen, dass die Androidumgebung nur unzureichend zur Förderung von Kompetenzen zu diesen Bereichen geeignet ist. Zum einen ist eine intensive Einarbeitung in spezifische Konzepte notwendig, wie die Entwicklungsumgebung, die Benutzungsschnittstelle und Multimediasupport. Zum anderen abstrahiert das Betriebssystem größtenteils von den Problemen auf den unteren Ebenen und übernimmt die Speicherverwaltung, das Ansprechen von Sensoren und Aktoren sowie die Zeitablaufsteuerung. Hier muss abgewogen werden, ob die Motivation von Studierenden, welche sich nachweislich durch den Einsatz von Android steigern lässt, eine Reduktion der Zeit für die Förderung von spezifischen Kompetenzen zur Entwicklung eingebetteter Systeme rechtfertigt.

Top-Down-Vorgehen

Ein nicht zielführendes, rein auf Bottom-Up-Vorgehensweise basierendes Praktikum wurde durch die geschilderte Problemlage, also einer schwierigen Fehlersuche bei der Identifikation in späten Entwurfsphasen, fehlendem Blick für die Sinnhaftigkeit und einem explorationsunfreundlichen Setting identifiziert. Es wird vorgeschlagen, zunächst das Gesamtsystem (Sensoren, Aktoren, Steuerung) zur Lösung des Anwendungsproblems zu beschreiben und daran die Analyse ausgewählter

[1] Produkte des Spielwarenherstellers Lego [http://mindstorms.lego.com/, Abruf: 12/2014].

Funktionseinheiten und den physikalischen Effekten anzuschließen [Brück et al., 2008].

Die Herausforderung bei einer ausschließlichen Top-Down-Vorgehensweise ist, dass hohe analytische Fähigkeiten zur Zerlegung des Problems in Teilprobleme notwendig sind. Für das ganzheitliche Design von eingebetteten Systemen ist dies zweifelsohne wichtig und eine typische Herangehensweise, für ein Praktikum in der ersten Studienphase jedoch nur bedingt geeignet. Bislang sind den Studierenden keine Lösungsansätze bekannt und die niederen Schichten eines Informatiksystems werden noch als Black-Box betrachtet. Insbesondere Studierende ohne Vorwissen in der Elektrotechnik begreifen Informatiksysteme als digitale Systeme, für die sie Software entwickeln, nicht jedoch als analog/digital gemischte Systeme.

Passender erscheint eine Einführung in die niederen Schichten, bei bereits vorgegebenen Ziel, sodass die Studierenden sich Schritt für Schritt durch Erweitern des Bauteilevorrates zu einem funktionierenden Subsystem heranarbeiten (Bottom-Up). Wichtig ist dann, dass die zuvor identifizierten Randbedingungen und physikalischen Eigenschaften in der Implementierung wiederzufinden sind. Hier kann ein Erkenntnisgewinn geschaffen werden, wenn eine trivial wirkende Lösung nach der Implementierung aufgrund der physikalischen Eigenschaften nicht korrekt ist und erweitert werden muss. Sinnvoll ist daher das Bottom-Up-Vorgehen mit einer Rückkopplung in späteren Versuchsreihen. Bedeutend für die Motivation bleibt, dass die einmal erstellte und als richtig deklarierte Lösung in den folgenden Laboreinheiten fortbesteht und gegebenenfalls nur erweitert oder modifiziert werden muss.

Peer-Review in der Projektphase

[Brück et al., 2008] schlagen vor, die erstellten Hardwarebeschreibungen in frühen Phasen des Projektes ausführlich zu dokumentieren und webbasiert in einem Peer-Review-Verfahren von den anderen Studierenden analysieren und bewerten zu lassen.

Dabei werden auch Kompetenzen zum wissenschaftlichen Arbeiten (Dokumentation des Systems) und sozial-kommunikative Kompetenzen durch Feedback (geben und nehmen) gefördert. Eine Herausforderung ist dabei die (noch) fehlende Kompetenz zur Analyse der Systeme. Die Studierenden haben zumeist keine Erfahrungen in der Modellierung und Programmierung, sodass eine Analyse des Designs anderer Gruppen eine sehr hohe kognitive Anforderung darstellt und die Studierenden gegebenenfalls überfordert. Dieses Vorgehen eignet sich demnach insbesondere in Nachbereitungsphasen und späteren Versuchen.

3.3.3. Entwicklung lernförderlicher Experimente – Design des Entwurfs- und Anwendungspraktikums

Für die neue Laborveranstaltung wurden im Rahmen von KOMINA acht aufeinander aufbauende Experimente (Laborphasen) mit je einer Vor- und Nachbereitungsphase entwickelt, welche einige der wichtigsten Kompetenzen gemäß EKSM unter Berücksichtigung der institutionellen Besonderheiten fördern sollen (vgl. Abschnitt 3.2).

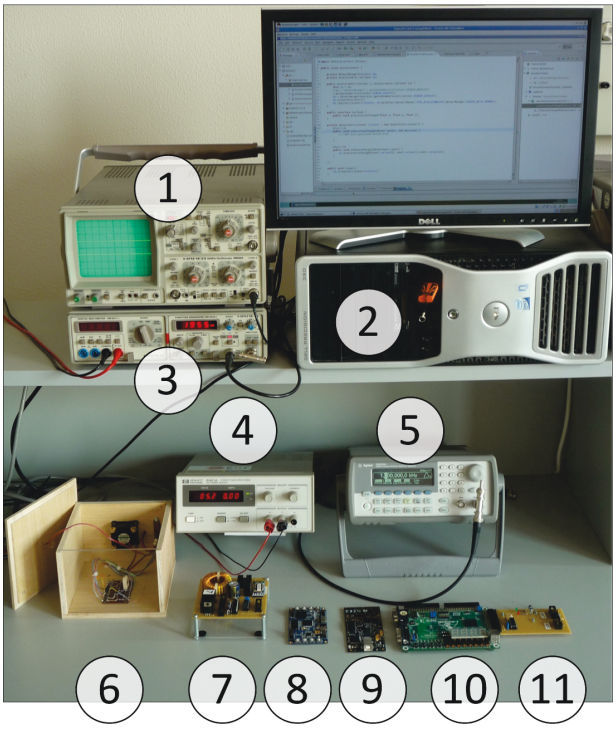


Abbildung 3.6.: Laborausstattung je Studierendengruppe [Büchner und Jaschke, 2013, S. 172]

Hauptaufgabe der Studierenden ist die Implementierung von Steuerungs- und Regelungsfunktionen für ein Modellhaus auf einen FPGA (vorwiegend Studierende mit Anwendungsfach Elektrotechnik) oder Mikrocontroller (übrige Anwendungsfächer). Diese Funktionen sollen via Smartphone gesteuert und visualisiert werden. Damit wird ein Bezug zur Lebenswelt der Studierenden hergestellt. Die Programmierung von Smartphones dient im Praktikum zur Förderung der Motivation (siehe Abschnitt 3.3.1).

Im Modellhaus (Abbildung 3.6, Nummer 6) sind verschiedene Sensoren und Ak-

toren integriert – Temperatursensoren innen/außen, Peltier-Element heizen/kühlen, Lichtsensor, Leuchtdiode, Feuchtigkeitssensor, Erschütterungssensor. Um den Studierenden das Ziel ihres Praktikums neben der kurzen Projektbeschreibung im Skript zu veranschaulichen, wurde zu Beginn ein Modellhaus mit allen zu implementierenden Funktionen demonstriert und per Smartphone gesteuert.

Bei der Wahl eines geeigneten Mikrocontrollers für das Entwurfs- und Anwendungspraktikum sollten unter anderem folgende Kriterien erfüllt werden:

- Unterstützung aktueller Konzepte (z. B. Direct Memory Access (DMA)),
- Programmierbar via Joint Test Action Group (JTAG) Schnittstelle,
- Ein-/Ausgänge auf Stiftleisten nach außen geführt (Anschluss an das Modellhaus),
- Integrierte Sensoren/Aktoren (Taster/LEDs),
- Integrierte Analog/Digital-Wandler und Digital/Analog-Wandler,
- Timer mit integriertem Signalgenerator für Pulsweitenmodulation.

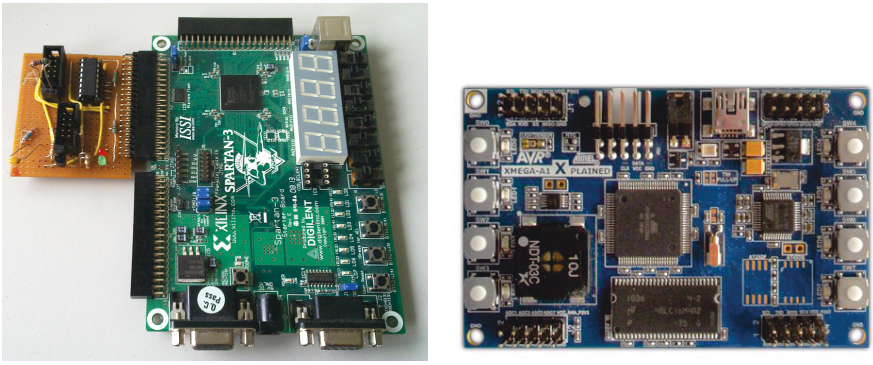
Sehr populär ist der Einsatz der Arduino Produktfamilie mit integriertem AVR Mikrocontroller. Diese ermöglicht durch eine integrierte Entwicklungsumgebung und Quelltextbibliotheken eine Programmierung mit einfachen, an die Arduino Plattform angepassten, Operationen in einem C-Dialekt (vgl. [Brand et al., 2011], [Sarik und Kymissis, 2010]). Adapterboards erlauben eine Erweiterung des Mikrocontrollers um applikationsspezifische Sensoren und Aktoren. Für die Sekundarstufe an allgemeinbildenden Schulen wird außerdem eine Anbindung an die grafische Programmierumgebung Scratch[2] angeboten. Beide Varianten abstrahieren von einer hardwarenahen Programmierung. Jede Funktionalität muss über zusätzliche Erweiterungen bereitgestellt werden (LEDs, Taster). Einen integrativen Ansatz bieten die *AVR MCU Xplained Kits*, welche bereits mit diversen Sensoren und Aktoren (Temperatursensor, Helligkeitssensor, Taster, LEDs, Lautsprecher) bestückt sind und nicht verwendete *I/O-Pins* über Stiftleisten nach außen führen. PWM-Kanäle, DMA und JTAG sind integriert. Letztlich wurde der *AVR ATxmega128A1 Xplained* auch aufgrund des modernen, gut dokumentierten Prozessors ausgewählt, da dieser ohne zusätzliche Bauteile für einfache Steuerungs- und Regelungsaufgaben geeignet und dennoch erweiterbar ist.

3.3.4. Laborausstattung und Lerngegenstand

Die Studierenden sollen im Verlauf des Praktikums Spannungen und Ströme an verschiedenen Bauteilen und Messpunkten der Schaltung ermitteln. Dafür steht ihnen ein Oszilloskop ① und ein digitales Multimeter ③ zur Verfügung. Ein La-

[2] Erziehungsorientierte visuelle Programmiersprache [http://scratch.mit.edu/, Abruf: 12/2014].

bornetzteil ④ liefert Gleich- und Wechselspannung und ist in der Strombegrenzung einstellbar, wohingegen der Funktionsgenerator ③ dazu verwendet wird, um Sinus-, Dreieck- oder Rechtecksignale zu erzeugen. Komplexere Signalverläufe können mit dem Arbiträrgenerator ⑤ in die Schaltung eingespeist werden. In der Implementierungsphase werden Sensoren ⑥ und Aktoren ⑦ des Modellhauses ⑥ ausgelesen beziehungsweise angesteuert. Dazu stehen den Studierenden das Mikrocontrollerboard ⑧, welches mit dem AVR-Dragon[3] ⑨ programmiert wird, zur Verfügung. Außerdem ist das *Spartan 3 FPGA Board* ⑩, das aufgrund fehlender analoger Eingänge mit Analogdigitalwandlern ⑪ erweitert wurde, vorhanden. Der *Virtual Workspace* (siehe Abschnitt 3.3.5) wird den Studierenden als USB-Stick ② ausgeliefert [Büchner und Jaschke, 2013].



(a) Xilinx Spartan 3 (inklusive Analog/ Digital-Wandler)

(b) AVR ATxmega128A1 Xplained

Abbildung 3.7.: Entwicklungsboards

Sollten die Studierenden ohne Anwendungsfach Elektrotechnik mehr Zeit zur Bewältigung der physikalisch-technischen Einführung benötigen, so war mit einem Puffer durch die zu erwartenden Kenntnisse bei der Implementierung in C zu rechnen. Gleich zwei neue Gebiete zu erschließen (Elektrotechnik und FPGAs) würde die Studierenden überfordern. Des Weiteren bleibt den Studierenden die Sinnhaftigkeit von VHDL für ihre persönliche Lernbiografie verborgen, da davon auszugehen ist, dass sie im späteren Berufsleben (Studierende mit Nebenfach Medienwissenschaften) diese Konzepte nicht benötigen werden. Das *Xilinx Spartan 3 FPGA-Board* (Abbildung 3.7) wurde auch bisher im HaPra verwendet und durch Analog/Digital-Wandler für das neue Entwurfs- und Anwendungspraktikum, also eine Schnittstelle zur physikalischen Realität, erweitert.

[3]Der AVR Dragon ist ein Entwicklungswerkzeug für 8-Bit und 32-Bit AVR Controller [http://www.atmel.com/tools/avrdragon.aspx, Abruf: 12/2014].

Praktikumsstruktur

Die Laborversuche wurden Bottom-Up strukturiert (Abbildung 3.8). In der physikalisch-technischen Einführung (Versuch 1-4) lernen die Studierenden die Basisbauteile der Elektrotechnik sowie die Grundschaltungen (Parallel, Reihe, Hochpass, Tiefpass) kennen und aufbauen. Anschließend werden einfache, später auch im Modellhaus verwendete, Sensoren in die Basisschaltungen integriert und Messwerte mithilfe des Multimeters und Oszilloskopes erfasst. Aktoren werden mittels Pulsweitenmodulation und einem Transistor als Schalter für größere Lasten im Modellhaus (Peltier-Element) angesteuert. Nach dieser Einführung sollten die Studierenden in der Lage sein einfache Schaltungen zu verstehen und aufzubauen. In den folgenden Versuchen zur Implementierung (Versuche 5-8) werden Basisoperationen, Bitoperationen, Schalten von Ausgängen und Lesen von Eingängen durch Registerzuweisungen durchgeführt. Anschließend wird die elektrische Spannung an Sensoren gemessen und in lesbare Werte transformiert, bevor zuletzt auch Aktoren via Pulsweitenmodulation angesteuert werden. Der letzte Versuch verbindet Sensorik, Aktorik und die Umgebung durch Steuerung und Regelung der Prozesse zu einem eingebetteten System.

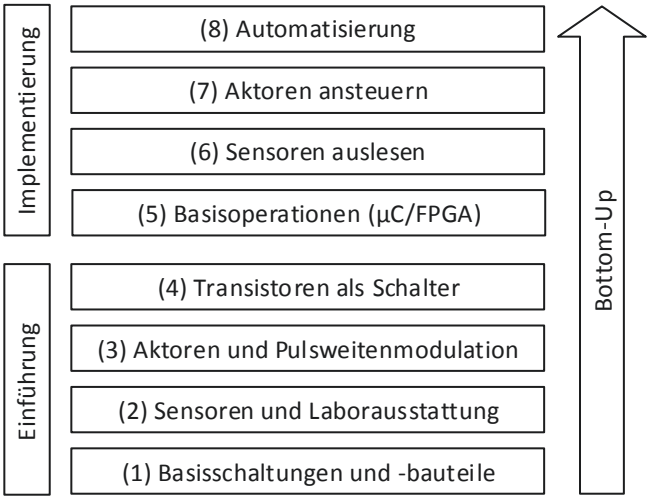


Abbildung 3.8.: Bottom-Up strukturierter Praktikumsverlauf [Jaschke et al., 2012, S. 3]

Vorbereitungsphasen (10 Stunden)

Die Vorbereitung auf die Laborversuche besteht aus vier Phasen:

Motivation Parallel zur originären Praktikumsaufgabe (Modellhausautomation) werden kleine Anwendungen für das Betriebssystem Google Android ent-

wickelt. Bei erfolgreichem Abschluss des Entwurfs- und Anwendungspraktikums kann das Modellhaus mit einem Smartphone via *Bluetooth* gesteuert werden. Dazu werden Bibliotheken bereitgestellt.

Information Die Studierenden sollen sich in der Informationsphase selbständig in die Grundlagen der Elektrotechnik, Messtechnik und Mikrocontrollerprogrammierung einarbeiten. Dazu werden ihnen die notwendige Literatur sowie Datenblätter ausgehändigt.

Vorentwicklung Angepasst an die folgende Versuchsdurchführung sowie den erarbeiteten Grundlagen sind einfache Schaltungen vorzubereiten und Programme für den Mikrocontroller zu schreiben. Dies umfasst auch Übungsaufgaben, wie die Berechnung von Spannung und Strömen in Widerstandsnetzwerken.

Selbsttest Jede Vorbereitungsphase wird einem Selbsttest unter Moodle[4] abgeschlossen, dessen Bestehen Voraussetzung für die Teilnahme in den Präsenzphasen ist. Ferner sollen die Studierenden in dieser Phase die Vorentwicklungen anderer Gruppen nach einem *Peer-Review*-Verfahren testen und beurteilen. Durch die Veröffentlichung der Lösungen aller Gruppen wird eine Steigerung der Motivation der Studierenden im Hinblick auf die Entwicklung über das Minimalziel hinausgehend erwartet.

Laborpräsenz (3 Stunden)

Die Laborphasen beginnen mit der Diskussion der Vorentwicklungen im Plenum. Sollten mehrere Studierende Fehler in den Selbsttests oder den Vorentwicklungen gehabt haben werden diese besprochen. Die Studierenden präsentieren ihre Lösungen am *Whiteboard* und fördern damit ihre sozial-kommunikativen Kompetenzen. Wissensdefizite werden aufgedeckt, woraufhin Tutoren versuchen im Rahmen kurzer Inputphasen das Vorwissen anzugleichen. Dadurch soll allen Studierenden ein erfolgreicher Abschluss der praktischen Aufgaben ermöglicht werden.

Nachbereitung (2 Stunden)

In den zweistündigen Nachbereitungsphasen sollen die Erfahrungen und der Erkenntnisgewinn dokumentiert und reflektiert werden. Messergebnisse, Signalverläufe und Schaltungsskizzen werden erstellt, um das Erlernte zu manifestieren und gleichzeitig den Nachweis der eigenen Leistung gegenüber den Tutoren zu erbringen.

3.3.5. *Virtual Workspace* als *Blended-Learning* Konzept

Verwandte Arbeiten (z. B. [Brejcha et al., 2011], [Wei-Feng et al., 2009]), verzichten auf Präsenzveranstaltungen und ermöglichen einen Fernzugang zu realer Hardware. [Reichenbach et al., 2011] entwickelten den Kurs *FPGA-Online*, in dem

[4] E-Learning Plattform an der Universität Siegen.

Studierende in einem Buchungssystem Hardware reservieren und anschließend in einer Browserumgebung programmieren und beobachten können. Dies ermöglicht eine effiziente Ressourcennutzung unter Beibehaltung realer Hardwareerfahrungen, statt Simulationen, was für die Entwicklung eingebetteter Systeme wichtig ist [Jaschke et al., 2011]. Dieses Vorgehen ist sinnvoll, solange kein direkter Hardwarezugang benötigt wird und Implementierungsaspekte im Vordergrund der Lehr-Lernprozesse stehen. Für das Sammeln von Erfahrungen mit Laborausstattung, analogen Bauteilen sowie ein eine direkte Betreuung sind Präsenzphasen unabdingbar.

Das Problem besteht darin, dass die Hardware den Studierenden in der Regel nicht ausgehändigt werden kann (Nutzung durch mehrere Gruppen), sodass diese in Vor- und Nachbereitungsphasen nicht zur Verfügung stehen. Eine Entwicklung einfacher Programme innerhalb der Vorbereitung ist nur möglich, wenn den Studierenden Entwicklungswerkzeuge bereitgestellt werden. Aufgrund der umfangreichen Umgebungen ist die Entwicklung eingebetteter Systeme zunächst schwierig [Marwedel, 2011]. Neben der Einführung neuer Technologien und Konzepte entstünde durch eine parallele Einführung der zugehörigen Entwicklungsumgebungen eine sehr steile Lernkurve (vgl. [Brück et al., 2008]). Problematischer ist jedoch, dass die Vielfalt an Betriebssystemen, Hardwarekomponenten und Konfigurationsmöglichkeiten der Entwicklungsumgebungen einen Support durch die Tutoren nahezu ausschließen. Als innovative Alternative wurde der *Virtual Workspace* entwickelt, der die traditionelle Erarbeitung von Wissen um praktische Labortätigkeiten in einem *Blended-Learning* Ansatz als Kombination aus Präsenzphasen und E-Learning verbindet (vgl. z. B. [Kerres, 2000], [Myrach und Montandon, 2007]).

Als *Virtual Workspace* wird ein USB-Stick bezeichnet, auf dem eine virtuelle Maschine mit dem Betriebssystem *Debian Linux* für die Open-Source Virtualisierungslösung *VirtualBox* der Firma *Oracle* installiert ist (vgl. [Büchner und Jaschke, 2013]). Um die zuvor beschriebene aufwändige Installation zu vermeiden wird den Studierenden mit diesem Konzept eine Entwicklungsumgebung inklusive aller Einstellungen, Verknüpfungen, Literatur etc. in einer virtuellen Maschine bereitgestellt. Die Entwicklungsumgebung, einmal am heimischen Arbeitsplatz zu Vorbereitungs- und Testphasen, aber auch an den PCs im Labor ist damit identisch und erfordert kein Kopieren von Einstellungen und Programmen. *VirtualBox* ist für die Betriebssysteme *Windows-, Macintosh-, Solaris-* und *Linux* verfügbar und somit von jedem Studierenden nutzbar. Um eine möglichst große Kompatibilität zu gewährleisten, wird ein *32-Bit Debian Linux* und ein *FAT32* formatiertes Dateisystem verwendet, dass kompatibel zu *64-Bit* Betriebssystemen als *Host* ist.

Den Studierenden stehen im *Virtual Workspace* vier Umgebungen sowie *Java* für webbasierte Simulation zur Verfügung (siehe Abbildung 3.9):

- Simulations-Umgebung,
- Mikrocontroller-Umgebung,

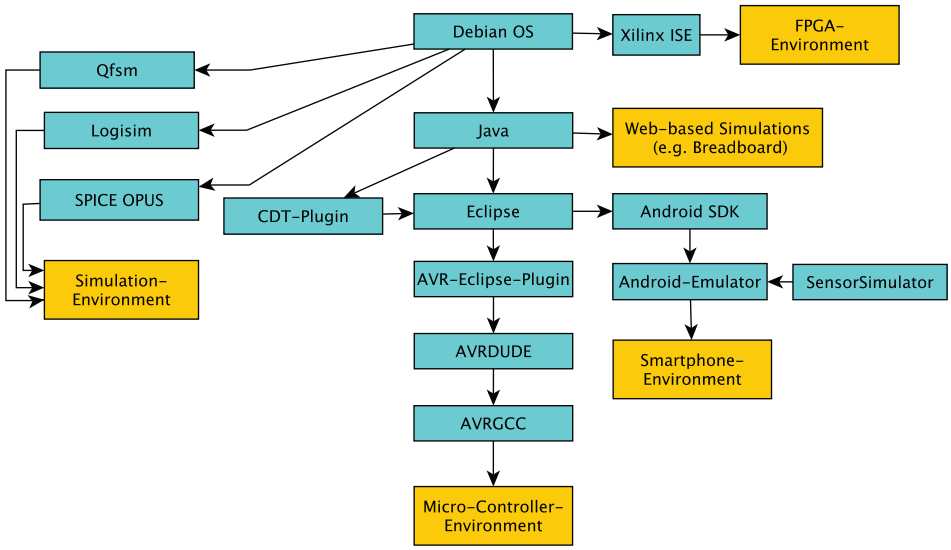


Abbildung 3.9.: Struktur des *Virtual Workspace* [Büchner und Jaschke, 2013, S. 174]

- FPGA-Umgebung,
- Smartphone-Umgebung.

Das Betriebssystem wurde für den Betrieb auf einem flashbasierten Speichermedium optimiert. Durch das *mounten* der Partition mit der Option *noatime* wird verhindert, dass bei jedem Dateizugriff ein Zeitstempel gespeichert wird. Das spart Speicherzugriffe, die bei Flashmedien begrenzt sind sowie Rechenoperationen. *Openbox*, der sich durch seine hohe Anpassungsfähigkeit bei geringem Ressourcenbedarf auszeichnet, wird als Fenstermanager eingesetzt.

Simulations-Umgebung

Simulationen sind geeignet für Experimente, welche sich real nicht durchführen lassen (zu teuer, zu gefährlich, nicht beobachtbar). Außerdem können Simulationen leichter modifiziert werden um in kürzerer Zeit viele Konfigurationen als Alternativen zu testen. Die Simulation von Schaltungen ist wichtig, da die Herstellung fehlerhafter Hardware zu Testzwecken zu teuer wäre. Dazu wird den Studierenden *SPICE OPUS* zur Simulation analoger, digitaler und gemischter elektrischer Schaltungen zur Verfügung gestellt. Das Verhalten von Logikbausteinen (UND, OR, AND etc.) ist nicht beobachtbar und würde in realen Experimenten lediglich eine Ausgabe ohne Sichtbarkeit der Zwischenschritte erlauben. Um dies zu ermöglichen, wird den Studierenden die Software *Logisim* bereitgestellt. Endliche

Automaten können im *Virtual Workspace* mit *Qfsm* modelliert und simuliert werden.

FPGA-Umgebung

Das *Xilinx ISE WebPack* steht den Studierenden als Simulations- und Entwicklungsumgebung für FPGAs zur Verfügung. Damit ist es den Studierenden möglich, ein System in VHDL zu beschreiben, zu synthetisieren und zu simulieren. Abgesehen von einem Zugriff auf die reale Hardware im Labor, ist der gesamte Entwicklungsprozess damit auch in den Vorbereitungsphasen durchführbar.

Mikrocontroller-Umgebung

Zur Programmierung der Mikrocontroller existieren alternative Entwicklungsumgebungen. Die bekanntesten sind *Atmel Studio* und *Eclipse*. Grundsätzlich kann jeder Texteditor zur C-Programmierung verwendet werden, da der Quellcode davon unabhängig compiliert wird. Das *Atmel Studio* benötigt die *Microsoft Visual Studio Shell* sowie *.NET 4.0 Laufzeitumgebung* und ist demnach nur unter Windows Betriebssystemen lauffähig. Da *Eclipse* die weiteren, zur Programmierung notwendigen Module integrieren kann und die Studierenden zudem schon *Eclipse* in Lehrveranstaltungen verwendet haben, wurde von der Installation mehrerer Editoren abgesehen. Alle Teile dieser Toolkette sind frei verfügbar. Das *CDT-Plugin* erweitert *Eclipse* zu einer Entwicklungsumgebung für *C/C++* und umfasst unter anderem einen Editor, die Projektverwaltung und Debugging-Werkzeuge. Das *AVR-Eclipse-Plugin* erweitert *CDT* um AVR-spezifische Konfigurationsmöglichkeiten, wie dem Setzen der *fuse- und lockbits*. Durch die Einbindung des *AVRGCC Compilers* und des *AVRDUDE* zur Übertragung und Manipulation des lauffähigen Codes auf den Mikrocontroller ist die Toolkette vollständig integriert.

Smartphone-Umgebung

Die Smartphone-Umgebung wird ebenfalls in *Eclipse* integriert. Für die androidspezifische Erweiterung ist das *ADT-Plugin* installiert, welches analog zum *CDT-Plugin* einen eigenen Editor (Java), sowie die Management und Debugging Werkzeuge umfasst. Verschiedene Android Versionen und APIs können somit verwaltet und parallel entwickelt werden. Mithilfe des Android-Emulators können die entwickelten Applikationen ohne vorhandenes Endgerät simuliert werden. Da dieser keine Sensoren enthält ist zusätzlich ein *SensorSimulator (Java Software)* installiert. Mit diesem kann man mehrere Sensoren simulieren und manipulieren. Alternativ können die Applikationen *gepackt* und auf einem Android Endgerät installiert werden.

Performanztest

Die Performanz des Systems wurde in einem typischen Arbeitsprozess und einer virtuellen Maschine mit *3,1 GHz CPU und 2048MB Arbeitsspeicher* getestet. Vom

Start der Maschine, über das Öffnen der *Eclipse*-Umgebung und dem *Compilieren* eines C-Programms mit 200 Zeilen Programmcode vergingen 30 Sekunden. Bei der Evaluation des Ansatzes stellte sich heraus, dass nur 12,8 % der befragten 31 Teilnehmer Performanzprobleme hatten (vgl. [Büchner und Jaschke, 2013]). Studierende der FPGA-Gruppen äußerten sich kritisch zum Einsatz des *Virtual Workspace*. Der Grund dafür liegt in der eingeschränkten Fähigkeit auf die parallele Schnittstelle des *Host Systems* zuzugreifen. Diese ist notwendig, um den synthetisierten Code auf den FPGA zu übertragen. Zu Beginn jeder Laborphase müssen die Studierenden also den erzeugten Code auf die Laborrechner überspielen und von dort auf den FPGA laden, was der ursprünglichen Intention widerspricht. Aus Sicht der Tutoren hat dieser Ansatz aber davon unabhängige Vorteile, wie die Gewährleistung einer besseren Vergleichbarkeit zwischen den Studierenden. Probleme im Arbeitsprozess sind auf ein Betriebssystem und eine Systemkonfiguration begrenzt und hängen nicht vom jeweiligen System der Studierenden ab. So können Fehler einfach reproduziert und für alle Gruppen gelöst werden.

3.4. Formative Evaluation des Entwurfs- und Anwendungspraktikums

Um in der ersten Erprobung des neuen Praktikums wertvolle Erkenntnisse zur Verbesserung der Versuche im Entwurfs- und Anwendungspraktikum und dessen Strukturierung zu erhalten wurde eine formative Evaluation durchgeführt [Jaschke et al., 2012]. In der Projektskizze von KOMINA war vorgesehen, die Kompetenzaneignung zweier Gruppen á 30 Studierender zu vergleichen. Eine Gruppe sollte das neue Entwurfs- und Anwendungspraktikum absolvieren, während eine Kontrollgruppe die Versuche des traditionellen Hardwarepraktikums durchführt. Aufgrund der geringen, hinter den Erwartungen zurückgebliebenen Teilnehmerzahlen von 30 Studierenden, wurde auf eine gegenüberstellende Studie verzichtet. Bei den beschriebenen heterogenen Gruppen wären Ergebnisse nicht quantitativ vergleichbar, da individuelle Leistungen zu stark gewichtet worden wären. Ferner existieren bislang keine empirisch überprüften Instrumente zur Kompetenzmessung in der technischen Informatik.

Es wurde eine strukturierte nicht-teilnehmende Beobachtung durchgeführt, die es ermöglicht, die Leistungen der Probanden kontinuierlich mit den kognitiven Anforderungen in den folgenden Versuchen abzugleichen und gegebenenfalls Anpassungen am Versuchsaufbau vorzunehmen (vgl. [Bortz und Döring, 2005]). Da die Räumlichkeiten des Labors keine verdeckte Beobachtung zuließen, konnte ein Wirken auf die Probanden durch die Anwesenheit der Beobachter nicht ausgeschlossen werden. Zur Minimierung dieser Effekte wurde den Teilnehmern das Vorgehen erläutert und klargestellt, dass es sich nicht um eine Leistungsbewertung handelt (vgl. [Cranach und Frentz, 1969]).

Im Labor führten sechs Gruppen die Versuche parallel durch, sodass die Beobachtung auf drei Personen aufgeteilt wurde. Um diese Erhebung vergleichbar zu machen, wurde ein gemeinsamer Beobachtungsbogen erstellt, welcher die wesentlichen Aufgaben, zu erwartende Fehler und die eingeplante Zeit enthält. Dieser wurde während eines Probedurchlaufes der neu entwickelten Versuche durch einen Studierenden erstellt, welcher im Studienverlauf (Informatik mit Anwendungsfach Medienwissenschaften) auf dem Niveau der Zielgruppe anzusiedeln war (siehe Tabelle 3.4).

Tabelle 3.4.: Beobachtungsbogen zur formativen Evaluation des Entwurfs- und Anwendungspraktikums [Jaschke et al., 2012, S. 6]

Ort, Datum	Probanden	
Beobachter	Unterschrift	
Aufgabe/Erwartungen	Beobachtungen	Zeit
1. Steckbrett verbinden		20 ( )
1.1 Spannungsversorgung		5 ( )
1.2 Widerstand		15 ( )
Erwartungen		
a) Kurzschluss		
b) Falsche Komponenten		

Im Folgenden werden die Versuche in den wichtigsten Punkten zusammenfassend beschrieben und dabei auf zentrale Beobachtungen und Teilaufgaben eingegangen, von denen wichtige Implikationen für eine Weiterentwicklung des EAP ausgingen. Im Teil Implementierung wird sich auf die vom Autor beobachteten und entwickelten Versuche zu Mikrocontrollern beschränkt. Die identifizierten Lernhürden wiederholten sich in den verschiedenen Versuchen, sodass diese vorrangig im Versuch des ersten Auftretens aufgeführt werden. Die exemplarischen Versuchsaufgaben sollen dem Leser einen Einblick in die Gestaltung der Aufgaben geben. Querverweise, Abbildungen, Hervorhebungen etc. (vgl. [Schäfer et al., 2012c]) wurden zugunsten der Lesbarkeit entfernt sowie die Formulierung angepasst. Da es sich hier um eine qualitative Auswertung der Lernhürden handelt bleiben die beobachteten Lernerfolge außer Acht.

Zur Analyse werden die Taxonomien nach [Büchner et al., 2013] und [Fuller et al., 2007] zu einer hybriden Version zusammengefasst. Die Dimension Komplexität zur Beschreibung des Systems wird entfernt, da das System im Entwurfs- und Anwendungspraktikum in allen Versuchen identisch ist und bei einer Einzelbetrachtung

keinen Mehrwert bietet. Die kognitive Stufe wird zugunsten einer im Bereich der Informatik präziseren Darstellungsform in zwei Dimensionen aufgeteilt (vgl. Abschnitt 2.3).

3.4.1. Versuch 1 – Basisschaltungen und Komponenten

Nach dem ersten Versuch sollen die Studierenden in der Lage sein einfache elektrische Widerstandsnetzwerke aufzubauen und zu analysieren (berechnen und ausmessen). Voraussetzung dafür ist, dass sie Ströme, Spannungen, Widerstände und Kapazitäten als Basisgrößen und Bauteile elektrischer Schaltungen beschreiben können. Ferner sollen sie erläutern können, inwiefern parasitäre Widerstände und Kapazitäten in elektrischen Schaltungen auftreten.

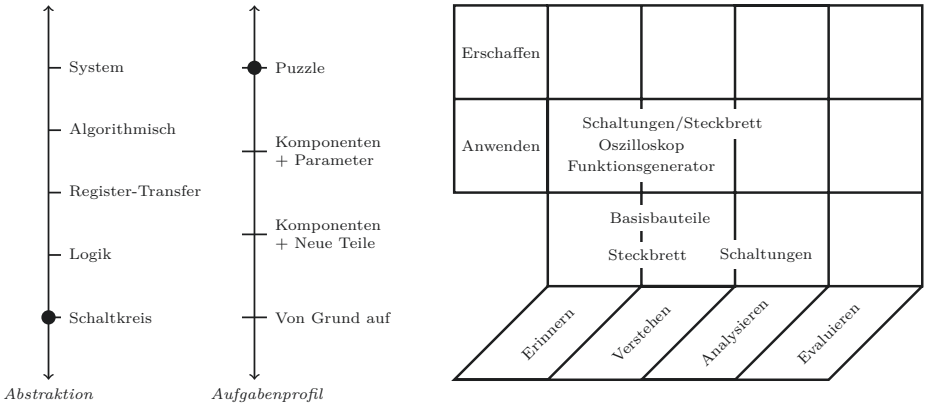


Abbildung 3.10.: Versuch 1 – Anwendung der Taxonomie

In der Vorbereitung sollen die Studierenden im Elektronikkompendium[5] Beiträge zu Stromkreisen, Widerständen, Kondensatoren, Basisschaltungen sowie Hoch- und Tiefpässen lesen. Die Bedeutung parasitärer Widerstände und Kapazitäten in der Informatik werden in [Hertwig und Brück, 2000] beschrieben. Darüber hinaus wurden den Studierenden eine Einführung und Bedienungsanleitung für die im Labor verwendeten Geräte gegeben.

Exemplarische Versuchsaufgaben der physikalisch-technischen Einführung (Versuch 1)

- Stecken Sie auf der Experimentierplatine die beiden Widerstandsnetzwerke. Prüfen Sie mit dem Digitalmultimeter, ob die verwendeten Widerstände den angegebenen

[5] Webseite über Elektronik, Computertechnik, Kommunikationstechnik und Netzwerktechnik für Schüler, Azubis und Studenten [http://www.elektronik-kompendium.de, Abruf: 12/2014].

Toleranzen genügen. Stellen Sie an der stabilisierten Spannungsquelle eine Spannung von 5V ein und begrenzen Sie den Strom auf 50 mA. Messen Sie mit den Digitalmultimetern alle Teilströme in allen Zweigen und Spannungen über allen Widerständen.

- Bauen Sie eine Tiefpassschaltung, bestehend aus einem oder mehreren Widerständen mit 2400 Ohm und einer Kapazität von 1200 pf, auf. Überprüfen Sie das Schaltverhalten am Ausgang der Tiefpassschaltung, indem Sie am Funktionsgenerator eine Rechteckspannung mit einer Frequenz von f = 50 kHz einstellen. Überprüfen Sie die Einstellungen mithilfe des Oszilloskops.

An die Studierenden werden Anforderungen auf unterschiedlichen kognitiven Stufen gestellt, welche in der Regel über das Verstehen von Sachverhalten hinaus, bis hin zur Analyse einfacher Schaltungen gehen (siehe Abbildung 3.10). Das Aufbauen der vorgegebenen Schaltungen erfordert, dass die Studierenden die Basisbauteile erkennen sowie den Verdrahtungsplan des Steckbretts verstehen. Zur Anwendung des Oszilloskops müssen die Studierenden verstehen, wie elektrische Spannung gemessen wird und sich erinnern, welche Einstellungen am Oszilloskop vorzunehmen sind.

Beobachtung

Die Studierenden verfolgten verschiedene Strategien zur Bestimmung des Widerstandswertes und der Toleranz. Einige Studierende benutzten eine im Rahmen der Vorbereitung zum Versuch entwickelten Android-Applikation. Andere verwendeten die in der Literatur angegebene Farbkodierungstabelle. Weitere nutzten, wie in der Aufgabenstellung angegeben, das Digitalmultimeter und bestimmten so den Widerstandswert. Wie auch im weiteren Versuchsverlauf konnte beobachtet werden, dass bei der Benutzung der Laborausstattung Fehlbedienungen auftraten - insbesondere bei der Verwendung des Oszilloskops und Multimeters, was auf verschiedene Ursachen zurückzuführen war, z. B. die Studierenden:

- haben die Bedienungsanleitungen nicht gelesen,
- konnten sich nicht an die relevanten Abschnitte erinnern,
- können sich nicht erinnern, welche Einstellungen in welchem Kontext vorzunehmen sind.

Ein Verstehen der elektrotechnischen Größen und Bauteile ist Vorbedingung zur Berechnung und Analyse einfacher Schaltungen. Lernhürden können demnach auf verschiedenen kognitiven Stufen auftreten (z. B. Verstehen von Bauteilen, Analyse von Schaltungen). In verschiedenen Situationen verfolgten die Studierenden eine Versuch-Irrtum-Strategie, indem solange Einstellungen ausprobiert wurden, bis das zu erwartende Verhalten zu beobachten war. Es stellt sich die Frage, inwiefern

sich Studierende besser auf die Anwendung eines unbekannten Instrumentariums vorbereiten werden können, ohne auf die Hilfe der Tutoren respektive Kommilitonen angewiesen zu sein. Weitere typische Fehler waren das Anschließen der Messinstrumente an falschen Punkten in der Schaltung (z. B. Strom parallel zum Widerstand messen) oder das unkorrekte Bestücken des Steckbretts.

In der Vorbereitung zur Versuchsreihe ist das Lesen der angegebenen Literatur allein offensichtlich nicht zielführend, auch wenn Lernen am Text immer eine bedeutsame Rolle spielt. Die Qualität des Textlernens hängt unter anderem vom Vorwissen der Lernenden ab. Je mehr Vorwissen vorhanden ist, desto eher können Studierende diesen aktiv verarbeiten und umso mehr wird aus Texten gelernt – Schlussfolgerungen ziehen, Probleme lösen [Wild und Möller, 2009].

Betrachtet man den Umfang des Skriptes zum Praktikum, ist es nachvollziehbar, dass mehrere hundert Seiten nicht aktiv verarbeitet werden können. Den Studierenden fehlt es an einer Orientierung im Fachgebiet, da weder die Arbeitsweise in Laborpraktika bekannt ist, noch sind die Studierenden mit der Terminologie vertraut. Im Labordesign erfolgte offensichtlich eine Fehlvorstellung von den Vorkenntnissen der Studierenden. Es wurde erwartet, dass auch diejenigen ohne Nebenfach Elektrotechnik elektrische Größen und einfache Stromkreise aus dem Physikunterricht in der Sekundarstufe II besitzen. Dies ist jedoch nicht der Fall, sodass es nicht möglich war im gegebenen Zeitrahmen das Wissensdefizit auszugleichen, insbesondere durch ein ausschließlich autodidaktisches Vorgehen.

3.4.2. Versuch 2 – Sensoren und Laborausstattung

Nach dem zweiten Versuch sollen die Studierenden in der Lage sein einfache elektrische Aktor- und Sensorschaltungen aufzubauen, zu verknüpfen und zu testen, sowie Passagen aus Datenblättern von Sensoren herauszustellen.

Die Studierenden erweitern in der Vorbereitungsphase ihre Kenntnisse zu elektrischen Bauteilen, um die Diode und lesen die Datenblätter zu den im Modellhaus eingebauten Aktoren und Sensoren. Ein einfacher Spannungsteiler aus Widerstand und den resistiven Helligkeits- und Temperatursensoren wird in der Vorbereitungsphase selbständig skizziert.

Exemplarische Versuchsaufgaben (Versuch 2)

- Bestimmen Sie anhand ihrer Schaltung und mithilfe des Datenblattes die Temperatur und Luftfeuchtigkeit im Laborraum. Halten Sie für das Protokoll die Ergebnisse fest. Notieren Sie auch, wie Sie diese Ergebnisse bestimmt haben.
- Zur Bestimmung der Kennlinie einer LED bauen Sie die Schaltung auf dem Steckbrett auf. Der Funktionsgenerator liefert eine Sinusspannung zur Ansteuerung der

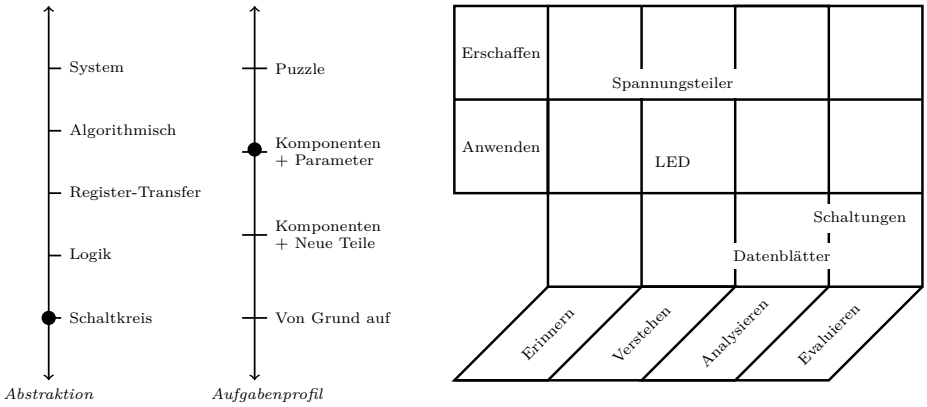


Abbildung 3.11.: Versuch 2 – Anwendung der Taxonomie

> LED. Als Funktionsgenerator dient der HAMEG HM8030. Stellen Sie die Amplitude auf den kleinsten Wert (min) und schalten Sie den Offset ab. Schließen Sie das Oszilloskop an die jeweiligen Kanäle an und wählen Sie den *X-Y-Modus*.

Einfache Schaltungen zu testen erfordert eine hohe kognitive Leistung, da die Studierenden zum einen eine Vorstellung einer „richtigen“ Schaltung besitzen müssen und zum anderen Bewertungskriterien und Vorgehensweisen zur Evaluation von Schaltungen besitzen müssen. Ein Spannungsteiler muss angepasst an die verwendeten Sensoren ausgewählt werden (Bestimmung der Parameter).

Beobachtung

Auch in dieser Durchführungsphase benötigten die Studierenden viel Unterstützung bei der Anwendung des Oszilloskops. Die richtigen Messpunkte für die Spannungsmessung wurden nur von wenigen Studierenden auf Anhieb und ohne Hilfe gewählt. Die meisten Studierenden benötigten mehr Zeit für die Bewältigung der Aufgaben als zuvor angenommen. Insbesondere die LED Kennlinie konnte von keiner Gruppe ohne Hilfe ermittelt werden, obwohl die Vorgehensweise zur Ermittlung skizziert war. Die Lernhürden aus der ersten Versuchsreihe wirkten sich also auf den zweiten Versuch aus. Die LED als nichtlineares Bauteil in Kombination mit der Bedienung des Oszilloskops überforderte die Studierenden im zweiten Versuch. Zu hinterfragen ist, inwiefern Studierende Kennlinien von Bauteilen im *X-Y-Modus* des Oszilloskops ermitteln können müssen und ob dieses zur Förderung der benannten Kompetenzen notwendig ist. Bei der Benutzung der Laborausstattung, insbesondere des Oszilloskops konnten keine Unterschiede zwischen den verschiedenen Gruppen (Studierende mit und ohne Nebenfach Elektrotechnik) festgestellt

werden, was sich damit begründen lässt, dass auch Studierende mit Nebenfach Elektrotechnik in der Regel keine praktischen Laborerfahrungen besitzen.

3.4.3. Versuch 3 – Aktoren und Pulsweitenmodulation

Die Studierenden sollen nach dem dritten Versuch in der Lage sein Analog/Digital-Wandlungsverfahren handschriftlich durchführen und diese unterscheiden zu können. Ferner sollen die Studierenden das Konzept der Pulsweitenmodulation darstellen können und das Wirken der Strom-/Spannungsverläufe auf einfache Aktoren (LED) erläutern. Zudem sollen die Studierenden einfache Aktor- und Sensorschaltungen testen können.

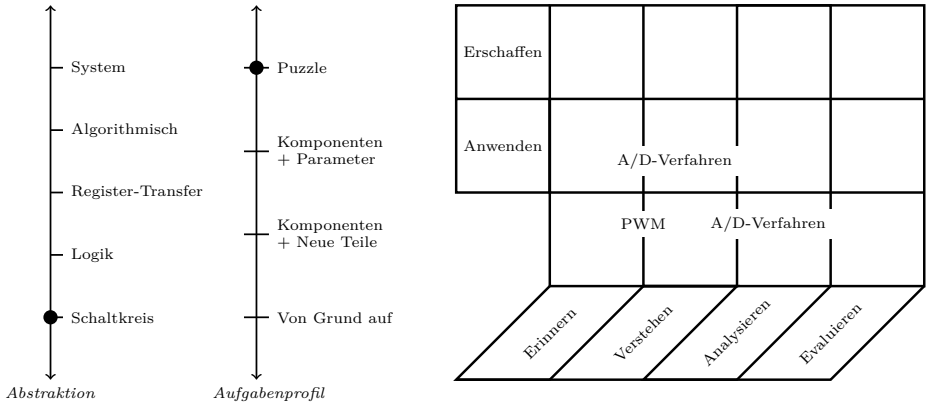


Abbildung 3.12.: Versuch 3 – Anwendung der Taxonomie

Die im Modellhaus verwendeten Aktoren und Sensoren werden in der Vorbereitungsphase vorgestellt und zu einfachen Schaltungen verbunden. Rechnerisch werden die zu erwartenden Ausgaben von 8-Bit Analog/Digital-Wandlern bei typischen Größen des umgebenden Prozesses ermittelt (Temperatur).

Exemplarische Versuchsaufgabe (Versuch 3)

Bauen Sie die Sensorschaltung zur Bestimmung der Helligkeit aus Versuch 2 zusätzlich auf dem Steckbrett auf. Benutzen Sie diese Schaltung, um die Helligkeit der LED zu bestimmen. Achten Sie darauf, dass der LED-Spot den Sensor beleuchtet. Nutzen Sie wieder den Arbiträrgenerator um eine PWM zu erzeugen. Stellen Sie erneut die in der Vorbereitung ermittelten Werte ein und überprüfen Sie ob die LED laut Helligkeitssensor(VT93N1) zu 50 % leuchtet. Überprüfen Sie auch die PWM-Stufen.

Beobachtung

Beobachtet wurde, dass die Studierenden im dritten Versuch weniger Hilfe bei der Verwendung des Oszilloskops benötigten, allerdings die Messpunkte an den aufzubauenden Schaltungen erst nach mehreren Versuchen korrekt gewählt wurden. Nach dem Experimentieren mit verschiedenen Parametern erkannten die Studierenden, wie diese miteinander in Verbindung stehen. Das Auge als Beispiel für die Trägheit von Sensoren und der notwendigen Frequenz > 25 Hz erkannten die Studierenden größtenteils. Warum eine Halbierung des *Duty-Cycles* nicht zu einer Halbierung der Helligkeit führte, musste von Tutoren erläutert werden.

3.4.4. Versuch 4 – Transistoren und Signalerzeugung

Nach der vierten Versuchsreihe sollen die Studierenden in der Lage sein die Funktionsweise von *MOSFETs* anhand der Übertragung von Eingangs- und Ausgangskennlinien zu untersuchen. Sie sollen die Randbedingungen eingebetteter Systeme verstehen.

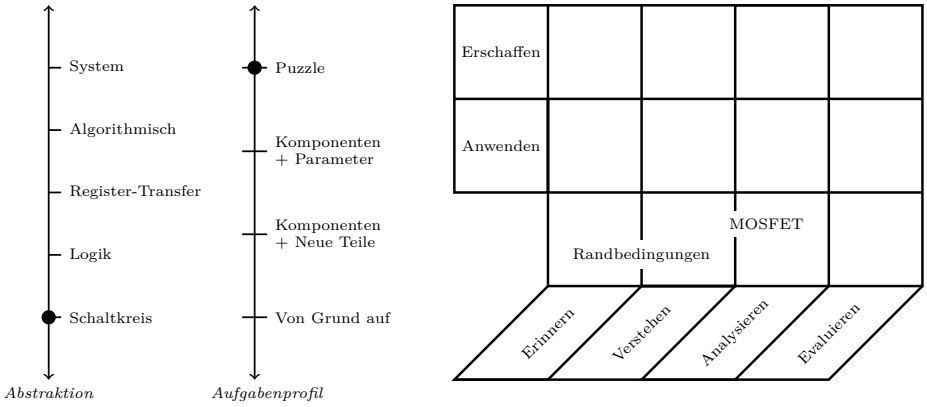


Abbildung 3.13.: Versuch 4 – Anwendung der Taxonomie

Die Studierenden erarbeiten sich die Grundlagen zu Transistoren und die Anwendung des Arbiträrgenerators erneut im Selbststudium. Dazu erstellen sie auch eine Liste von Einstellungen, welche dann im Versuch überprüft werden sollen.

Exemplarische Versuchsaufgabe (Versuch 4)

Bauen Sie als Nächstes auf der Experimentierplatine die skizzierte Messanordnung zur Aufnahme der FET-Eingangskennlinien auf. Bei der Generatorspannung *Gen1* handelt

es sich um die bereits verwendete Sinusspannung mit der Frequenz von 1,44KHz, einer Amplitude von 9 Volt und einer Offsetspannung von 4,5 Volt.

Beobachtung

Beobachtet werden konnte, dass Studierende dann erfolgreich die Aufgabe bewältigen konnten, wenn die in der Vorbereitung ermittelten Einstellungen und Schaltungen korrekt waren. Mussten nach der Einführung Änderungen vorgenommen werden, so wurden Fehler beim Aufbau der Schaltungen und Einstellungen durch die Versuch-Irrtum-Strategie behoben. Auch nach der vierten Versuchsreihe hat die Studierende die Bedienung des Oszilloskops und Arbiträrgenerators überfordert. Zumeist besteht eine Vorstellung von dem zu erwarteten Signalverläufen, sodass mittels Versuch-Irrtum-Strategie solange getestet wird, bis das Signal dem erwarteten Verlauf entspricht. Daher wurden oftmals die Einstellungen anderer Gruppen übernommen.

3.4.5. Versuch 5 – Basisoperationen (µC/FPGA)

Die Studierenden sollen in der Lage sein den Umgang mit der Programmiersprache C und der Hardware zu erläutern und – unter Berücksichtigung der Auswirkungen von Hardwarekomponenten auf die Software – Programme zu entwickeln.

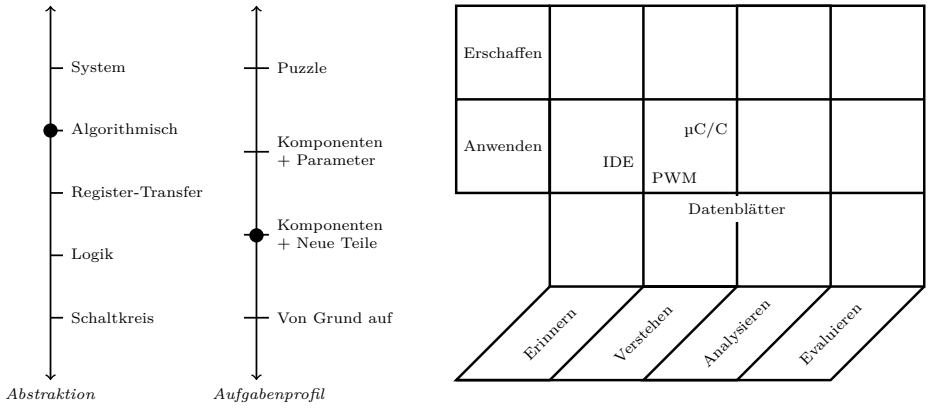


Abbildung 3.14.: Versuch 5 – Anwendung der Taxonomie

In der Vorbereitungsphase wird das *AVR ATxmega128A1 Xplained Board*, der Programmieradapter *AVR Dragon* und die Toolkette eingeführt. Die Studierenden erarbeiten sich selbständig in die Spezifika des Controllers ein – DMA Kanäle, Kryptographie, Analog/Digital-Wandler, Digital/Analog-Wandler, Eventsystem.

Sie testen die Mikrocontrollerumgebung des *Virtual Workspace* erstmals mit einem Demoprogramm zur Ansteuerung einer LED.

Exemplarische Versuchsaufgabe der Implementierungsphase

Stellen Sie sich vor, durch schaltungsbedingte Constraints (Einschränkungen, Rahmenbedingungen) darf der Strombedarf zum Betrieb des im Folgenden zu programmierenden Systems $20mA$ zu keinem Zeitpunkt überschreiten. Schalten Sie einen 10 Ohm Messwiderstand in Reihe zum Mikrocontroller (siehe Versorgungskabel für den Mikrocontroller). Lassen Sie sich die am Widerstand abfallende Spannung am Oszilloskop anzeigen und ermitteln Sie damit den Strom in mA. Das System hat nur die Aufgabe, 4 LEDs mit PWM (2kHz) zu je 50% Duty-Cycle zu dimmen. Beachten Sie dabei, dass diese im Mikrocontroller auf verschiedene Weise realisiert werden kann. Wählen Sie eine Variante nach Belieben.

Beobachtung

Nur wenige Studierende waren in der Lage die Register zur Erzeugung der Pulsweitenmodulation auf Anhieb richtig zu belegen. Die Verwendung von *Pull-Up Widerständen* und die Bedeutung innerhalb der Programmierung wurde von nahezu keiner der Gruppen verstanden.

Die Studierenden konnten sich nicht die Funktionsweise des Mikrocontrollers mithilfe des Datenblatts erschließen. Darüber hinaus wurden Bitoperationen erraten, bis das Verhalten des Mikrocontrollers den Erwartungen entsprach. Es konnte kein strukturiertes Vorgehen bei der Entwicklung der einfachen Programme erkannt werden, was mit der fehlenden Orientierung im Fach zusammenhängt. Die Terminologie in den Datenblättern – in englischer Sprache – war den Studierenden unbekannt, sodass sich die Funktionsweise des Mikrocontrollers nicht erschloss.

3.4.6. Versuch 6 – Analog/Digital-Wandler – Sensoren

Nach der sechsten Versuchsreihe sollen die Studierenden in der Lage sein eine strukturierte Vorgehensweise bei der Fehleranalyse von eigenem und fremdem Programmcode anzuwenden. Sie sollen C-Code interpretieren können und hinsichtlich der Aufgabenstellungen erweitern. Sie sollen in der Lage sein das Verhalten verschiedener Sensoren (linear/nichtlinear) zu erkunden und auf die Programmierung zu übertragen.

Um das angeschlossene Modellhaus anzusteuern sollen die Studierenden vorgegebene Bibliotheken analysieren und Grundlagen, wie *LSB, MSB* und Registerzuweisungen recherchieren.

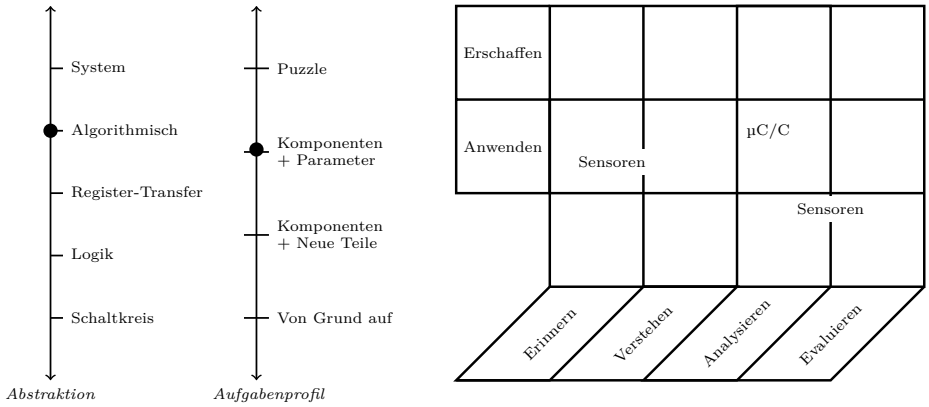


Abbildung 3.15.: Versuch 6 – Anwendung der Taxonomie

Exemplarische Versuchsaufgaben (Versuch 6)

- Die zurückgegebenen Sensorwerte sind 12-Bit lang. Um diese zwölf Bit anzuzeigen, benutzen Sie die auf dem Board vorhandenen LEDs. Da deren Anzahl jedoch nicht ausreicht, empfiehlt es sich, lediglich die acht **hochwertigsten** Bits mit den LEDs anzeigen zu lassen. Sie verlieren dadurch natürlich Genauigkeit, haben aber auf der anderen Seite noch am ehesten die Möglichkeit die Korrektheit Ihrer Werte abzuschätzen.
- Die wichtigsten Einstellungen der ADCs können Sie aus diesem Programm übernehmen. Beachten Sie jedoch, dass diesmal nicht der Onboard-Sensor, sondern ein externer Sensor betrieben werden soll. Benutzen Sie dafür unter anderem folgende Parameter:
  - Temperatursensor 0: Channel 0, MUXPOS_PIN1
  - Temperatursensor 1: Channel 1, MUXPOS_PIN3
  - Helligkeitssensor: Channel 3, MUXPOS_PIN7
  - Spannung: 1 V intern
  - ADC-Einstellung: Signed, Single-Ended

Beobachtung

Bitoperationen wurden größtenteils nicht auf Anhieb korrekt angewandt. Ferner konnten die Studierenden keine sinnvolle Registerbelegung aus den Datenblättern ableiten, wenngleich alle notwendigen Parameter (siehe Aufgabenstellung) benannt

wurden. Die Studierenden konnten die analogen Werte und die zugehörigen physikalischen Größen (z. B. Temperatur) nicht aus den Datenblättern ableiten.

3.4.7. Versuch 7 – Pulsweitenmodulation – Aktoren

Nach dieser Versuchsreihe sollen die Studierenden in der Lage sein Auswirkungen verschiedener Aktoren zu erkunden und auf die Programmierung zu übertragen.

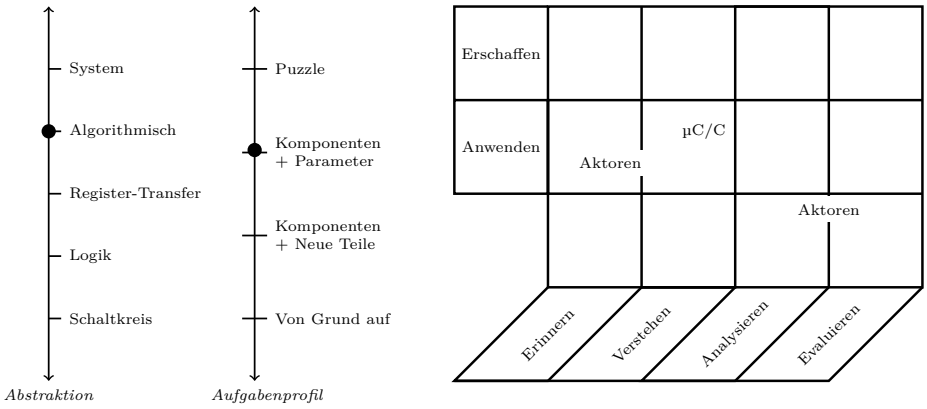


Abbildung 3.16.: Versuch 7 – Anwendung der Taxonomie

Exemplarische Versuchsaufgaben (Versuch 7)

- Schreiben Sie ein Programm, das einen Aktor (Innenlüfter oder Peltier-Element) im Haus über einen Schalter auf dem Mikrocontroller/FPGA an- und ausschaltet. Wenn der Aktor an ist, soll seine Leistung über eine PWM zu 50 % gedrosselt werden.
- Kontrollieren Sie mit ihrer Lösung von Versuchsreihe 6, ob sich die Sensorwerte im Haus über die Aktoren ändern lassen. Sie müssen noch keine Regelung implementieren.

Beobachtung

Beobachtet werden konnte, dass die Konfiguration der Register nun zumeist ohne Hilfe, wenn auch nach einigen Fehlversuchen, erfolgreich war. Der Programmierstil wurde zunehmend strukturierter, sodass Bitmasken zur Registermanipulation eingesetzt wurden. Dennoch gelingt es nicht allen Studierenden, die Angaben in Datenblättern zu interpretieren oder die notwendige Passage für die Aufgabenstellung zu identifizieren.

3.4.8. Versuch 8 – Automatisierung

Nach dem letzten Versuch sollen die Studieren in der Lage sein eigenverantwortlich alleine und in Gruppen Aufgaben zu lösen und Ergebnisse zu präsentieren um somit im beruflichen Umfeld mit Ingenieuren und Elektrotechnikern als Informatikanwender zu kommunizieren. Sie kennen die Funktionsweise von unterschiedlichen Sensoren und Aktoren von aktuellen eingebetteten Systemen, verstehen den Zusammenhang zwischen Hardwarekonzepten und den Softwarekomponenten.

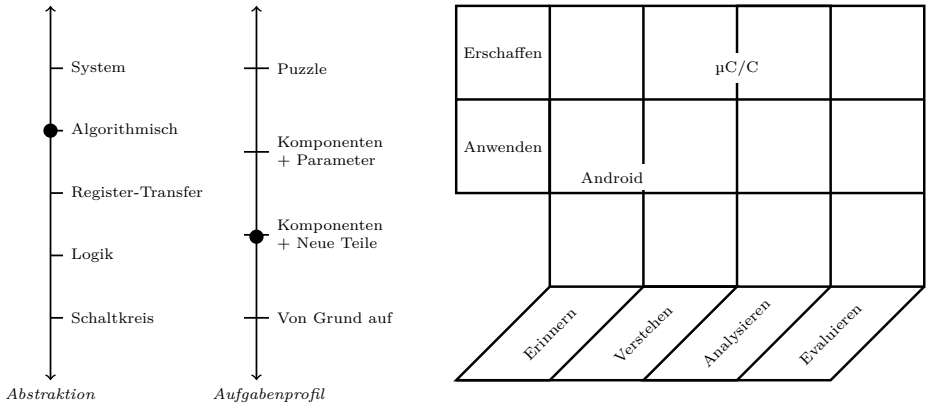


Abbildung 3.17.: Versuch 8 – Anwendung der Taxonomie

In der letzten Versuchsreihe sollen die Studierenden die vorherigen Teilprogramme verknüpfen, um die technischen Prozesse des Modellhauses zu regeln.

Exemplarische Versuchsaufgaben (Versuch 8)

- Setzen Sie die in der Einleitung erklärte Aufgabenstellung um. Fangen sie dazu erst mit der Funktionalität der Taster/LEDs 1-4 an. Wenn dies funktioniert, versuchen Sie die automatische Regelung der Temperatur umzusetzen.
- Damit Sie besser ablesen können, wie stark das Peltier-Element momentan heizt oder kühlt, können Sie die LEDs 1 und 2 mit der selben PWM ansteuern wie das Peltier-Element. Alternativ können Sie ein Blinkmuster implementieren.

Beobachtung

Es konnte beobachtet werden, dass die Studierenden in der Lage waren, Sensoren auszulesen und Aktoren anzusteuern. Eine Regelung haben die meisten Studenten bislang nicht implementiert, sodass auch keine Algorithmen dafür bekannt waren und von den Tutoren eingeführt werden mussten.

3.4.9. Analyse

Die Versuche der physikalisch-technischen Einführung beschränken sich auf Aufgaben der untersten Abstraktionsebene mit der Kombination von Bauteilen auf der Ebene *Puzzle* bis zur Parametrisierung von Komponenten. Dabei lassen sich Diskrepanzen zwischen den intendierten Kompetenzen, den Aufgabenstellungen und den Arbeiten in den Vorbereitungsphasen erkennen. Bereits im ersten Versuch werden beispielsweise Schaltungen auf einem Steckbrett aufgebaut, mit Signalen des Funktionsgenerators gespeist und mittels Oszilloskop gemessen (Stufe *Verstehen/ Anwenden*). In den folgenden Versuchen werden weitere Bauteile (LED, Sensoren) und Konzepte (Analog/Digital-Wandlung, Pulsweitenmodulation) eingeführt. Ein Problem besteht darin, dass die Studierenden zur Bewältigung der ersten Versuchsaufgabe Konzepte anwenden sollen, welche bislang nicht Teil des Studiums waren und nur theoretisch im Selbststudium erarbeitet wurden. Analog zu den folgenden Versuchen muss auch hier eine Schrittweise Einführung eines Fachkonzeptes bis zur Ebene *Verstehen/Analysieren* erfolgen und erst im anschließenden Versuch zur Anwendung und Kombination verschiedener Konzepte übergegangen werden. Die Hürde beim Übergang zum zweiten Versuch ist sehr groß, da zuvor unbekannte Schaltungen (Spannungsteiler) nun bereits dimensioniert werden sollen, ohne mittels Einführung von Sensoren einen sinnstiftenden Zusammenhang zu schaffen. Die als wichtig hervorgehobenen Randbedingungen auf niederen Abstraktionsebenen besitzen im aktuellen Labordesign keine besondere Stellung und beschränken sich überwiegend auf die Ebene *Erinnern/Verstehen*. Ferner ist zu erkennen, dass die eingeführten Bauteile *MOSFET*, der Aufbau einer Hochpass- und einer Tiefpassschaltung sowie die Kennlinie der LED für die Versuche fünf bis acht nicht relevant sind. Abgesehen vom zweiten Versuch waren die Aufgaben auf das Profil *Puzzle* beschränkt. Dies stellt einen geeigneten Ansatz zur Förderung der Entwicklung eingebetteter Systeme dar, im von der Implementierung im Sinne eines vorgegebenen Designs, für das überwiegend Kompetenzen auf den Ebenen Verstehen bis *Verstehen/Anwenden* erforderlich sind, zu einem selbständigen Design in weiterführenden Veranstaltungen überzugehen.

In der Implementierungsphase wird die Steuerung und Regelung der technischen Prozesse des Modellhauses realisiert. Zunächst werden die dazu notwendigen Basisoperationen und Register sowie die Entwicklungsumgebung eingeführt. Problematisch ist, dass die Studierenden lediglich über Verständnis für die Funktionalität von Sensoren und Aktoren, nicht jedoch für die Schnittstelle zum eingebetteten System verfügen. Intendiert war das Verstehen der Technologien von Mikrocontrollern, was offensichtlich aufgrund der Funktionsvielfalt und Systemkomplexität nicht innerhalb eines Versuches gelingen kann. Hier empfiehlt sich eine schrittweise Einführung von der externen Hardware zur internen Repräsentation analoger Werte. Insbesondere die Analyse fremder Programme überfordert die Studierenden. Studierende mit Erfahrung in der Programmierung von Mikrocontrollern besitzen einen Programmierstil, der eine effektive Wiederverwendung und Wartbarkeit

ermöglicht. Für Novizen ist dieser aufgrund der noch unbekannten Konstrukte, wie Bitoperationen, Bitmasken, Registern etc. aber nur schwer verständlich. Das führt dazu, dass Codeblöcke vollständig übernommen werden ohne die zugrundeliegenden Überlegungen und Funktionsprinzipien zu verstehen. Für die praktischen Laborversuche diente die Entwicklung kleinerer Applikationen für Android lediglich der Motivation. Eine weiterführende didaktische Analyse der Lernhürden im Entwurfs- und Anwendungspraktikum wird im folgenden Kapitel erörtert und ergänzt diese summarische Beschreibung der Voranalyse.

3.4.10. Zusammenfassung und Fazit

Die Ergebnisse aus KOMINA tragen zu einer Verbesserung von Lehr-Lernprozessen bei, indem von kulturspezifisch geprägten zu theoretisch fundierten Konzepten einer Kompetenzorientierung bei der Entwicklung eingebetteter Systeme übergegangen wird. Dazu wurde zunächst die Struktur von Kompetenzen und deren Facetten erforscht. Ein normativ hergeleitetes Kompetenzstrukturmodell wurde durch eine schriftliche Befragung von Experten der technischen Informatik empirisch verfeinert, um die zunächst intersubjektiv hergeleitete Struktur von Kompetenzen hinsichtlich Vollständigkeit und Wichtigkeit zu validieren.

Für die exemplarische Umsetzung des Kompetenzstrukturmodells wurden Versuche für ein Entwurfs- und Anwendungspraktikum, die eine hinreichende Nähe zu elektrisch-physikalischen Problemstellungen bieten, entwickelt. Lerngegenstand ist ein mit Sensoren und Aktoren ausgestattetes Modellhaus (lernförderliche Hardware). Eine virtuelle Arbeitsumgebung erlaubt die Entwicklung von Programmen in einem *Blended-Learning* Ansatz.

Um wertvolle Erkenntnisse zur Verbesserung der Versuche im Entwurfs- und Anwendungspraktikum und dessen Strukturierung zu erhalten, wurde eine formative Evaluation durchgeführt und vier Lernhürden (LH_i) identifiziert.

$\mathbf{LH_1}$ – Dokumentationen und Datenblätter zur Analyse von Funktionsprinzipien

Wissen über die Terminologie und über das Fachgebiet sind Voraussetzung, um in Lehr-Lernprozessen sinnvoll zu agieren. Ohne dies sind weder die Aufgabe selbst noch die Datenblätter der zur Entwicklung notwendigen Hard- und Softwarekomponenten zu verstehen.

$\mathbf{LH_2}$ – Vorgehensweise zur Strukturierung des Entwicklungsprozesses

Ohne Kenntnisse zum Entwicklungsprozess und Vorgehensweisen zu dessen Strukturierung ist es nicht möglich die Aufgaben mit akzeptablem Zeitaufwand zu bewältigen, da zwangsläufig eine Versuch-Irrtum-Strategie angewandt wird.

LH_3 – Laborausstattung als Mittel zur Erzeugung und Messung elektrischer Größen

Dies beinhaltet die Grundlagen der Elektrotechnik, da nicht eindeutig zu differenzieren ist, ob diese Lernhürde aufgrund der mangelnden Kenntnisse der Elektrotechnik oder dem fehlenden Verständnis für den Umgang mit der Laborausstattung auftritt.

LH_4 – Register und Bitoperationen als Konzepte hardwarenaher Programmierung

Bei der hardwarenahen Programmierung werden zwar analog zur Anwendungssoftware die Sprachkonstrukte (Schleifen, Bedingungen etc.) in C verwendet, dennoch sind Bitoperationen und Registerzuweisungen zumeist unbekannt und müssen strukturiert eingeführt werden.

4. Informatikdidaktische Verfeinerung des Kompetenzstrukturmodells

> "Der Versuch, sich in einem komplexen Wissensstoff sequentiell als fortlaufende Liste der wichtigsten Begriffe einzuprägen, ist ziemlich sicher zum Scheitern verurteilt" [Edelmann und Wittmann, 2012, S. 137].

Die Forschungsarbeiten von [Brinda, 2001], [Brinda, 2004] und [Freischlad, 2010] führten didaktische Systeme als Strukturierungsinstrument von Lehr-Lernprozessen ein. Während [Brinda, 2004] die objektorientierte Modellierung fokussierte, diskutierte und transferierte [Freischlad, 2010] das Konzept der didaktischen Systeme auf das Themengebiet Internetworking. Die Entwicklung eines didaktischen Systems für das Design eingebetteter Systeme ist anzustreben, da es eine theoretische Fundierung und konkrete Ausgestaltung von Lehr-Lernprozessen zu einer informatikdidaktischen Teildisziplin ermöglicht.

Nach dem derzeitigen Forschungsstand ist die Entwicklung eines didaktischen Systems für das Design eingebetteter Systeme jedoch nicht realisierbar, da zunächst eine präzisere Vorstellung von den zu erlangenden Kompetenzen künftiger Entwickler in weiteren Grundlagenforschungen zum Kompetenzstrukturmodell gewonnen werden muss. Dazu ist es notwendig die Kompetenzsubdimensionen des empirisch verfeinerten Kompetenzstrukturmodells weiter auszudifferenzieren sowie identifizierte Lernhürden zu analysieren. Die Komponenten didaktischer Systeme dienen einerseits als Inspiration für die empirische Verfeinerung des Kompetenzstrukturmodells und anderseits als entferntes Forschungsziel, an das die aktuelle Kompetenzforschung zu eingebetteten Systemen ausgerichtet werden kann. Diese Komponenten sind somit zunächst an das Forschungsgebiet Hochschuldidaktik der technischen Informatik anzupassen beziehungsweise zu erweitern.

4.1. Weiterentwicklung didaktischer Systeme

Im Folgenden werden die drei Komponenten didaktischer Systeme – Wissensstrukturen, Aufgabenklassen und Lernhilfen – anhand von Beispielen diskutiert und adaptiert. Dabei wird immer berücksichtigt, dass das Forschungsgebiet des Au-

tors, verglichen mit der Erforschung von Kompetenzen zur objektorientierten Modellierung, noch wenig theoretisch fundiert ist und die Lehre oftmals aus einer Abbilddidaktik des Faches resultiert. Der Fokus dieser Arbeit liegt auf der Ableitung und Visualisierung von *kognitiven Strukturen* (siehe Abschnitt 4.1.1) als Voraussetzung für die Erforschung von Aufgabenklassen und lernförderlicher Soft- beziehungsweise Hardware. Bei der Erforschung von Komponenten eines didaktischen Systems in der technischen Informatik wird durch die Forschungsarbeit des Autors zunächst eine durchgängige Integration von Ergebnissen der Kompetenzforschung – kognitive Strukturen und Kompetenzstrukturmodelle – angestrebt. Erfahrungen und Ergebnisse im Forschungsprojekt KOMINA, das empirisch verfeinerte Kompetenzstrukturmodell, leisten insoweit ebenso einen Beitrag wie die Vorarbeiten des Autors, welche die Entwicklung der Taxonomie zur Vergleichbarkeit fachdidaktischer Publikationen erweiterten. Ferner werden die intersubjektiv vergleichbaren, im entwickelten Entwurfs- und Anwendungspraktikum erzielten Beobachtungsergebnisse, an denen der Autor maßgeblich beteiligt war, in die Forschungsarbeit einbezogen. Das empirisch verfeinerte Kompetenzstrukturmodell ist in vier Kompetenzdimensionen gegliedert, welche nach der weiteren Ausdifferenzierung als Grundlage zur Darstellung von kognitiven Strukturen in einem didaktischen System für das Design eingebetteter Systeme dienen können. Bis zu dessen Entwicklung existierte keine theoretisch fundierte Vorstellung über die zu erlangenden Kompetenzen der Zielgruppe (vgl. Kapitel 2). Lernförderliche Soft- und Hardware wurde im Rahmen des Entwurfs- und Anwendungspraktikums betrachtet (siehe Kapitel 3) und beinhaltet eine Form des *Blended-Learning* in Laborpraktika, wodurch personelle Ressourcen eingespart werden. Den Studierenden wird ein reibungsloser, auf die Förderung von Entwicklungskompetenzen fokussierender Einstieg geboten. Die identifizierten Lernhürden implizierten den weiteren Bedarf an Medienunterstützung in hardwareorientierten Praktika (vgl. Kapitel 3). Ferner wurde festgestellt, dass eine mögliche Variante der Entwicklung von Kompetenzen für das Design eingebetteter Systeme in der Gestaltung von Lehr-Lernprozessen und Aufgaben nach einem systemorientierten Ansatz besteht. Danach werden die wesentlichen Komponenten eingebetteter Systeme mit Bezug zur physikalischen Realität schrittweise eingeführt und damit die zielgruppenspezifischen Kompetenzen zur Entwicklung eingebetteter Systeme gefördert.

In Abbildung 4.1 werden die Vorarbeiten zu den drei Komponenten didaktischer Systeme zur Förderung von Kompetenzen für das Design eingebetteter Systeme sowie die Forschungsarbeiten im Projekt KOMINA exemplarisch abgebildet (Blöcke von links nach rechts). Zur Komponente kognitive Strukturen wird die Vernetzung von Kompetenzen erforscht. Nicht-kognitive Dimensionen werden hier am Rande betrachtet. Ferner erfolgt hier eine Ausdifferenzierung von Kompetenzsubdimensionen und die Analyse kognitiver Anforderungen an Studierende, welche die impliziten Mikrosystemkenntnisse einschließen sowie Pfade der Kompetenzaneignung, resultierend aus Relationen zwischen Fachkonzepten der Kompetenzfacetten, einbeziehen. Aufgabenklassen wurden bislang wenig erforscht und beschränken sich in

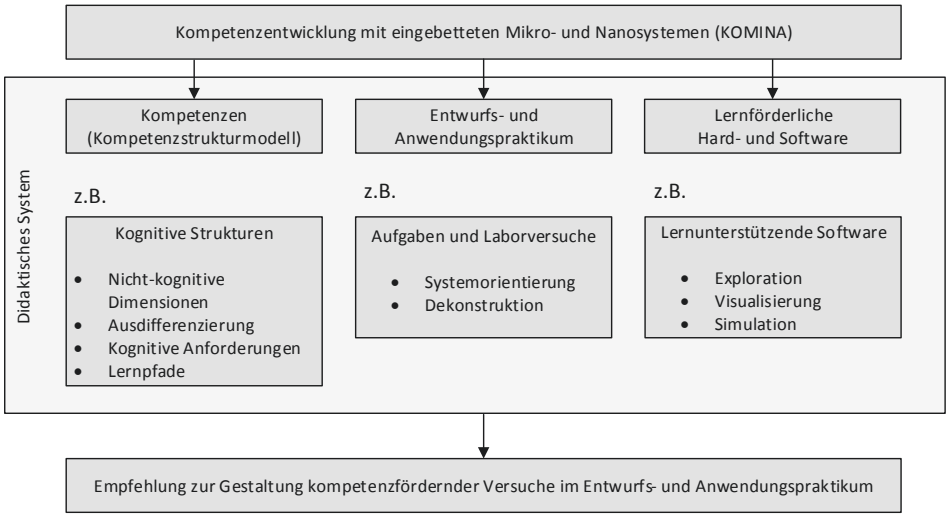


Abbildung 4.1.: Didaktische Systeme als Strukturierungshilfe

dieser Arbeit auf die Betrachtung eines systemorientierten Zugangs zu eingebetteten Systemen. Die Dekonstruktion eines gegebenen Systems stellt eine Alternative zur ausschließlichen Neuimplentierung von Komponenten dar und ist an konkrete Lehr-Lernprozesse und die Zielgruppe anzupassen. Hier fließen Erkenntnisse aus dem Entwurfs- und Anwendungspraktikum ein. Lernförderliche Software allein kann nicht zur Förderung von Kompetenzen für das Entwickeln eingebetteter Systeme beitragen, sondern ist an die gegebene Hardwareumgebung anzupassen. Ferner ist die Entwicklung beziehungsweise der Einsatz spezieller Hardware für die Lehre anzustreben. Die entwickelte lernunterstützende Software umfasst die Simulation und Visualisierung des Lerngegenstandes *Modellhaus* und ermöglicht die Exploration von Sprachkonstrukten in C.

Das empirisch verfeinerte Kompetenzstrukturmodell (vgl. Abschnitt 3.2) dient als Ausgangspunkt für die Ableitung kognitiver Strukturen. Eine Strukturierung, welche die bislang isolierten Kompetenzsubdimensionen verbindet, existiert bislang nicht, wenngleich auf einer abstrakten Ebene beispielsweise die Auswirkungen der nicht-kognitiven Kompetenzen auf die kognitiven Dimensionen beschrieben wurden (siehe Anhang A.1). Die kognitiven Anforderungsprofile an die Studierenden werden durch eine exemplarische Ausdifferenzierung der Kompetenzsubdimension *(K2.3) Design* präzisiert. Die Aufgaben des Entwurfs- und Anwendungspraktikums werden aufgrund der identifizierten Lernhürden verfeinert und nach einem systemorientierten Ansatz mit Phasen der Dekonstruktion erweitert (vgl. Abschnitt 4.5). Eine Voranalyse dazu erfolgte in Kapitel 3. Die entwickelten Lernunterstützungen (vgl. Abschnitt 3.3.3) werden durch Software zur Visualisierung und Simulation

erweitert. Diese Arbeit leistet damit einen wesentlichen Beitrag zur Grundlagenforschung zu didaktischen Systemen für das Design eingebetteter Systeme.

4.1.1. Kognitive Strukturen und nicht-kognitive Dimensionen

Der Autor verwendet den Begriff der *kognitiven Strukturen* in Anlehnung an Kluwe und Dörner (vgl. [Edelmann und Wittmann, 2012, S. 197]) als Kombination von heuristischen und epistemischen Strukturen, also die Gesamtheit der Voraussetzungen zur Aufgabenbewältigung und Problemlösung. Heuristiken kommen dann zum Einsatz, wenn die Wissensstrukturen der Studierenden nicht zur Aufgabenbewältigung ausreichen, also eine noch unbekannte Folge von Schritten in komplexen Handlungssituationen (Problemen) zu bewältigen sind.

> „Ein Individuum steht einem Problem gegenüber, wenn es sich in einem inneren oder äußeren Zustand befindet, den es aus irgendwelchen Gründen nicht für wünschenswert hält, aber im Moment nicht über die Mittel verfügt, um den unerwünschten Zustand in den wünschenswerten Zielzustand zu überführen“ [Dörner, 1979, S. 10].

Die Methode von Versuch und Irrtum ist ein Beispiel für Heuristiken, welche in der Evaluation des Entwurfs- und Anwendungspraktikums als ungeeignet herausgestellt wurde. Zwar ist eine wiederholte Überprüfung von Hypothesen durch erneutes Ausprobieren in bestimmten Kontexten sinnvoll. Wird aber nur scheinbar eine Hypothese gebildet, beispielsweise durch das unüberlegte Verändern einzelner Parameter in der Programmierung, so kann nicht von planvollem Handeln zur Förderung von Kompetenzen ausgegangen werden. Wesentlich für die Bildung geeigneter Heuristiken, wie der *Zerlegung in Teilprobleme* oder der *Umstrukturierung*, ist das Sammeln von Erfahrungen (vgl. [Edelmann und Wittmann, 2012]). Diese Erfahrungen können aufgrund des Studienverlaufes der Zielgruppe im Entwurfs- und Anwendungspraktikum noch nicht sehr ausgeprägt sein, sodass bei der Gestaltung von Lehr-Lernprozessen nur kleinschrittige Problemstellungen und nicht die selbständige Lösung des gesamten Anwendungsproblems, Regelung des peripheren technischen Prozesses, beispielsweise Temperatur, bestehen dürfen.

Weinert beschreibt Kompetenzen als die erlernbaren kognitiven Fähigkeiten und Fertigkeiten diesen wünschenswerten Zustand herbeizuführen und schließt die nicht-kognitive Dimension mit ein, was die Forschungsergebnisse in KOMINA uneingeschränkt stützen. Zu den zwölf wichtigsten Kompetenzen des erforschten Kompetenzstrukturmodells für das Entwickeln eingebetteter Mikro- und Nanosysteme (Klasse A) gehören vier nicht-kognitive Kompetenzfacetten. Die von Weinert aufgeführten motivationalen und volitionalen Fähigkeiten und Fertigkeiten (*K4.3*) wurden von den Experten für die Entwicklung eingebetteter Systeme als am wichtigsten eingestuft – noch über die Kompetenzen für das Design und die Implementierung eingebetteter Systeme hinaus (siehe Tabelle 3.3). Ein Grund dafür ist, dass diese Kompetenzfacetten die Förderung der Kompetenzen anderer Dimen-

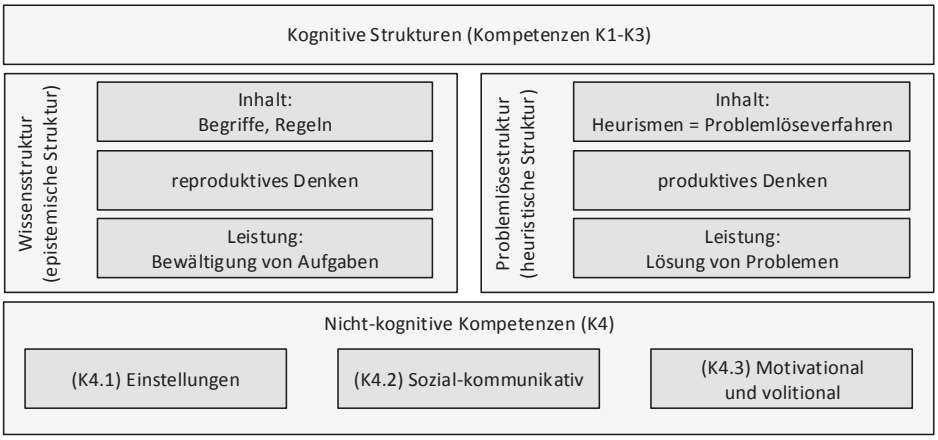


Abbildung 4.2.: Erweitertes Modell der kognitiven Strukturen (vgl. [Edelmann und Wittmann, 2012])

sionen sowie grundsätzlich die Motivation zu einer Spezialisierung innerhalb der Informatik wesentlich beeinflussen. In den vorgestellten Empfehlungen zu Curricula nehmen die nicht-kognitiven Kompetenzen keine besondere Stellung ein, da diese nicht den Inhaltsbereichen zugeordnet werden können. Dennoch werden Eigenschaften künftiger Entwickler beispielsweise in den Präambeln der Empfehlungen aufgeführt. Eine Strukturierung in einer eigenen Dimension, welche eine Förderung aller weiteren Kompetenzen wesentlich beeinflusst, ist daher angemessen und bei der Gestaltung von Laborversuchen explizit zu beachten. Für einen verantwortungsvollen Einsatz der technologischen Möglichkeiten unter Beachtung von ethisch-moralischen Gesichtspunkten ist daher nach Möglichkeit auch eine kritische Diskussion in hardwareorientierten Praktika zu verorten. In einer an Technik orientierten Wissenschaft, die alle Lebensbereiche des Menschen tangiert, geht ansonsten der diese auszeichnende sozio-technische Bezug unter. Technische Machbarkeit sollte daher in Bezug auf die Auswirkungen auf das sozio-technische System untersucht werden. Bei der Entwicklung eines didaktischen Systems für das Design eingebetteter Systeme sollte daher die Komponente der kognitiven Strukturen die Möglichkeit der Integration von nicht-kognitiven Bestandteilen bieten.

Wissensstrukturen

Wissensstrukturen, also fachdidaktisches Metawissen, haben sich als geeignet zur Strukturierung von Lehr-Lernprozessen erwiesen und erfüllen dazu drei didaktische Funktionen – Orientierungsfunktion, Organisationsfunktion und Diskussionsfunktion (vgl. Abschnitt 2.5.3). Zur Darstellung können Graphen verwendet werden, deren Knoten Wissenselemente beziehungsweise Lerneinheiten repräsen-

tieren. Die Transitionen beschreiben die Art der Relation. [Brinda, 2004] nutzt Und-Oder-Graphen, bei denen eine Oder-Verknüpfung *ist hilfreich für* und eine Und-Verknüpfung *ist notwendig für* modelliert. Die Anforderungen an die Darstellungsform beschreibt [Freischlad, 2010, S. 75ff] mit Bezug auf [Brinda, 2004, S. 181f] mit den Kriterien Ausdrucksstärke, Übersichtlichkeit und Nachvollziehbarkeit. Fünf Arten zufallsfreier Relationen (R) definieren eine Sequenzierung von Fachkonzepten in Lehr-Lernprozessen (vgl. Abschnitt 2.5.3).

R1: Beobachtbare Objekte vor nicht sichtbaren Abläufen.

R2: Zuerst das untergeordnete Fachkonzept und dann das übergeordnete Fachkonzept.

R3: Einzelne Elemente vor zusammengesetzten Wissenselementen.

R4: Ein anschauliches Beispiel zur Illustration, als Kontext oder zur Schaffung eines Problembewusstseins.

R5: Deklaratives Wissen vor Handlungswissen.

[Schwidrowski, 2010] empfiehlt, den fünften Beziehungstyp nicht unreflektiert zu übernehmen, da der Lernende vom Handeln über Können zum Wissen geführt werde (vgl. [Jank und Meyer, 2005]). Bezogen auf komplexe Anforderungssituationen, wie dem Design eingebetteter Systeme, kann dem nicht zugestimmt werden. Erfahrungen aus der formativen Evaluation des Entwurfs- und Anwendungspraktikums (siehe Abschnitt 3.4) haben gezeigt, dass kein strukturiertes Vorgehen möglich ist, ehe deklaratives Wissen – bezogen auf die Terminologie und naturwissenschaftliche Grundlagen – vorliegt. Natürlich kann diese Aussage nicht für alle Kompetenzen gleichermaßen gelten. Das isolierte Aneignen von Wissen über eine Programmiersprache (Syntax und Semantik) führt nicht zum Handeln, obwohl dieses Wissen schrittweise durch Versuche erarbeitet werden kann. Dies gilt solange, bis über Basisstrukturen (Schleifen, Bedingungen, Funktionen etc.) zu hardwarenahen Programmierkonzepten übergegangen wird. Hier kann die Programmierung nicht mehr isoliert betrachtet werden, sondern hängt von der Architektur, den verwendeten Technologien sowie dem zu regelnden Prozess ab. Da bei der hardwarenahen Programmierung das Verhalten des Systems auch von äußeren Faktoren abhängig ist, verfügen die Studierenden nicht über die Fähigkeit das nach außen sichtbare Verhalten des Systems nur auf einen Aspekt der Programmierung zurückzuführen. Ein Erkenntnisgewinn aus Implementierungsvarianten – durch Anwendung einer Versuch-Irrtum-Strategie – ist damit nicht mehr möglich, sodass auch Handeln nicht zum Wissen führt. Je nach Lehr-Lernprozess ist es also notwendig, zunächst deklaratives Wissen über die Umwelt, Randbedingungen und weitere systembeeinflussende Faktoren zu vermitteln. Das muss bei der Reihenfolge von Laborversuchen und damit auch bei der Beschreibung kognitiver Strukturen im Sinne dieser Arbeit berücksichtigt werden.

Ausgehend von der Integration von Lehr-Lernprozessen nach dem Ansatz der Dekonstruktion existiert derzeit keine entsprechende Relation. Bei der Dekonstruktion wird erst das zusammengesetzte Wissen, dann das einzelne Element betrachtet. Demnach scheint ein Widerspruch zwischen dem Ansatz der Dekonstruktion und der zufallsfreien Beziehung R3 zu bestehen. Eine neue Relation, in der zunächst der übergeordnete Zusammenhang von Komponenten und anschließend mögliche Designentscheidungen untersucht werden, wäre eine Möglichkeit zur Erweiterung der Relationen von kognitiven Strukturen. Die Intention zu deren Einführung ist allerdings analog zur Relation des anschaulichen Beispiels (R4). Für diese Arbeit ist diese Relation um die Dekonstruktion eines Teilsystems zu erweitern zu: *R4, ein anschauliches Beispiel beziehungsweise existierendes Teilsystem zur Illustration, als Kontext, zur Schaffung eines Problembewusstseins und als Dekonstruktionsobjekt.* Das anschauliche Beispiel dient vorwiegend der Motivation für folgende Lerneinheiten. Ergänzt man den systemorientierten Ansatz der Dekonstruktion, sind Beispiele zu wählen, die zur Dekonstruktion geeignete Teilsysteme oder Komponenten enthalten, welche nicht als Black-Box betrachtet werden müssen – beispielsweise aufgrund einer für die Studierenden nicht auflösbaren Komplexität.

Damit bleiben in dieser Arbeit die Relationen der Wissenselemente erhalten, beziehen das Konzept der Dekonstruktion mit ein und eignen sich auch zur Beschreibung kognitiver Strukturen, da Wissen als Voraussetzung zum planvollen Handeln – durch Erfahrung – auch im Begriff der kognitiven Strukturen verankert ist.

Heuristische Strukturen

Im Vergleich zum allgemeinbildenden Informatikunterricht, in dem reproduktives Denken gefordert und gefördert wird, soll im Informatikstudium dem produktiven Denken und Lösen von Problemen eine stärkere Rolle beigemessen werden. Weinert (vgl. Kapitel 3) nahm direkt Bezug zu Problemlösefähigkeiten. Insbesondere in technischen und naturwissenschaftlichen Studiengängen steht die Lösung von Problemen im Vordergrund.

> „Aufgaben sind geistige Anforderungen, für deren Bewältigung Methoden bekannt sind. [...] Aufgaben erfordern nur reproduktives Denken, beim Problemlösen muss etwas Neues geschaffen werden“ [Dörner, 1979, S. 10].

Die Aussage „etwas Neues geschaffen werden“ meint nicht die Ebene *Erschaffen* der Taxonomie nach [Fuller et al., 2007], sondern den Erkenntniszugewinn durch Verknüpfung bereits bekannter Konzepte, beispielsweise durch eine neue Sicht auf den Problembereich und Erweiterung des Lösungsvorrates. In der formativen Evaluation des Entwurfs- und Anwendungspraktikums wurde festgestellt, dass die Studierenden oftmals eine Versuch-Irrtum-Strategie zum Überwinden der Lernhürden angewandt haben, wie es in unübersichtlichen Problemsituationen üblich ist [Edelmann und Wittmann, 2012]. Unübersichtlich sind diese deshalb, weil den Studierenden Vorwissen und Erfahrung in Laborpraktika fehlte und andererseits

eine mangelnde Orientierung im Fach festzustellen war (vgl. Abschnitt 3.4). In einer nicht-teilnehmenden Beobachtung ist allerdings nicht zweifelsfrei erkennbar, ob es sich bei einem Lösungsansatz um die Überprüfung von Hypothesen oder eine Versuch-Irrtum-Strategie handelt.

> „Allerdings liegt bei Menschen selten ein blindes Ausprobieren vor. Vielmehr zeigt sich wenigstens ansatzweise die Anwendung von Strategien als sukzessive Prüfung von Hypothesen“ [Edelmann und Wittmann, 2012].

Durch die Vielzahl beobachteter Fehlversuche kann jedoch meistens nicht von einem planvollen Handeln ausgegangen werden. Das gewünschte Verhalten durch das Verändern einzelner Parameter zu erreichen kann nur dann als sukzessive Prüfung von Hypothesen bezeichnet werden, wenn die Studierenden sich nicht schrittweise an die Lösung herantasten, sondern konkrete Vorstellungen von den Auswirkungen ihrer Implementierungsvariante, beispielsweise Registerbelegungen, auf das System besitzen und in der Arbeitsgruppe diskutieren. Es konnte jedoch beobachtet werden, dass die Parameter ohne eine Diskussion mehrfach verändert und getestet wurden. Eine Reflexion der Ergebnisse innerhalb einer Arbeitsgruppe erfolgte zumeist nicht.

Sollen die Studierenden eigene Problemlösestrategien entwickeln, so sind auch Laborphasen zu gestalten, welche ein Problem und nicht eine Folge von isolierten Aufgaben betrachten. Dieses Problem darf dann nicht ausschließlich durch eine Barriere, resultierend aus einer mangelnden epistemischen Struktur, bestehen, sondern muss durch die Beschreibung des unerwünschten Anfangszustands und des erwünschten Zielzustands definiert werden. Der erwünschte Zielzustand ist in Form des Modellhauses und seiner Funktionen gegeben. Alle Komponenten und Werkzeuge zur Entwicklung der Steuerung und Regelung sind bereitgestellt. Die Existenz eines Problems innerhalb einzelner Versuche hängt von den Vorerfahrungen der Studierenden ab. Einige Studierende besaßen bereits Kompetenzen im Bereich der Implementierung von Mikrocontrollern, sodass sie ihre Fähigkeiten auf den Prozess und Lerngegenstand übertragen konnten. Die Ergebnisse aus der ersten formativen Evaluation lassen den Schluss zu, dass die Studierenden aufgrund ihrer Vorerfahrungen auch einfache Steuerungen als Problem auffassen.

4.1.2. Anforderungen an die Visualisierung kognitiver Strukturen

Die Erfüllung der Kriterien Ausdrucksstärke, Übersichtlichkeit und Nachvollziehbarkeit stellt bei der Visualisierung kognitiver Strukturen in der technischen Informatik eine besondere Herausforderung dar. Einerseits kann davon ausgegangen werden, dass die Zielgruppe im Forschungsprojekt – verglichen mit Schülerinnen und Schülern an allgemeinbildenden Schulen – in der Lage ist komplexere Visualisierungen kognitiver Strukturen nachzuvollziehen, andererseits sind die Fachkonzepte und intendierten Kompetenzen vielschichtiger, was insbesondere an den unterschiedlichen kognitiven Anforderungsstufen liegt.

Offensichtlich ist, dass immer ein Kompromiss zwischen einer möglichst schlichten Darstellungsform mit wenigen Mitteln sowie einer hohen Ausdrucksstärke und damit einem sinnvollen navigieren im Lehr-Lernprozess gefunden werden muss. Dies gilt für die Darstellung der Fachkonzepte als Knoten und für die Kanten zur Definition einer Relation kognitiver Strukturen. [Freischlad, 2010] erweitert die Knoten um Tabellen, welche Grob- und Feinziele innerhalb der Unterrichtseinheit beschreiben. Die Übersichtlichkeit der Darstellung von Wissensstrukturen ändert sich damit nicht, wobei die Ausdrucksstärke nur scheinbar zunimmt. Vielmehr ist die tabellarische Darstellung ein zusätzlicher Verfeinerungsschritt, der die Navigation innerhalb einer Unterrichtseinheit ermöglicht. Aus den Erkenntnissen der Diskussion zur Kompetenzorientierung in der technischen Informatik, in der zunächst die Struktur von Kompetenzen erforscht wurde sowie den Erfahrungen aus der exemplarischen Umsetzung des Kompetenzstrukturmodells in einem Entwurfs- und Anwendungspraktikum lassen sich in Anlehnung an [Brinda, 2004] angepasste Anforderungen an die Visualisierung kognitiver Strukturen begründen.

Übersichtlichkeit

Die Übersichtlichkeit beschreibt den Anspruch, dass der Grad der Detaillierung beziehungsweise der Abstraktion angemessen ist und nur wenige Darstellungsformen, wie Knoten und Kanten, zu verwenden sind (vgl. [Brinda, 2004]).

Die Orientierungsfunktion steht in dieser Arbeit nicht im Vordergrund. Studierende sollen sich nicht nur anhand der kognitiven Struktur leiten lassen, sondern selbständig im Lehr-Lernprozess orientieren. Zunächst ist die Strukturierung von Lehr-Lernprozessen in einem didaktischen System mithilfe der kognitiven Strukturen zu erforschen. Das Entwurfs- und Anwendungspraktikum und dessen Weiterentwicklung dienen als exemplarische Umsetzung des empirisch verfeinerten Kompetenzstrukturmodells. Den Studierenden kann das abstrakte, auf Komptenzsubdimensionen sowie wenige Kompetenzfacetten beschränkte Modell nicht zur Navigation innerhalb des Praktikums dienen. Zunächst werden im Rahmen dieser Arbeit erste Erkenntnisse bezüglich möglicher Lernpfade zur Förderung von Kompetenzen für das Entwickeln eingebetteter Systeme gewonnen. Ziel ist die Bildung einer Diskussionsgrundlage für Forscher und Lehrende, was erst im darauf folgenden Schritt die Entwicklung konkreter Lehr-Lernprozesse unterstützen soll. Die Diskussionsfunktion als Mittel zur Grundlagenforschung in der Hochschuldidaktik der technischen Informatik und die Organisationsfunktion bei der Gestaltung von Lehr-Lernprozessen (für die Lehrenden) ist daher im Rahmen dieses Forschungsprojektes von höherer Priorität.

Ausdrucksstärke

Ausdrucksstärke ist die Voraussetzung dafür, dass Lehrende und Lernende sich anhand der kognitiven Strukturen im Lehr-Lernprozess orientieren können (vgl. [Brinda, 2004]).

Aufgrund der beschriebenen Priorisierung zugunsten der Diskussionsfunktion ist die Ausdrucksstärke sehr wichtig, um kognitive Strukturen umfassend darstellen und beschreiben zu können. Dafür muss die zu wählende Darstellungsform verschiedene Aspekte einer Kompetenz und deren Relationen zueinander gleichzeitig erlauben. Der Autor verzichtet insoweit bewusst auf eine Verfeinerung auf Lernzielebene, da dieses Konzept – die informatikdidaktische Verfeinerung – als Schnittstelle zwischen dem empirisch überprüften Kompetenzstrukturmodell und der Gestaltung konkreter Lehrveranstaltungen fungieren soll. Referenzen auf das noch zu verfeinernde Kompetenzstrukturmodell für das Entwickeln eingebetteter Mikro- und Nanosysteme eignen sich damit als Beschreibung der Knoten und Fachkonzepte, welche dann vom Lehrenden an konkrete Lehr-Lernprozesse angepasst werden können. Die Darstellungsform muss die zu fördernde Kompetenz, deren kognitive Stufe sowie die Relationen zu Kompetenzen und Voraussetzungen (Vorwissen) beinhalten. Außerdem ist zu konkreten Lehr-Lernprozessen die Abstraktionsebene anzugeben.

Nachvollziehbarkeit

Die Nachvollziehbarkeit beschreibt das Ziel, eine leicht verständliche standardisierte Darstellungsform zu verwenden (vgl. [Brinda, 2004]).

Auf der einen Seite ermöglicht die Verwendung einer solchen Darstellungsform eine Verarbeitung mit existierenden Werkzeugen, andererseits handelt es sich bei der Visualisierung kognitiver Strukturen um ein spezielles Konzept zur theoretischen Fundierung von Lehr-Lernkonzepten, welches eher von einem erweiterten Satz an darauf angepassten Beschreibungsmitteln profitiert, als von einem existierenden, aber dafür eingeschränkten Werkzeug. Die wenigen Beschreibungsmittel für Fachkonzepte, Relationen, Abstraktionsebenen und kognitive Anforderungen lassen sich mit diversen grafischen Werkzeugen und einfachen geometrischen Formen veranschaulichen.

Abhängigkeit kognitiver Strukturen vom Kompetenzniveau

Das intendierte Kompetenzniveau, welches einem bestimmten Fachkonzept zugewiesen wird, bestimmt den Aufbau der zugehörigen kognitiven Struktur und legt damit die Vorwissensbeziehungen fest. Im Rahmen dieser Arbeit werden immer konkrete Lehr-Lernprozesse thematisiert, da die zu fördernden Kompetenzen und deren Niveaustufen von diesen abhängen und nur in Grenzen generalisierbar sind. Soll eine kognitive Struktur für konkrete Lehr-Lernprozesse oder eine bestimmte Lernhürde abgeleitet werden, muss die kognitive Stufe aus der formulierten Kompetenzbeschreibung hervorgehen. Gezeigt wird die Notwendigkeit der Integration von Niveaustufen in kognitiven Strukturen am Beispiel des Konzeptes Analog/Digital-Wandlung. Im Entwurfs- und Anwendungspraktikum ist der zu verwendende Wandler aufgrund des zur Verfügung stehenden Systems (AVR ATxemga128A1 Explained) gegeben. Dies muss nicht zwingend der Fall sein, sodass Studierende

in ganzheitlichen Entwicklungsprojekten auch einen Wandler auswählen müssen. Aus der Kompetenzbeschreibung „die Studierenden sollen in der Lage sein, einen Analog/Digital-Wandler zu verwenden“ lassen sich verschiedene Kombinationen von Wissensstrukturen abhängig von der Intention der Lehrperson ableiten (z. B. Abbildung 4.3).

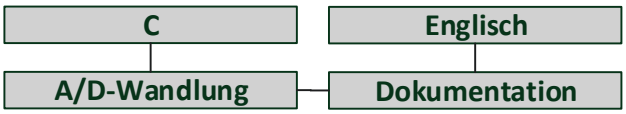


Abbildung 4.3.: Fachkonzepte einer Wissensstruktur Analog/Digital-Wandlung – Variante 1

Bei vorhandener Umgebung und vorgegebenem Wandler müssen die Studierenden die Dokumentation verstehen, um diesen beispielsweise zu konfigurieren (Registerbelegung eines Mikrocontrollers). Dazu ist das Anwenden der englischen Sprache notwendig, da hier das Verhalten des Wandlers sowie die Rahmenbedingungen und Einschränkungen beschrieben werden.

Eine von dieser Variante abweichende Struktur ergibt sich zur Kompetenzbeschreibung „die Studierenden sollen in der Lage sein einen Analog/Digital-Wandler für eine Domäne auszuwählen und zu verwenden“ (z.B. Abbildung 4.4). Hier muss zunächst die Domäne, also der periphere technische Prozess, analysiert werden, um einen geeigneten Wandler auswählen zu können. Dazu gehören physikalische Grundlagen, wie die Arbeitsweise von Sensoren und Aktoren, um beispielsweise eine Temperatur in elektrische Signale zu wandeln. Je nach Art des Signals sind Kenntnisse über das Abtasttheorem notwendig.

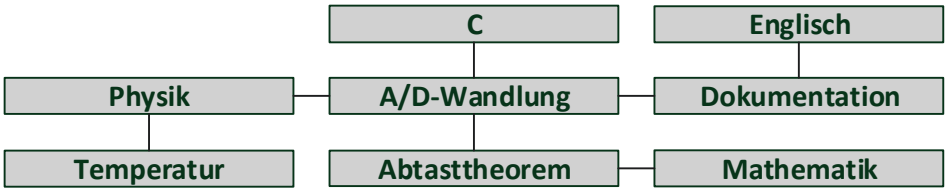


Abbildung 4.4.: Fachkonzepte einer Wissensstruktur Analog/Digital-Wandlung – Variante 2

Die vorgenannten Ausführungen zeigen, dass die Visualisierung einer kognitiven Struktur von der kognitiven Stufe und dem Kontext, dem Lehr-Lernprozess, abhängig ist. Wichtig für die Ableitung kognitiver Strukturen ist, dass für jede inten-

dierte Kompetenz im konkreten Lehr-Lernprozess eine Strukturierung abgeleitet werden muss und keine allgemeine kognitive Struktur existieren kann.

Für die Beschreibung in Curricula sind kognitive Strukturen aufgrund dieser speziellen Sicht auf Kompetenzen nicht geeignet. Bei der informatikdidaktischen Verfeinerung des Kompetenzstrukturmodells kommt ihnen jedoch eine besondere Bedeutung zu, da Kompetenzen unterschiedlicher Dimensionen in einer gemeinsamen Beschreibung strukturiert werden und eine an konkrete Lehr-Lernprozesse angepasste Gliederung vorschlagen. Die Diskussionsfunktion didaktischer Systeme wird dadurch gefördert und Lehrende bei der Gestaltung fachdidaktisch begründeter Lehr-Lernprozsse unterstützt.

Aus den Abbildungen 4.3 und 4.4 geht hervor, dass die Darstellung der Fachkonzepte analog zu bisherigen Wissensstrukturen didaktischer Systeme das Kriterium der Ausdrucksstärke nicht erfüllen und um die Bezeichnung der intendierten Niveaustufe zu kognitiven Strukturen ergänzt werden müssen.

Die Taxonomie nach [Fuller et al., 2007] (vgl. Kapitel 2) hat sich zur Beschreibung der kognitiven Stufe von Kompetenzen der technischen Informatik als geeignet erwiesen. Um die Übersichtlichkeit in der Visualisierung kognitiver Strukturen zu bewahren wird jedem Fachkonzept, repräsentiert als Knoten, ein Bezeichner der Taxonomie hinzugefügt. Der Bezeichner besteht aus einem Buchstaben für produzierende Tätigkeiten (E für Erschaffen, A für Anwenden) beziehungsweise keinem Buchstaben für interpretierende Tätigkeiten (siehe Abbildung 4.5). Eine Zahl beschreibt die Spalte in der Taxonomie (1=Erinnern, 2=Verstehen, 3=Analysieren, 4=Evaluieren).

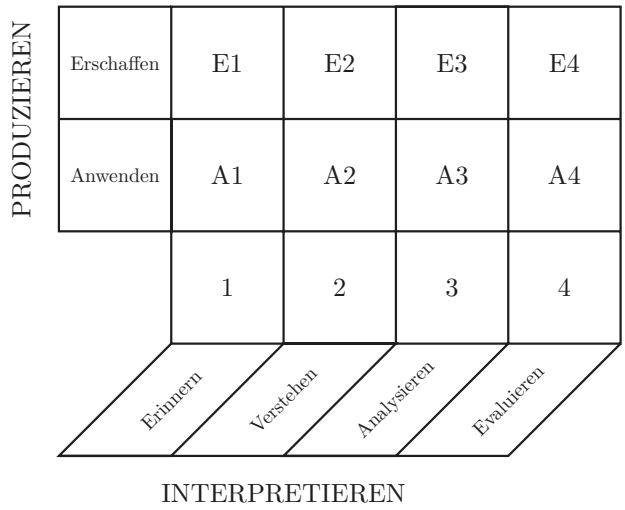


Abbildung 4.5.: Zuordnung zur Taxonomie [Jaschke, 2013]

Die Zuordnung von Niveaustufen zu den Tätigkeiten von Entwicklern eingebetteter Systeme ist dann problematisch, wenn die Begriffe in Aufgabenstellungen nicht korrekt verwendet werden und die eigentliche Tätigkeit von der Beschreibung abweicht (vgl. Kapitel 3). Zur Verdeutlichung dafür sind typische Aufgaben im Entwurfs- und Anwendungspraktikum zweckmäßig, welche repräsentativ für Lehrveranstaltungen der technischen Informatik sind, insbesondere mit Mikrocontrollern als Lerngegenstand. Beispielhaft sind hier Aufgaben zu nennen, in denen die Erfassung analoger Werte sowie einfache Steuerungen oder Regelungen programmiert werden. Existiert bereits ein Programm, in dem Parameter an die veränderte Domäne angepasst werden, entspricht dies der *Adaptierung (A2)* (siehe Abbildungen 4.5 und 2.1). Existiert ein Design, also eine vollständige Beschreibung der Lösungsstruktur für das Problem, wird diese Tätigkeit der *Implementierung (A2)* zugeordnet. Für die Studierenden im Entwurfs- und Anwendungspraktikum existiert aber zur Regelung des Systems keine vollständige Lösungsstruktur, also vorgegebene Komponenten. Große Teile einfacher Regelungen müssen selbständig entwickelt werden, weshalb die Fachkonzepte nicht im Sinne des Begriffs *Implementierung* codiert werden können. Hier wird, je nach Vorerfahrung der Studierenden, implizit die Lösungsstruktur ermittelt sowie strukturiert (Design) und anschließend implementiert, was den Anschein einer ausschließlichen Implementierung erweckt. Die kognitive Anforderung ist eher den Stufen *(E2/E3)* zuzuordnen (vgl. Abbildung 3.17). Dies stellt bei der Ableitung der kognitiven Strukturen ein Indiz für die notwendige Aufteilung in einzelne Kompetenzen, welche unabhängig voneinander betrachtet werden müssen, dar.

Als Beispiel für die Zuordnung eines Fachkonzeptes zur Taxonomie dient der Analog/Digital-Wandler. Die Studierenden sollen in der Lage sein

1 sich an das Konzept der Analog/Digital-Wandlung zu erinnern,

2 das Funktionsprinzip eines Analog/Digital-Wandlers zu erläutern,

3 verschiedene Funktionsprinzipien zu differenzieren,

4 eingesetzte Funktionsprinzipien in gegebenen Lösungen zu beurteilen,

A1 einen gegebenen Analog/Digital-Wandler zu verwenden,

A2 einen Analog/Digital-Wandler in neuen Domänen zu verwenden,

A3 abhängig von der Domäne einen geeigneten Analog/Digital-Wandler zu verwenden,

A4 (entfällt),

E1 (entfällt),

E2 einen Analog/Digital-Wandler zu entwickeln,

E3 einen Analog/Digital-Wandler für eine neue Domäne zu entwickeln

E4 und die Qualität eines Analog/Digital-Wandlers zu beurteilen und zu verbessern.

Markierung des Lernstandes in kognitiven Strukturen

[Brinda, 2004] grenzt Wissensstrukturen von Graphen zur Repräsentation kognitiver Strukturen ab, da diese einen Zustand definieren, Wissensstrukturen jedoch der Beschreibung von Wissenserwerb, also einem Prozess, dienen. Betrachtet man die Organisationsfunktion in didaktischen Systemen ist die Zustandsbeschreibung von nicht unerheblichem Wert, da sowohl Lehrende als auch Studierende in der Lage sind den Fortschritt innerhalb des Lehr-Lernprozesses zu verfolgen. Ferner genügt eine Unterscheidung allein nach diesem Kriterium (Zustand/Prozess) nicht.

> „Knoten im Graph für Wissensstrukturen erfüllen ohne Einschränkung die Darstellung des Lernstands, wenn jedem Knoten ein Grobziel und mehrere operationalisierte Feinziele zugeordnet werden. [...] In Wissensstrukturen wird der Lernstand durch das Erreichen von Knoten unabhängig voneinander beschrieben“ [Freischlad, 2010, S. 90f].

Durch die Markierung aller erreichten Knoten ist also implizit ein Zustand (Lernstand) beschrieben. Das heißt, dass zur Verfolgung der Kompetenzförderung Fachkonzepte und dazu erzielte kognitive Stufen kenntlich gemacht werden können. Da des Weiteren Abhängigkeiten zwischen den Kompetenzen und den Voraussetzungen existieren, impliziert das Erreichen einer Kompetenz auch das Erreichen weiterer vorausgesetzter Kompetenzen, welche in kognitiven Strukturen beschrieben werden können. Eine allgemeingültige Modellierung dieser Beziehungen ist – wie bereits erläutert – weder möglich noch sinnvoll, da sie immer vom konkreten Lehr-Lernprozess abhängt.

Darstellung kognitiver Strukturen

[Freischlad, 2010] repräsentierte eine Unterrichtseinheit mit je einem Knoten, weshalb es nicht möglich war das zugrunde liegende Fachkonzept und ein konkretes Produkt zugleich darzustellen. In dieser Arbeit werden kognitive Strukturen zu Fachkonzepten abgeleitet, welche in direktem Zusammenhang zu einer der identifizierten Lernhürden (LH_1-LH_4) stehen. Alternativ eignen sich kognitive Strukturen für eine Vorgliederung einer Lehrveranstaltung, mit den wichtigsten Teilen eines Systems als Ausgangspunkt. Dann ist eine Mischform von Fachkonzepten und Systemkomponenten sinnvoll, welche später schrittweise spezifiziert werden. Durch die Relationen zwischen Fachkonzepten, Systemkomponenten, anschaulichen Beispielen etc. (R1-R5, vgl. Kapitel 2) ist implizit die Reihenfolge der Kompetenzaneignung gegeben, welche jedoch bei gleichrangigen Vorkenntnissen Spielraum bei der Strukturierung von Lehr-Lernprozessen bietet.

Zu beachten ist, dass aufgrund fehlender Instrumente zur Erfassung von Kompetenzen und deren Facetten derzeit keine validierte Methode zur Markierung

von Knoten kognitiver Strukturen existiert. Dennoch lassen subjektiv geprägte Eindrücke durch Beobachtungen einen Schluss auf den Lernstand der Studierenden zu. Da dieser von der kognitiven Struktur und damit vom konkreten Lehr-Lernprozess abhängig ist, kann keine Markierung des Lernstandes bezogen auf die gesamte Entwicklung eingebetteter Systeme erfolgen, sondern beschränkt sich auf einen Ausschnitt zur aktuellen Problemstellung (aktuelle kognitive Struktur).

Zusammenfassend müssen zur Darstellung kognitiver Strukturen Lernstände, Sequenzen der Kompetenzförderung, Vorbedingungen, kognitive Stufen sowie die Abstraktionsebene zugleich beschrieben werden. Ferner ist die Art der Relation als Begründung der Reihenfolge und gegebenenfalls damit in Verbindung stehende heuristische Strukturen, also Arten der Problembewältigung, welche hier gefördert werden sollen, aufzuführen. In Abbildung 4.6 ist eine solche Relation angegeben. Jedes Rechteck repräsentiert eine Kompetenzsubdimension beziehungsweise Kompetenz aus dem empirisch verfeinerten Kompetenzstrukturmodell respektive ein Teilsystem. Um einen Bezug zur Beschreibung der zugehörigen Kompetenzfacetten herzustellen, wird das Rechteck mit der zugehörigen Nummer versehen (A). Die bei der Analyse von Lernhürden im Vordergrund stehenden Kompetenzen werden mit der im konkreten Lehr-Lernprozess angestrebten kognitiven Stufe versehen (B). Die gerichteten Kanten beschreiben die Art der zufallsfreien Relationen, welche die Erarbeitungsreihenfolge zum Erreichen der intendierten Kompetenz definieren (C). Die kognitive Stufe der notwendigen Kompetenzen wird am Anfang der Kante notiert (D). Bedingt durch die Abhängigkeit der kognitiven Struktur von der Abstraktionsebene, auf der das Fachkonzept betrachtet wird, erfolgt deren Darstellung mit (E). Dadurch ist es möglich, konkrete Lehr-Lernprozesse hinsichtlich ihrer kognitiven Anforderungen (Stufen) zu untersuchen und transparent zu visualisieren, wobei hier die Ausdrucksstärke zur Gewährleistung der Diskussionsfunktion von didaktischen Systemen im Vordergrund steht.

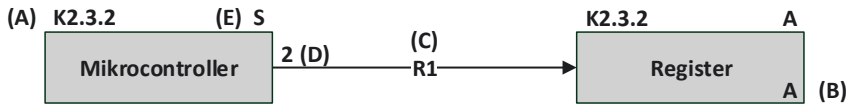


Abbildung 4.6.: Visualisierung kognitiver Strukturen

4.2. Ausdifferenzierung der Subdimension K2.3 Design

Bereits bei der Gestaltung von Laborversuchen wurde deutlich, dass eine Ableitung von geeigneten Aufgaben aufgrund der Breite des Kompetenzstrukturmodells nur schwer möglich ist und eine textuelle Beschreibung weniger Kompetenzen nur eine geringe Ausdrucksstärke besitzt. Zur fachdidaktischen Diskussion von Lehr-Lernprozessen ist jedoch eine präzise Formulierung der intendierten Kompetenzen unabdingbar. Die Beschränkung auf eine Subdimension und damit ein Fachkonzept ist unzureichend (z. B. *Design (K2.3)*), da dann bei der Visualisierung kognitiver Strukturen viele Knoten auf eine einzige Kompetenzsubdimension verweisen würden. Als Erweiterung zur tabellarischen Darstellung des Modells und den Kompetenzbeschreibungen der Klasse A (vgl. Kapitel 3) werden die Kompetenzen des empirisch verfeinerten Kompetenzstrukturmodells anhand der verwendeten Ankerbeispiele summarisch in Anhang A.1 beschrieben. In Übereinstimmung mit [ACM/IEEE, 2005] ist die zu erreichende übergeordnete Kompetenz künftiger Entwickler eingebetteter Systeme wie folgt definiert:

> „Computer engineers should be able to *design* and *implement* systems that involve the integration of software and hardware devices [emphasis added]“ [ACM/IEEE, 2005].

Diese Kompetenz gilt für die technische Informatik und beschränkt sich nicht auf die Softwareentwicklung, sondern bezieht die Hardware des Systems mit ein. Bei Entwicklern von eingebetteten Systemen sind die Haupttätigkeiten analog – *Design und Implementierung*. Wie in Kapitel 1 und Kapitel 2 gezeigt wurde, erfordert das Design eingebetteter Systeme eine höhere kognitive Stufe als deren Implementierung, was sich jedoch nicht auf die Schwierigkeit bezieht, denn diese hängt vom konkreten Kontext ab.

Aufgrund der Komplexität eingebetteter Systeme und deren Systementwurf ist die Modellierung als Teil des Designs essentiell. Während die Implementierung einer simplen Steuerung auf einem Mikrocontroller ohne eine Modellierung möglich ist (vgl. Kapitel 3), gilt Selbiges nicht für komplexe zuverlässige Systeme mit beispielsweise parallelen Prozessen und Kommunikation über die Systemgrenzen hinweg. In diesem Fall ist eine formale Repräsentation des Systems mit den umgebenden physikalischen Prozessen notwendig [Jaschke, 2013]. Im Hinblick auf künftige Tätigkeiten der Studierenden in der Forschung und im industriellen Umfeld werden Fähigkeiten und Fertigkeiten für ein holistisches Design statt einer ausschließlichen Implementierungskompetenz gefordert, wenngleich Erfahrungen in der Implementierung durchaus Implikationen für das Systemdesign hinsichtlich Machbarkeit und Performanz geben können. Diese in der hardwarenahen Programmierung gewonnenen Erkenntnisse erlauben im Designprozess eingebetteter Systeme die Auswahl bestimmter Lösungsalternativen. Ferner erfolgt bei der Implementierung einfacher

Steuerungen implizit das Design (vgl. Abschnitt 4.1.1).

Zur Ableitung von Empfehlungen zu Lehr-Lernprozessen und Laborpraktika sowie der Entwicklung eines didaktischen Systems für das Design eingebetteter Systeme und die weiterführende didaktische Diskussion wird zunächst die Dimension *(K2.3) Design* ausdifferenziert. Zum einen stellt diese, bezogen auf das Forschungsgebiet, eine zentrale anzustrebende Kompetenzdimension dar, welche über die Implementierung eines Systems hinaus zur ganzheitlichen Entwicklung eingebetteter Systeme befähigen soll. Zum anderen werden, durch die hohe kognitive Anforderung, Relationen zu Kompetenzen auf verschiedenen Stufen erwartet, welche sich daher zur exemplarischen Beschreibung des Vorgehens zur Entwicklung eines didaktischen Systems eignen.

Die in Kapitel 2 analysierten Empfehlungen der ACM, IEEE (vgl. [ACM/IEEE, 2008], [GI, 2011]) und ARTIST (vgl. [Artist Education Group, 2003]) beinhalten zahlreiche Fachkonzepte, beschreiben jedoch die Ziele von Lehrveranstaltungen nur in geringem Umfang beziehungsweise nicht im Sinne der Kompetenzorientierung, sondern aus der Perspektive der Lehrenden und geben keine Hinweise auf mögliche kompetenzfördernde Aktivitäten der Studierenden:

> „A basic outcome of the experimental part of the curriculum for realtime computing should be to show students that realtime systems introduce additional difficulties in system design and programming“ [Caspi et al., 2005, S. 601]

Die Empfehlungen [ACM/IEEE, 2004a] und [ACM/IEEE, 2005] verknüpfen hingegen den Inhaltsbereich mit Lernzielen. Über diese Empfehlungen hinaus dienen die Lehrbücher *Embedded Systems Design* [Marwedel, 2011] und *Introduction to Embedded Systems* [Lee und Seshia, 2012] als Quellmaterial für die Ausdifferenzierung. Beide fokussieren die Einführung in die Entwicklung eingebetteter Systeme für Bachelorstudierende in höheren Semestern. Die Besonderheit dieser Lehrbücher liegt in der expliziten Positionierung und Begründung des Vorgehens im Bildungsprozess (vgl. [Marwedel und Engel, 2011], [Marwedel, 2005], [Lee und Seshia, 2012]).

Die Analyse ergab eine Aufteilung der Subdimension *K2.3 (Design)* in sechs Kategorien [Jaschke, 2013]. Jede Kategorie wird durch ihr Profil beschrieben, wobei das Vorgehen am Beispiel *(K2.3.1) Modelle* in Tabelle 4.1 gezeigt wird. Profil bedeutet in diesem Zusammenhang, dass die im Quellenkorpus identifizierten Beschreibungen von Kompetenzen und Zielen (Ankerbeispiele) zu einer Kompetenzbeschreibung zusammengefasst wurden (vgl. Kapitel 3).

(K2.3.1) Modelle Die Studierenden sollen in der Lage sein, Modellierungstechniken, wie die grafische Modellierung und Beschreibungssprachen, anzuwenden, um Modelle für verschiedene Problembereiche zu erstellen.

Ankerbeispiele	„Model and simulate a digital system using schematic diagrams. Create state and transition diagrams for simple problem domains. Model and simulate a digital system using a hardware description language, such as VHDL or Verilog. Demonstrate the ability to apply the techniques of modeling and simulation to a range of problem areas“ [Jaschke, 2013].
Profil	„Students should be able to apply techniques of modeling like schematic diagrams and description languages to create models for a range of problem areas“ [Jaschke, 2013].
Übersetzung	Studierende sollen in der Lage sein, Modellierungstechniken, wie die grafische Modellierung und Beschreibungssprachen, anzuwenden, um Modelle für verschiedene Problembereiche zu erstellen.

Tabelle 4.1.: Exemplarische Ausdifferenzierung von *(K2.3.1) Modellierung* [Jaschke, 2013]

(K2.3.2) Architekturen Die Studierenden sollen in der Lage sein, die Stärken, Schwächen und die Beziehung verschiedener Architekturen eingebetteter Systeme zu unterscheiden.

(K2.3.3) Technologien Die Studierenden verstehen die Grundsätze eingebetteter Software, eingebetteter Hardware und Kommunikation sowie Schnittstellen zur physikalischen Umgebung.

(K2.3.4) Randbedingungen Die Studierenden sind in der Lage, Randbedingungen eingebetteter Systeme zu bewerten und Zielkonflikte zu analysieren.

(K2.3.5) Zeitverhalten Die Studierenden verstehen Nebenläufigkeiten und Zeitaspekte.

(K2.3.6) Tools Die Studierenden können Hard- und Softwarewerkzeuge anwenden.

Nach der Verfeinerung einer Kompetenzsubdimension des Kompetenzstrukturmodells ist es möglich Fachkonzepte bestimmten Kompetenzen zuzuweisen und in kognitiven Strukturen zu verwenden. Um diese Ergebnisse im Kontext einordnen zu können, werden die ermittelten Kategorien im Folgenden summarisch beschrieben und überprüft, inwiefern diese bereits als Teil des Entwurfs- und Anwendungspraktikums berücksichtigt werden.

K2.3.1 Modelle

Die Begriffe Design und Modellierung werden oftmals synonym verwendet, hier aber differenziert, da Design in anderen Kontexten als zeichnerischer oder gestalterischer Entwurf (Produktdesign, Architektur) betrachtet wird. Dies ist insofern äquivalent zur Modellierung, als dass ein System mit seinen Komponenten und seinem Verhalten mit einer grafischen Modellierungssprache beschrieben werden kann. Modellierung ist also eine von der Realität abstrahierende Beschreibung/Abbildung, in der irrelevante Details entfernt werden. Diese Beschreibung ist eine Teilkomponente des Designs von eingebetteten Systemen, welche die weiteren Kompetenzsubdimensionen – Architekturen, Technologien, Randbedingungen, Zeitverhalten und Tools – verbindet.

Profil: Die Studierenden sollen in der Lage sein Modellierungstechniken, wie die grafische Modellierung und Beschreibungssprachen, anzuwenden, um Modelle für verschiedene Problembereiche zu erstellen.

[Marwedel, 2011] und [Gajski, 1994] vergleichen Modelle und Sprachen für das Design eingebetteter Systeme auf verschiedenen Ebenen des Systementwurfs. Für den Einstieg werden Modellierungen als geeignet angesehen, welche mit wenigen Modellierungselementen das Verhalten eingebetteter Systeme beschreiben.

> „Analog wie beim Übergang von Assembler nach C-Code bietet auch die modellbasierte Entwicklung eine höhere Abstraktionsebene und ausdrucksstärkere Modellierungselemente. Wie bereits erwähnt, werden dadurch die Modelle einfacher und intuitiver, sodass die Komplexität der Modelle im Vergleich zur Komplexität des Quellcodes, der dieselbe Funktionalität realisiert, meist sehr viel geringer ist. [...] Das von außen sichtbare Verhalten der Komponente lässt sich auf verschiedene Arten und Weisen definieren. Beispielsweise können Verhaltensmodelle verwendet werden, oder Sequenzdiagramme können die exemplarische Nutzung aufzeigen“ [Berns et al., 2010].

Wichtig bei der Gestaltung von Lehr-Lernprozessen ist, dass niedere Schichten Teil der Ausbildung bleiben. Dies steht im Widerspruch zu der Erforschung von Ansätzen, in denen Modelle durchgängig in ausführbaren Code beziehungsweise ein Hardwarelayout überführt werden sollen (modellgetriebene Entwicklung). [Lee und Seshia, 2010] heben die Wichtigkeit der niederen Abstraktionsebenen hervor.

> „We have to also fundamentally change the abstractions that are used. Timing properties of software, for example, cannot be effectively introduced at higher levels of abstraction if they are entirely absent from the lower levels of abstraction on which these are built“ [Lee und Seshia, 2010].

Im Entwurfs- und Anwendungspraktikum wird das System von einer Gruppe der Studierenden in VHDL modelliert. Die andere Gruppe implementiert in C. Obwohl

in beiden Fällen eine textuelle Beschreibung vorliegt, handelt es sich um Kompetenzen unterschiedlicher Subdimension und unterschiedlicher kognitiver Stufe, was hier zur Unterscheidung von Modellierung und Implementierung nochmals verdeutlicht werden muss. Ein Modell in einer Hardwarebeschreibungssprache beschreibt eine ausführbare Spezifikation eines Systems. Durch Synthesetools lässt sich diese Spezifikation in ein Hardwarelayout überführen und als ASIC fertigen beziehungsweise in einem FPGA implementieren. Eine Implementierung im eigentlichen Sinne meint die Anwendung einer Programmiersprache zur Umsetzung des Designs.

K2.3.2 Architekturen

Die verwendeten Architekturen im Design eingebetteter Systeme hängen von verschiedenen Faktoren ab, vor allem dem Anwendungszweck in einem zu steuernden, regelnden oder überwachende peripheren technischen Prozess. Die Studierenden müssen dazu zunächst die verschiedenen Einsatzbereiche eingebetteter Systeme mit den wichtigsten Randbedingungen (K2.3.4) kennen und die Auswirkung dieser Randbedingungen verstehen.

Kompetenzen zur Architektur eingebetteter Systeme sind eng verbunden mit Technologien. Bei der Auswahl einer Architektur zur Lösung des Problems werden implizit Technologien (Software-, Hardware- und Vernetzungskonzepte) selektiert.

Profil: Die Studierenden sollen in der Lage sein die Stärken, Schwächen und die Beziehung verschiedener Architekturen eingebetteter Systeme zu unterscheiden.

Dazu ist es notwendig die Eigenschaften von Architekturen und deren Vor- und Nachteile mit den Randbedingungen eingebetteter Systeme zu verknüpfen. Innerhalb der Hochschuldidaktik der technischen Informatik werden meist vier grundlegende Architekturen verwendet – µCs, FPGAs, DSPs und ASICs, wobei die Reihenfolge der Nennung auch für deren Verbreitung steht. Meistens, insbesondere in Bachelorstudiengängen, werden Mikrocontrollerboards verwendet.

Die Auswahl einer zur Lösung eines Problems geeigneten Architektur in der Entwicklung eingebetteter Systeme ist nicht Teil einführender Lehrveranstaltungen. Hier wird die Systemhardware meist vorgegeben und sich auf die Implementierung von Systemkomponenten beschränkt. Hinsichtlich der kognitiven Anforderungen (vgl. Kapitel 2) ist dieses Vorgehen sinnvoll, da die Studierenden sonst überfordert werden könnten. Der ganzheitliche Entwurf von Software und Hardware, zum Beispiel mit einem ASIC, ist Lehrveranstaltungen in höheren Fachsemestern oder im Masterstudium vorbehalten. Für die Zielgruppe im Forschungsprojekt des Autors, Studierende ohne Vorkenntnisse im Bereich eingebetteter Systeme, entschied man sich für die Verwendung von Mikrocontrollern. Zur Steuerung des Modellhauses werden weder besondere Anforderungen an die Rechenleistung noch an den Programmspeicher oder die Anzahl der Schnittstellen gestellt.

K2.3.3 Technologien

Je nach zu verwendender Soft- und Hardwarearchitektur (*K2.3.2*) werden unterschiedliche Technologien, was hier konkrete Software, Hardware, Kommunikation und Schnittstellen eingebetteter Systeme meint, verwendet. Außerdem sind verschiedene Schnittstellen zur physikalischen Umgebung sowie systeminternen und -externen Kommunikation zu verstehen. In einführenden Lehrveranstaltungen eignen sich insbesondere einfache Kommunikationsmechanismen – *Universal Asynchronous Receiver Transmitter (UART)*. Schnittstellen zur physikalischen Umgebung beinhalten neben der Aktorik und Sensorik des Systems auch die Erfassung der physikalischen Wirkprinzipien. Analog/Digital-Wandler sind dabei integraler Bestandteil der einführenden Lehrveranstaltung, was die digitale Verarbeitung analoger, zeitkontinuierlicher Signale der physikalischen Realität ermöglicht.

Profil: Die Studierenden verstehen die Grundsätze eingebetteter Software, eingebetteter Hardware und Kommunikation sowie Schnittstellen zur physikalischen Umgebung.

Um von der Hardwareebene weitestgehend zu abstrahieren, werden Betriebssysteme eingesetzt. Diese übernehmen die Ressourcenverwaltung und eignen sich als Einstieg in die Entwicklung eingebetteter Systeme dadurch, dass mit vorgegebenen Funktionen Basisoperationen, wie das Schalten eines Ausgangs, durchgeführt werden können, ohne dass Kenntnisse über die Systemhardware erforderlich sind. Diese Abstraktion verhindert das Verständnis der zugrundeliegenden Prozesse und eignet sich daher entweder für weiterführende Lehr-Lernprozesse, in denen Echtzeitbetriebssysteme, Mehrkernprozessoren etc. zur Lösung komplexerer Probleme verwendet werden, oder aber zur Motivation für das Gebiet, indem spielerisch durch einfache Befehle das Verhalten des Systems beeinflusst werden kann.

K2.3.4 Randbedingungen

Gilt es eine Komponente für ein konkretes System auszuwählen oder zu verwenden, sind die Randbedingungen des Systems zu verstehen, was den technischen Prozess der Umgebung mit einschließt. Typische Randbedingungen sind hier Leistungsaufnahme oder zulässige Umgebungstemperaturen.

Profil: Die Studierenden sind in der Lage, Randbedingungen eingebetteter Systeme zu bewerten und Zielkonflikte zu analysieren.

Die funktionalen Anforderungen an eingebettete Systeme beschreiben das Verhalten des Systems in der Umgebung – die konkrete Lösung des Anwendungsproblems. Dieses Verhalten wird im Entwurfs- und Anwendungspraktikum in der Aufgabenstellung beschrieben und von den Studierenden durch die Programmierung unter Verwendung vorgegebener Bibliotheken erfüllt. Nicht-funktionale Anforderungen spielen im Praktikum eine untergeordnete Rolle. Dies spiegelt nicht die industrielle Realität wieder, in der bei allen Designentscheidungen nicht-funktionale Anforde-

rungen, beispielsweise ökonomische oder fertigungstechnische Bedingungen, einen sehr hohen Stellenwert haben. In Anbetracht der Verortung von Praktika im Curriculum von Bachelorstudiengängen ist die Unterrepräsentation von Randbedingungen sicherlich angemessen. Diese komplexen Zusammenhänge, welche zudem aus industriellen Bedingungen resultieren, lassen sich nicht effektiv in einer Laborumgebung abbilden. Durch die Begrenzung der Stromaufnahme in der exemplarischen Versuchsaufgabe 5 (siehe Kapitel 4) wird den Studierenden gezeigt, dass nicht-funktionale Anforderungen direkte Auswirkungen auf die Implementierung haben und gleichrangig zu funktionalen Anforderungen sind.

Randbedingungen werden einerseits durch den peripheren technischen Prozess bestimmt, andererseits durch Designentscheidungen, welche die Studierenden in diesem Fall nicht mehr beeinflussen können (Auswahl des Mikrocontrollerboards). Die Auflösung des integrierten Analog/Digital-Wandlers bestimmt die Genauigkeit der Regelungen. Ebenso beeinflussen die zur Verfügung stehenden *Timer* das Zeitverhalten. Für die Aufgabenstellung zum Lerngegenstand im Entwurfs- und Anwendungspraktikum, einer Hausautomation, sind diese Randbedingungen jedoch nur theoretisch vorhanden, da beispielsweise der Messfehler bei der Temperaturermittlung <0,1 Grad Celsius beträgt und damit keine einschränkende Eigenschaft ist. Dennoch sollten Randbedingungen, wenn möglich, Teil von Aufgabenstellungen sein, damit die Studierenden auch verstehen, dass bei der Wahl von Bauteilen oder der Verwendung von Funktionen entsprechende Passagen der Datenblätter geprüft, analysiert und im Design berücksichtigt werden müssen.

K2.3.5 Zeitverhalten

Das Zeitverhalten eingebetteter Systeme wurde bereits als wesentliches Merkmal für die besonderen Anforderungen benannt. Weiche und harte Echtzeitanforderungen resultieren aus dem Einsatzgebiet in oftmals sicherheitskritischen Anwendungen (z.B. Airbag). Parallele Prozesse und eine kontinuierliche physikalische Realität führen zu einer hohen Komplexität bei der Betrachtung des Systemverhaltens bezogen auf eine diskrete Zeit im System.

Profil: Die Studierenden verstehen Nebenläufigkeiten und Zeitaspekte.

Reaktive eingebettete Systeme müssen auf das Auftreten externer Signale reagieren, ohne Echtzeitanforderungen zu verletzen. Im Entwurfs- und Anwendungspraktikum wird das Zeitverhalten eingebetteter Systeme verdeutlicht, indem die Studierenden in den ersten Versuchen lediglich physikalische Größen erfassen und ein Ergebnis ausgeben. Hier spielt die Zeit, abgesehen von der Generierung einer Pulsweitenmodulation, eine untergeordnete Rolle. Später soll das Verhalten des Systems durch das Betätigen von Tastern beeinflusst werden. Nun wird deutlich, dass zyklisches Abfragen von Eingängen dazu führt, dass Signale nicht erkannt werden, wenn zum Zeitpunkt des Tastendrucks das Programm in einer Warteschleife verharrt.

K2.3.6 Tools

Das Anwenden von Entwicklungswerkzeugen ist essentiell im Design eingebetteter Systeme, da die hohe Systemkomplexität ohne entsprechende Werkzeuge nicht zu bewältigen wäre. Eine Motivation zur Umgestaltung des Entwurfs- und Anwendungspraktikums war die Schwierigkeit der Anwendung professioneller Entwurfssoftware (vgl. Kapitel 3).

Profil: Die Studierenden können Hard- und Softwarewerkzeuge anwenden.

Nun beschränkt man sich auf die den Studierenden bereits bekannte Entwicklungsumgebung *Eclipse*, was einen einfacheren Zugang ermöglicht. Für die Lehre zu eingebetteten Systemen im Allgemeinen sind insbesondere in den ersten Semestern einfache oder bestenfalls speziell für die Lehre entwickelte Tools zu verwenden. Nach einer Spezialisierung im Masterstudium können dann entsprechende professionelle Tools eingesetzt werden, welche dann auch in der beruflichen Praxis Anwendung finden.

Implizites Mikrosystemverständnis von Studienanfängern

In der formativen Evaluation des Entwurfs- und Anwendungspraktikums wurde festgestellt, dass die Studierenden allenfalls geringfügige Kenntnisse zu physikalischen Prinzipien und Elektrotechnik aus dem Physikunterricht an der Sekundarstufe II besitzen. Gleiches gilt auch für Technologien und Konzepte eingebetteter Systeme, obwohl die Studierenden hier bereits Informatik im vierten Fachsemester studieren. In der Projektskizze von KOMINA war die Erforschung des impliziten Mikrosystemverständnisses von Schülern und Erstsemesterstudierenden ein Ziel. Hier sollte erforscht werden, inwiefern Schüler respektive Studienanfänger bereits über ein implizites Verständnis von eingebetteten Systemen verfügen. Dass die Systeme in vielen Bereichen dem Anwender verborgen bleiben, gilt beispielsweise nicht für Smartphones aus der Lebenswelt der Studierenden oder Robotik, welche unter Umständen bereits Teil des Informatikunterrichts in der Sekundarstufe II an allgemeinbildenden Schulen sind. Aufgrund der Verkürzung des Projektzeitraums blieb dieser Teil unerforscht. Um für künftige Arbeiten eine Datenbasis bieten zu können, wurden im Rahmen von KOMINA zwei Befragungen von Erstsemesterstudierenden der Informatik zur Identifizierung von Vorkenntnissen durchgeführt, bislang aber nicht ausgewertet. Die Analyse dieser Datenbasis soll im Rahmen dieser Arbeit einen weiteren Einblick in die Struktur der Kompetenzen der Studierenden bieten, um festzustellen, mit welchen Vorkenntnissen die Schüler ihr universitäres Studium beginnen und welche Erfahrungen sie bereits mit Mikrocontrollersteuerungen – analog zum Entwurfs- und Anwendungspraktikum – gemacht haben. Auch wenn davon ausgegangen werden muss, dass Arbeiten in der Sekundarstufe II verglichen mit der universitären Lehre eher rudimentär sind und die Implementierung oder grafische Programmierung einfacher Steuerungen nur wenig mit dem ganzheitlichen Entwurf eingebetteter Systeme gemein haben, er-

lauben entsprechende Vorkenntnisse eine schnelle Orientierung im Fach und den entsprechenden Termini.

Im Jahre 2011 erfolgten zwei Befragungen an der Universitäten Siegen und Erlangen-Nürnberg (siehe Anhang A.6 und A.7). An der Universität Siegen wurden 30 Studierende, an der Universität Erlangen-Nürnberg 67 Studierende befragt. Die Auswertung dieser Befragung stützt die Erkenntnisse aus der formativen Evaluation des Entwurfs- und Anwendungspraktikums insofern, als das festgestellt wurde, dass mit sehr geringen und heterogenen Vorkenntnissen zu rechnen ist. Dies soll keinerlei Wertung des Bildungsniveaus darstellen, sondern vielmehr aufzeigen, dass eingebettete Systeme trotz zahlreicher fachdidaktischer Publikationen und deren Relevanz noch nicht durchgängig in allgemeinbildenden Schulen eingesetzt werden. Ferner zeigt die Befragung, dass viele Studierende ihre Entscheidung für das Studienfach Informatik nicht aus den Vorerfahrungen in der Schule treffen. Keiner der Befragten belegte einen Leistungskurs in Informatik. Ein Anteil von ca. 30 % der Befragten besitzt keine schulischen Vorkenntnisse in der Informatik. Lego Mindstorms als Beispiel für eingebettete Systeme kennen ca. 42 % der Erstsemesterstudierenden. Mikrocontroller sind hingegen nur ca. 18 % der Studierenden bekannt, wobei drei Studierende diese nutzen und besitzen. FPGAs sind ca. 84 % der Studierenden unbekannt, was bedeutet, dass Mikrocontroller den Novizen in der Informatik eher bekannt sind und sich damit gegebenenfalls auch eher als Einstieg in die Entwicklung eingebetteter Systeme eignen, obgleich auch hier die Zahl sehr gering ist.

Die wenigsten Studierenden besitzen Kenntnisse über hardwarenahe Programmiersprachen, Mikrocontroller, FPGAs oder Roboter. Außerdem gaben weniger als 50 % der Studierenden an, eingebettete Systeme zu kennen.

Bei den Studierenden, welche angaben eingebettete Systeme zu kennen, wurde dies durch widersprüchliche Antworten teilweise widerlegt. Die Studierenden benannten auch eine Dreifachsteckdose als eingebettetes System, was offensichtlich nicht korrekt ist und den Schluss zulässt, dass diese Studierenden keine Vorstellung von eingebetteten Systemen besitzen. Die Ergebnisse dieser Befragungen sind insofern überraschend, als dass die an KOMINA beteiligten Forscher annahmen, dass Studierende der Informatik eine gewisse Orientierung im Fach besitzen und zumindest den Terminus eingebettetes System und Beispiele für den Einsatz dieser Systeme kennen. Dies ist jedoch nicht der Fall, sodass man bei der Gestaltung von Lehr-Lernprozessen im Bachelorstudiengang Informatik von keinerlei Vorwissen auszugehen hat. Bezogen auf die informatikdidaktische Verfeinerung des Kompetenzstrukturmodells heißt das, dass alle Kompetenzen als Voraussetzung nicht implizit als vorhanden angenommen werden dürfen, sondern Teil bei der Gestaltung von Lehr-Lernprozessen sein müssen. Das gilt auch für Kompetenzen, welche fachfremd sind oder zunächst vernachlässigbar scheinen (z. B. Englisch).

4.3. Ableitung kognitiver Strukturen zu identifizierten Lernhürden

Die Ableitung einer kognitiven Struktur für alle Fachkonzepte, wie auch die Ausdifferenzierung aller Subdimensionen des Kompetenzstrukturmodells, ist nicht innerhalb eines Forschungsprojektes durchführbar. Um zu einer Empfehlung für die Gestaltung von Lehr-Lernprozessen zu gelangen, ist eine kognitive Struktur je zu erlangender Kompetenzfacette abzuleiten. Durch eine Verknüpfung der Strukturen wird ersichtlich, welche Kompetenzen zur Lösung eines Problems vorhanden sein müssen. Die sich ergebende Gliederung kann dabei weder vollständig noch frei von Redundanzen sein, da die Art der Relation von der kognitiven Stufe der zugehörigen Kompetenz und der Intention des Lehrenden in diesem konkreten Lehr-Lernprozess abhängt. Ferner ist zu berücksichtigen, welche Voraussetzungen Studierende besitzen. In Kapitel 3 wurden die identifizierten Lernhürden zusammengefasst. Eine Empfehlung zur Weiterentwicklung des Entwurfs- und Anwendungspraktikums kann gegeben werden, wenn zu jeder Hürde (LH_{1-4}) eine kognitive Struktur abgeleitet wird, welche die zur Hürde gehörenden Fachkonzepte beziehungsweise Kompetenzen als Knoten beinhaltet. An diese mögliche Erarbeitungsreihenfolge kann dann die Struktur des Praktikums angepasst werden. Um Redundanzen zu vermeiden, werden Knoten, die in mehreren kognitiven Strukturen relevant sind, nur einmal beschrieben.

4.3.1. Kognitive Struktur zu $\mathbf{LH_1}$ – *Dokumentationen und Datenblätter*

Eine Lernhürde wurde bei der Analyse von Dokumentationen – Datenblätter, Anleitungen, Quellcode – identifiziert, was sich auf verschiedene Aufgaben auswirkt. Wissen über die Terminologie und über das Fachgebiet ist Voraussetzung, um in Lehr-Lernprozessen sinnvoll zu agieren. Ohne dies sind weder die Aufgabe selbst noch die Datenblätter der zur Entwicklung notwendigen Hard- und Softwarekomponenten zu verstehen. Die Kompetenzen *(K2.3.5) Zeitverhalten* und *(K2.3.3) Technologien* sind Beispiele für Kompetenzen, deren zugehörige Fachkonzepte in Datenblättern beschrieben werden (siehe Abbildung 4.7). Um Designentscheidungen zu treffen, ist das Verstehen einzelner Aspekte eines Datenblattes nicht ausreichend. Die Studierenden müssen dort gegebenenfalls aufgeführte Realisierungsalternativen vergleichen können. Ebenso sind Datenblätter unterschiedlicher Bauteile zu analysieren, um das für den Anwendungsfall geeignetste auszuwählen. Im Entwurfs- und Anwendungspraktikum wurde erst im zweiten Versuch die Kompetenz zur Analyse von Datenblättern hervorgehoben (siehe Abbildung 3.11). Im ersten Versuch (siehe Abbildung 3.10) wurden allerdings das Oszilloskop, der Funktionsgenerator und das im Labor verwendete Steckbrett auf der Ebene *A2* angestrebt.

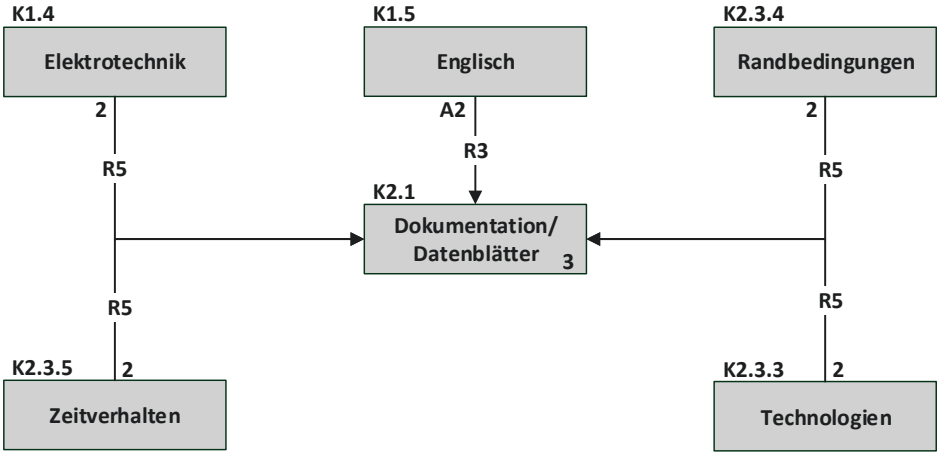


Abbildung 4.7.: Kognitive Struktur – Dokumentation

Englisch

Im Common European Framework of Reference for Languages (CEFR) werden sechs Stufen der Sprachkompetenz unterschieden (A1, A2, B1, B2, C1, C2). In diesem Beispiel zur Lernhürde LH_1 steht das Leseverstehen bei der Analyse von Datenblättern im Vordergrund. In anderen Kontexten, z. B. mit Relationen zu nicht-kognitiven Kompetenzen, in denen die Kommunikation im Team fokussiert werden, sind Hörverstehen und Sprechen von Bedeutung. Bei dem Verfassen von Dokumentationen, was den Quellcode bei der Programmierung einschließt, ist das Schreiben in englischer Sprache relevant. Für die Analyse von Datenblättern ist ein Leseverstehen auf Stufe B1 bis B2 erforderlich.

> „B1: Can understand the main points of clear standard input on familiar matters regularly encountered in work, school, leisure, etc.
>
> B2: Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialization [Council of Europe, 2001, S. 24]“.

Ausgehend von den Erwartungen an Studierende mit allgemeiner Hochschulreife kann mit einem Leseverständnis im Bereich B2 bis C1 gerechnet werden (vgl. [KMK, 2012]). Studierende können demnach die Hauptpunkte technischer Diskussionen in ihrer Disziplin (Informatik) verstehen. Als komplexe Texte werden solche mit überdurchschnittlich vielen Fachbegriffen aufgefasst, was offensichtlich für Datenblätter gilt. Abgesehen von diesen Fachbegriffen wird einfache Sprache verwendet (siehe Abbildung 4.8).

Technologien

Hier kann auf die Ausdifferenzierung der Subdimension *Design (K.2.3)* zurückgegriffen werden, da Fachkonzepte innerhalb der Dokumentation die Fachkonzepte eingebetteter Systeme betreffen. Diese sind beispielsweise mit der Kompetenz *Technologien (K2.3.3)* verknüpft. Um die Zusammenhänge verschiedener Komponenten im System anhand der Datenblätter zu analysieren, ist ein Verstehen der von der Technologie abhängigen Termini, Diagramme und Wirkungsweisen notwendig. In dieser kognitiven Struktur meint Technologie beispielsweise die Pulsweitenmodulation. Begriffe wie *Duty-Cycle* müssen verstanden werden, um die in den Aufgabenstellungen definierten Signale mithilfe der Abbildungen zur Pulsweitenmodulation (siehe Abbildung 4.13) zu erzeugen.

Zusammenfassend besteht also das Problem darin, dass von den Studierenden die Anwendung der in den Datenblättern beschriebenen Technologie nach Analyse der dortigen Beschreibung erwartet wird. Eine Erklärung der Funktionsweise ist aber nicht die eigentliche Intention der Dokumentation. Diese umfasst lediglich die Eigenschaften des Bauteils und dessen Einsatz in verschiedenen Szenarien. Konkret bedeutet dies, dass zunächst die Technologie in Form didaktisch aufbereiteter Materialien eingeführt werden muss und erst dann zu Aufgaben mit Datenblättern übergegangen werden kann.

Zeitverhalten

Das Zeitverhalten eingebetteter Systeme hängt von den Komponenten des Systems und der Implementierung ab. So ist beispielsweise die Frequenz einer Analog/Digital-Wandlung an das gegebene Signal anzupassen. Die zu verwendenden Parameter einer Pulsweitenmodulation werden durch die anzusteuernde Aktorik – beispielsweise der LED – bestimmt. Bevor die Studierenden in der Lage sind die für die Pulsweitenmodulation notwendigen Registerbelegungen durch eine Analyse des Datenblattes des Mikrocontrollers zu bestimmen, muss zunächst verstanden werden, welche Auswirkungen verschiedene Ansätze zur Erzeugung der Pulsweitenmodulation haben. Die Studierenden nutzen in der Implementierungsphase zunächst eine Softwarepulsweitenmodulation, das heißt sie schalten in einer Schleife einen Ausgang ein, warten wenige Millisekunden und schalten diesen wieder aus. Zur Lösung der aktuellen Aufgabenstellung ist dieses Vorgehen richtig. Die Studierenden wissen zu diesem Zeitpunkt nicht, dass der Mikrocontroller eine Pulsweitenmodulation in Hardware generieren kann. Dies hat wesentliche Auswirkungen auf das Zeitverhalten der Steuerung. Sobald die Studierenden zusätzlich Tastendrücke auswerten müssen fällt auf, dass diese nicht erkannt werden, solange sich das Programm des Mikrocontrollers in einer Warteschleife befindet. Bevor die entsprechenden Aussagen in Datenblättern verstanden werden können, sind die Behandlung von Ereignissen und die Nutzung von *Timern* zur Betrachtung möglicher Lösungsalternativen zu erläutern, da den Studierenden die Auswirkung auf das Zeitverhalten des Gesamtsystems ansonsten nicht bewusst ist. Die Behand-

lung von Ereignissen und die Nutzung von *Timern* zur Betrachtung möglicher Lösungsalternativen haben erst zu erfolgen, bevor entsprechende Passagen in Datenblättern verstanden werden können.

Randbedingungen

Gilt es eine Komponente für ein konkretes System auszuwählen sind die Randbedingungen des Systems zu verstehen, was den technischen Prozess der Umgebung mit einschließt. Typische Randbedingungen sind Leistungsaufnahme oder zulässige Umgebungstemperaturen. Im Entwurfs- und Anwendungspraktikum sollen zwei Gruppen von LEDs angesteuert werden ohne eine maximale Leistungsaufnahme zu übersteigen. Hier muss die Pulsweitenmodulation verstanden werden, um die in der Aufgabe geforderten Randbedingungen einzuhalten, beispielsweise eine dimmbare LED, welche aufgrund der Trägheit des Auges nicht als Blinkmuster wahrgenommen wird.

Elektrotechnik

Komponenten eingebetteter Systeme, Sensoren, Aktoren etc. besitzen eine bestimmte elektrische Charakteristik, welche dem Datenblatt entnommen werden kann. Dazu müssen die Studierenden die angegebenen Schaltungen und Diagramme verstehen, was auch elektrische Größen und deren Bedeutung umfasst. Bereits in der physikalisch-technischen Einführung des Entwurfs- und Anwendungspraktikums sollen die Studierenden einen Spannungsteiler dimensionieren. Mithilfe eines temperaturabhängigen Widerstandes (vgl. [Philips Semiconductors, 2000]) soll so die Temperatur im Raum ermittelt werden. Dazu müssen die Studierenden Grundschaltungen kennen und berechnen können (Reihenschaltung) sowie Basisgrößen beziehungsweise deren Zusammenhang verstehen (Strom, Spannung).

Fazit

Dokumentationen und Datenblätter fassen das zu Bauteilen gehörende Wissen zusammen und beinhalten teilweise Anwendungsbeispiele, aus denen Lösungsalternativen resultieren. Bevor die Dokumentationen analysiert werden können, beispielsweise um ein Bauteil auszuwählen, müssen die darin enthaltenen Fachkonzepte verstanden werden (Relation R5). Es wurde gezeigt, dass diese Fachkonzepte verschiedene Kompetenzsubdimensionen betreffen. Die englische Sprache muss dazu angewandt werden, was in diesem Kontext ein Sprachniveau auf Stufe B2 meint.

4.3.2. Kognitive Struktur zu LH_2 – *Vorgehensweise*

Die zweite Lernhürde wurde in der Vorgehensweise der Studierenden identifiziert. Ohne die Fähigkeit Entwicklungsprozesse analysieren zu können, also in Abhängigkeit von einer Problemstellung das geeignete Vorgehen auszuwählen, ist es nicht möglich mit akzeptablem Zeitaufwand die Aufgaben zu bewältigen, da wahrscheinlich eine Versuch-Irrtum-Strategie angewandt wird. *(K2.3.2) Architekturen* und

CHARACTERISTICS

T_{amb} = 25 °C, in liquid, unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
R_{25}	sensor resistance	I_{cont} = 1 mA				
	KTY81-210		1980	–	2020	Ω
	KTY81-220		1960	–	2040	Ω
	KTY81-221		1960	–	2000	Ω
	KTY81-222		2000	–	2040	Ω
	KTY81-250		1900	–	2100	Ω
	KTY81-251		1900	–	2000	Ω
	KTY81-252		2000	–	2100	Ω
TC	temperature coefficient		–	0.79	–	%/K
R_{100}/R_{25}	resistance ratio	T_{amb} = 100 °C and 25 °C	1.676	1.696	1.716	
R_{-55}/R_{25}	resistance ratio	T_{amb} = –55 °C and 25 °C	0.480	0.490	0.500	
τ	thermal time constant; note 1	in still air	–	30	–	s
		in still liquid; note 2	–	5	–	s
		in flowing liquid; note 2	–	3	–	s
	rated temperature range		–55	–	+150	°C

Notes

1. The thermal time constant is the time taken for the sensor to reach 63.2% of the total temperature difference. For example, if a sensor with a temperature of 25 °C is moved to an environment with an ambient temperature of 100 °C, the time for the sensor to reach a temperature of 72.4 °C is the thermal time constant.

Abbildung 4.8.: Datenblattauszug KTY81-220 [Philips Semiconductors, 2000, S. 3]

(K2.3.3) Technologien stehen in Abbildung 4.9 stellvertretend für deklaratives Vorwissen zur Analyse von Lösungsansätzen.

Lösungsansätze, Architekturen und Technologien

Verschiedene Lösungsansätze – in unterschiedlichen Kontexten – dienen im Entwicklungsprozess als Erfahrungswerte. Dazu ist ein Verstehen verschiedener Architekturen und Technologien als Lösungsvarianten notwendig. Die Zielgruppe hat bislang keine Erfahrung in der Entwicklung eingebetteter Systeme, sodass auch kein Vorrat an möglichen Lösungsansätzen vorhanden ist. Wie bereits erläutert, kann eine Pulsweitenmodulation in Hardware oder in Software (Schleife) generiert werden, was den Studierenden nicht bekannt ist. Das heißt, dass verschiedene Ansätze strukturiert eingeführt werden müssen, was die jeweiligen Vor- und Nachteile einschließt. Der Lerngegenstand, die Sensoren, Aktoren und das Mikrocontrollerboard sind bereits gegeben, sodass der Eindruck entsteht, dass Kenntnisse über Designentscheidungen der Lehrenden keine Relevanz haben. Vielmehr dienen diese Designentscheidungen aber der didaktischen Reduktion vom ganzheitlichen Entwurf hin zu einzelnen Aspekten hardwarenaher Programmierung. Kenntnisse über diese Entscheidungen und damit die Auswahl von Teillösungen können den Ent-

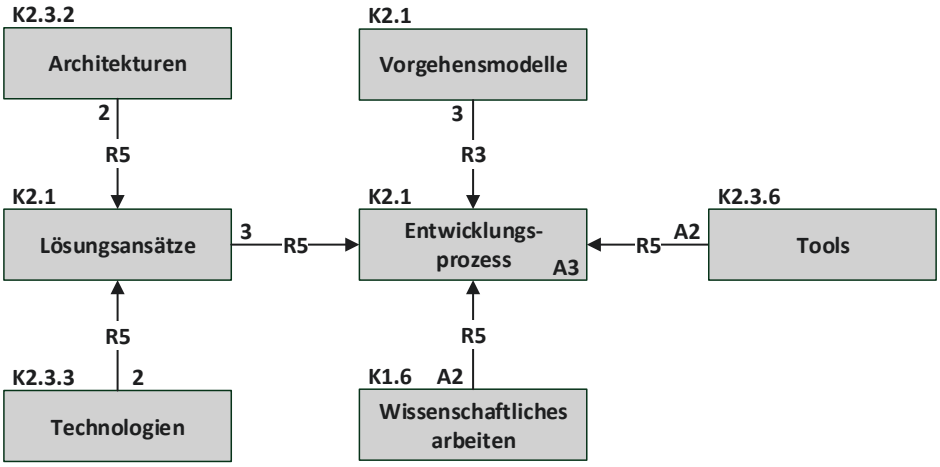


Abbildung 4.9.: Kognitive Struktur – Vorgehensweise

wicklungsprozess der Studierenden unterstützen. Dazu ist es erforderlich, dass Architekturen und Technologien insofern verstanden werden, als dass die Eigenschaften zu bestimmten Lösungsansätzen führen und im konkreten Entwicklungsprozess dann analysiert werden müssen.

Vorgehensmodelle

Das Vorgehen bei der Entwicklung ist, abgesehen von einer Grobstrukturierung in Vorgehensmodellen, abhängig von den Erfahrungen der Entwickler mit Lösungsansätzen in anderen Kontexten. Die Studierenden haben erste Erfahrungen in der Softwareentwicklung in einführenden Modulen sammeln können. Dort werden auch verschiedene Vorgehensmodelle vorgestellt, welche sich nur bedingt zur Entwicklung eingebetteter Systeme eignen (vgl. Kapitel 2). Im Entwurfs- und Anwendungspraktikum steht jedoch nicht der ganzheitliche Entwurf, sondern die hardwarenahe Programmierung im Vordergrund. Damit Studierende, die erstmals in einem solchen Entwicklungsprozess agieren, sich nicht in der Fülle möglicher Vorgehensweisen verlieren und insbesondere auf eine Versuch-Irrtum basierende Strategie verzichten, ist das Vorgehen über die verschiedenen Versuchsphasen hinweg zu diskutieren und zu visualisieren, was Kompetenzen aus dem wissenschaftlichen Arbeiten tangiert.

Wissenschaftliches Arbeiten

In Zusammenhang mit Vorgehensmodellen für das Design eingebetteter Systeme muss zur Aufgabenbewältigung wissenschaftlich gearbeitet werden. Jeder Schritt im Entwicklungsprozess ist zu dokumentieren. Ferner ist bei Aufgaben zur Pro-

grammierung zunächst eine Hypothese in der Vorbereitungsphase zu bilden, welche dann im Labor überprüft wird. Hierunter wird auch verstanden, dass vor jeder Laboreinheit diese Hypothese, also die vorgeschlagenen Handlungen, vorgestellt und diskutiert werden.

Tools

Tools sind auf allen Ebenen der Entwicklung anzuwenden, um eingebettete Systeme mit Computerunterstützung zu entwickeln. Durch die Funktionsvielfalt professioneller Software ist eine Orientierung bei der Navigation und Nutzung der verschiedenen Eingabe- und Konfigurationsmasken für Studierende schwierig. Im Entwurfs- und Anwendungspraktikum müssen die Studierenden verschiedene Hard- und Softwaretools anwenden. In der physikalisch-technischen Einführung stellt die Anwendung des Oszilloskopes eine große Hürde dar. Im weiteren Verlauf ist eine Reihe von Werkzeugen (Toolkette) anzuwenden, welche vollständig in *Eclipse* integriert ist (vgl. Kapitel 3).

Fazit

Bevor die Studierenden versuchen die Hausautomation mittels Anwenden einer Versuch-Irrtum-Strategie zu entwickeln, müssen die zu möglichen Lösungsansätzen gehörigen Architekturen, Technologien etc. verstanden werden. Ferner sind die erforderlichen Tools einzuführen. Damit die Studierenden sich im noch unbekannten Entwicklungsprozess orientieren können und strukturiert vorgehen, müssen die dafür wesentlichen Kompetenzen im wissenschaftlichen Arbeiten angewandt und verschiedene Vorgehensmodelle analysiert werden können. Das schließt die Analyse von Lösungsansätzen ein.

4.3.3. Kognitive Struktur zu LH_3 – *Laborausstattung*

Bei der Anwendung der Laborausstattung ist nicht klar zu differenzieren, ob mangelnde Kenntnisse zu Grundlagen der Elektrotechnik oder die Bedienung der Geräte die Barriere darstellt. Wie mit der kognitiven Struktur zu LH_3 gezeigt wurde, sind die Kompetenzen zu *(K1.4) Elektrotechnik* und *(K1.6) Wissenschaftliches Arbeiten* zur Analyse von Datenblättern eng mit den Kompetenzen *(K2.3) Design* verknüpft und können hier als Vorbedingung identifiziert werden.

Messtechnik und elektrische Größen

Neben einem Verständnis für die zu messenden elektrischen Größen sind Grundlagen der Messtechnik notwendig. Den Studierenden im Entwurfs- und Anwendungspraktikum war nicht klar, dass Spannung parallel zum Widerstand und Strom in Reihe gemessen wird. Hier ist es wichtig zu verstehen, wie verschiedene elektrische Größen – Strom, Spannung, Feld – wirken, um auch die Messung dieser mittels Oszilloskop und Multimeter durchführen sowie zusätzliche Messschaltungen anwenden zu können.

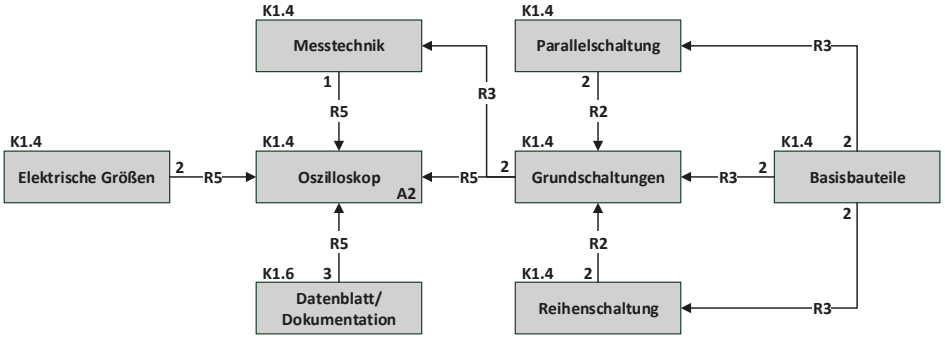


Abbildung 4.10.: Kognitive Struktur – Laborausstattung

Grundschaltungen

Das Verstehen der verschiedenen Grundschaltungen der Elektrotechnik, also gemischte Schaltungen aus Basisbauteilen, wie Widerstände und Kondensatoren, muss, wie auch deren Funktionsweise, verstanden werden, um die gemessenen Werte interpretieren zu können. Als wesentlich für die Einführung zu eingebetteten Systemen und ihrer Sensorik werden die Reihen- und Parallelschaltung von Widerständen sowie Hoch- und Tiefpassfilter gesehen. Problematisch ist, dass trotz Verstehen der Grundschaltungen noch immer eine große Hürde in der Anwendung des Oszilloskopes zu überwinden ist, da dessen Funktionsvielfalt die Studierenden im Labor überfordert (siehe Abbildung 4.11).

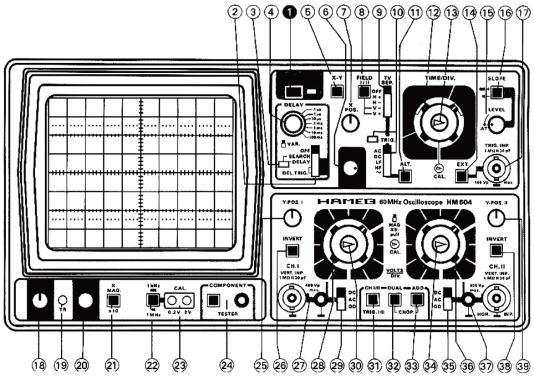


Abbildung 4.11.: Oszilloskop – Überforderung durch Funktionsvielfalt [Schäfer et al., 2012c]

Fazit

Die Relevanz von Kompetenzen im wissenschaftlichen Arbeiten im Sinne des Kompetenzstrukturmodells (siehe Anhang A.1), was die Analyse der Dokumentationen einschließt, wurde bereits in der kognitiven Stufe zu LH_1 beschrieben. Ferner wird hier aber deutlich, dass die Elektrotechnik, welche eher als Mittel zum Zweck in der technischen Informatik erscheint, nicht als gegeben angenommen werden kann und strukturiert im Rahmen des Praktikums eingeführt werden muss. Diese kognitive Struktur zeigt eine mögliche Erarbeitungsreihenfolge und belegt gleichzeitig, dass die derzeitige Ausdifferenzierung nicht ausreicht.

4.3.4. Kognitive Struktur zu LH_4 – *Register und Bitoperationen*

Hardwarenahe Programmierung mit C hat wenig gemein mit der Programmierung von Anwendungssoftware. Zwar sind die Sprachkonstrukte (Schleifen, Bedingungen etc.) dieselben, dennoch sind Bitoperationen und Registerzuweisungen zumeist unbekannt und müssen strukturiert eingeführt werden. Register werden im Entwurfs- und Anwendungspraktikum auf der algorithmischen Ebene der Abstraktionsdimension betrachtet. Die Programmierung des Mikrocontrollers erfolgt anhand der Parametrisierung der Register. Die Eigenschaften und Funktionsweisen von Mikrocontrollern müssen verstanden werden, um zu der Erkenntnis zu gelangen, dass der konkrete Einsatz im technischen Prozess von der Programmierung abhängt. Der Mikrocontroller soll daher auf der Abstraktionsstufe *System* betrachtet werden. Analog zu den vorherigen Wissensstrukturen ist die Anwendung der englischen Fachsprache notwendig, um Datenblätter einer Komponente zu analysieren. Diese Fachsprache muss alle zur Steuerung der Funktion notwendigen Termini der Fachkonzepte umfassen und außerdem eine Navigation im Dokument selbst erlauben. Dazu ist neben der Fachsprache also auch ein Basiswortschatz notwendig.

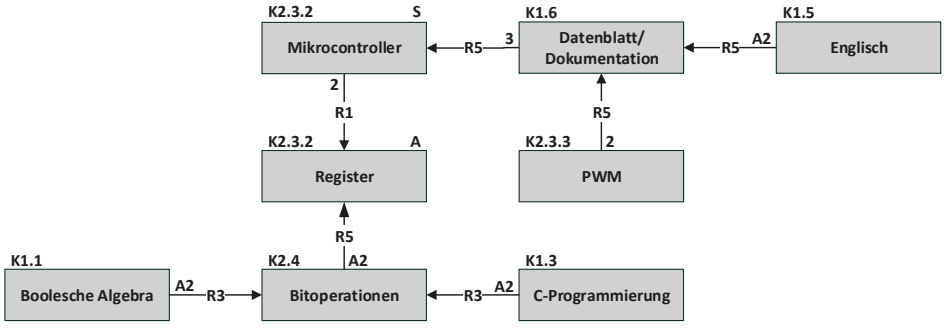


Abbildung 4.12.: Kognitive Struktur – Register und Bitoperationen

Mikrocontroller

Die Dokumentation (die Datenblätter) elektrotechnischer Komponenten ist in der Regel sehr umfangreich. Bei Mikrocontrollern umfassen diese nicht nur die Funktionen, sondern auch technische Grundlagen wie *Pull-Up-Widerstände* sowie die elektrischen Spezifikationen und Beispielschaltungen. Um konkrete Architekturen und Komponenten, hier Mikrocontroller (ATxmega128A1), verstehen zu können ist eine Analyse des Datenblattes erforderlich.

Pulsweitenmodulation, Datenblatt/Dokumentation und Englisch

Pulsweitenmodulation steht hier stellvertretend für eine in Hardware realisierte Funktion, welche mit entsprechenden Registern parametrisiert wird. Ein Verständnis derer Eigenschaften ist erforderlich, um die Beschreibung konkreter Realisierungen in der Dokumentation zu analysieren. Dazu ist, wie bereits erläutert, auch die Anwendung der englischen Sprache notwendig. In Abbildung 4.13 kann das Verhalten des Ausgangs bei entsprechender Registerbelegung analysiert werden.

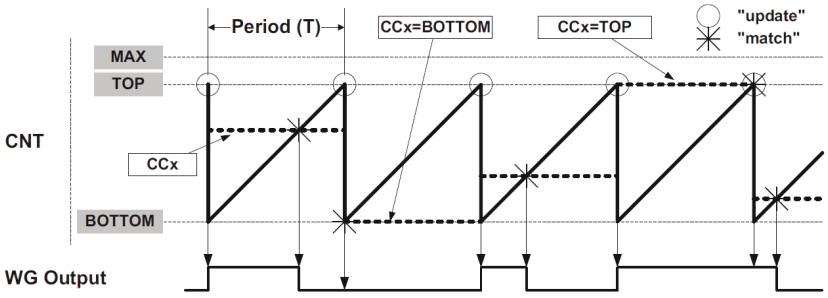


Abbildung 4.13.: Single-slope PWM [Atmel Corporation, 2013, S. 160]

Register

Das Setzen und Löschen einzelner Bits in Registern bestimmt das Verhalten des Mikrocontrollers. Die zur Funktion zugehörige Belegung ist in den Datenblättern zu analysieren. Dazu ist auch ein Verstehen der Architektur und Registervarianten (8-, 16-, 24-, und 32-Bit) sowie das Anwenden von Bitoperationen wichtig.

Bitoperationen und Boolesche Algebra

Bitoperatoren sind notwendig, um den Status von Bits abzufragen, zu setzen oder zu manipulieren, z. B. schalten einzelner *Pull-Up-Widerstände* des Mikrocontrollers. Dazu existieren Operatoren, welche mehrere Boolesche Operatoren (UND, ODER, OR, XOR, NOT und XOR) verknüpfen. Diese müssen nachvollzogen werden, um fremden Code zu analysieren sowie selbst zu erstellen. Dazu ist das Anwen-

den der C-Programmierung in Verbindung mit Booleschen Operatoren erforderlich (siehe Listing 4.1). Boolesche Algebra sollte den Studierenden aus Grundlagenveranstaltungen bekannt sein.

```
1  //Bit(s) setzen
2  uint8_t a = 0x08; /* 00001000 */
3  a |= (1<<2);      /* 00001100 */
4
5  uint8_t a = 0x08;   /* 00001000 */
6  a |= (1<<2)|(1<<1); /* 00001110 */
7
8  //Bit(s) loeschen
9  uint8_t a = 0x0F; /* 00001111 */
10 a &= ~(1<<2);     /* 00001011 */
11
12 uint8_t a = 0x0F;       /* 00001111 */
13 a &= ~((1<<2)|(1<<1)); /* 00001001 */
14
15 //Bit(s) toggeln
16 uint8_t a = 0x0F; /* 00001111 */
17 a ^= (1<<2);      /* 00001011 */
18 a ^= (1<<2);      /* 00001111 */
```

Listing 4.1: (Unbekannte) Bitoperationen

Fazit

Es wird deutlich, dass die zunächst einfach wirkende Programmierung von Mikrocontrollern durch Registerbelegungen von verschiedenen Kompetenzen auf unterschiedlichen Ebenen abhängt. Kompetenzen als Voraussetzungen müssen mit dem neuen Anwendungsgebiet verknüpft werden. Der Lerngegenstand Mikrocontroller muss dabei zunächst auf Systemebene eingeführt werden, woraus ersichtlich wird, welche Register für welche Funktionen verantwortlich sind.

Zusammenfassung

Die im Rahmen dieser Arbeit abgeleiteten kognitiven Strukturen verbinden die mit den Lernhürden (LH_{1-4}) verknüpften Kompetenzen und Fachkonzepte und ermöglichen so die Diskussion zu Lehr-Lernprozessen, welche durch ihre Sequenzierung implizit verschiedene Vorwissensrelationen besitzen. Da diese nicht den gesamten Lehr-Lernprozess beschreiben, sind hier nicht alle Relationstypen, Abstraktionsebenen sowie kognitiven Stufen gleichermaßen vertreten. Dennoch wird deutlich, dass das Erreichen einer Kompetenz – gegebenenfalls auf einer bestimmten Abstraktionsebene – von dem Erreichen kognitiver Niveaustufen anderer Kompetenzfacetten abhängig ist und über die Visualisierung kognitiver Strukturen zugänglich wird.

4.4. Kompetenzfördernde Software

[Freischlad, 2010] betrachtet Lernsoftware für entdeckendes Lernen als Komponente didaktischer Systeme. Durch die Auswahl von Lernsoftware sei bereits ein Handlungsrahmen vorgegeben, mit der Konsequenz, dass die Aufmerksamkeit der Lernenden eingeschränkt werde, da nur Entdeckungen möglich seien, die der Entwickler vorgesehen habe [Brinda, 2004]. Bei der Entwicklung lernförderlicher Software für das Entwurfs- und Anwendungspraktikum steht nicht die Lernsoftware im Zentrum, sondern das didaktische Setting im Labor, das vorwiegend lernunterstützende Hardware und die Entwicklungsumgebung *Virtual Workspace* beinhaltet (vgl. Kapitel 3). Die dort integrierten Simulationsumgebungen wurden nicht ausschließlich für Lehr-Lernprozesse entwickelt. Die Kritik, dass Lehr-Lernprozesse sich oftmals an der Lernsoftware ausrichten würden und aufgrund des hohen Aufwandes vor dem Erzielen von Ergebnissen die Akzeptanz fehle, ist hier daher nicht berechtigt (vgl. [Hinostroza et al., 2000], [Freischlad, 2010]). Bei der Entwicklung lernförderlicher Software zum Entwurfs- und Anwendungspraktikum stand der Lehr-Lernprozess, genauer die identifizierten Lernhürden, im Vordergrund [Jaschke und Büchner, 2013]. Wichtig war, dass sich die Lernumgebung an die individuellen Lernhürden, wie sie sich bei sich ändernden heterogenen Gruppen ergeben, flexibel anpassen lässt. Die Erkundung des Systems mit Lernsoftware steht im Gegensatz zu [Stechert und Schubert, 2007] sowie [Brinda, 2004] nicht im Mittelpunkt. Dies erfolgt in realen Laborexperimenten.

> „Entdeckendes Lernen mit realen Informatiksystemen erfordert eine Reduktion ihrer Komplexität. Im Sinne von Explorationsmodulen soll es möglich sein, Fachkonzepte sowohl fokussiert wie auch im Verbund zu untersuchen. In der Informatik werden Simulationen verwendet, um Erkenntnisse zu erlangen, die in der Realität beispielsweise wegen ihrer Komplexität nicht zugänglich sind“ [Freischlad, 2010, S. 156].

Unzugänglich sind diese Erkenntnisse unter Umständen nicht nur aufgrund ihrer Komplexität sondern auch bedingt durch die Wahl eines Aufgabenprofils. Wird eine Aufgabe in der Form gestellt, dass durch die Verwendung von Bibliotheken ein *Puzzle-Ansatz* (vgl. Abschnitt 2.3) zur Lösung ausreicht, bleiben die in Bibliotheken bereitgestellten Basisoperationen verborgen. Daher ist es wichtig ein System durch mehrere Sichten zu erkunden (vgl. [Stechert, 2009]).

„Explorative Learning and Visualization Environment (ELVE)“

Die hier vorgestellte Umgebung wurde in der Hauptverantwortung des Autors am Lehrstuhl Didaktik der Informatik und E-Learning der Universität Siegen entwickelt. Um die beschriebenen Hürden bei der Implementierung einfacher Algorithmen und Bitoperationen zu überwinden und so die Programmierung vom Mikrocontroller mit Registerbelegungen zu erlernen, wurde die Plattform *Explorative*

Learning and Visualization Environment (ELVE) als lernunterstützende Software entwickelt (vgl. [Jaschke und Büchner, 2013]). ELVE erlaubt Aufgaben in einer Art Lückentext zu stellen, in denen die Studierenden verschiedene Registerkonfigurationen testen und die Auswirkungen auf das System beobachten können. Es besteht aus drei Komponenten, einem *Backend* zur Aufgabengestaltung für Tutoren, einem *Frontend* zur Lernunterstützung für Studierende sowie einem Webserver zur Bereitstellung von ELVE (siehe Abbildung 4.14).

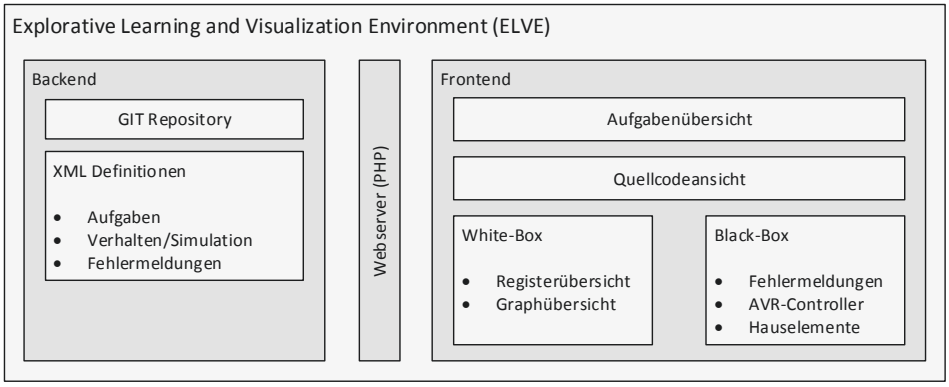


Abbildung 4.14.: Struktur – Explorative Learning and Visualization Environment (ELVE)

Das webbasierte HTML-Frontend besteht aus drei Bereichen sowie einer übergeordneten Ebene zur Navigation zwischen verschiedenen Aufgaben. Systemvoraussetzung ist ein *javascript*-fähiger Browser.

Wichtig ist, dass ELVE weder eine Emulation noch eine vollständige Simulation der realen Hardwareumgebung (ATXmega128A1/Modellhaus), sondern eine didaktisch reduzierte und aufbereitete Visualisierung des Systems darstellt. Durch die Flexibilität bei der Gestaltung von Aufgaben im Backend ist es möglich, ein beliebiges Verhalten eines Systems auf Basis von Verhaltensregeln und Benutzereingaben grafisch darzustellen sowie einfache Berechnungen, beispielsweise des *Duty-Cycles* einer Pulsweitenmodulation, durchzuführen. Hierbei handelt es sich also um ein sehr beschränktes Simulationsmodell.

Quellcodeansicht

In der Quellcodeansicht im linken Teil der Abbildung 4.15 wird den Studierenden ein Codegerüst in einem typischen Design einer Entwicklungsumgebung inklusive Zeilennummern dargestellt. *Header-Dateien* mit Standardbibliotheken für Ein- und Ausgabe, mikrocontrollerspezifischen Registerbeschreibungen und spezielle, für Aufgaben des Modellhauses angepasste, Bibliotheken sind per *Includeanweisung* eingebunden und verlinkt, sodass die Studierenden direkten Zugriff darauf

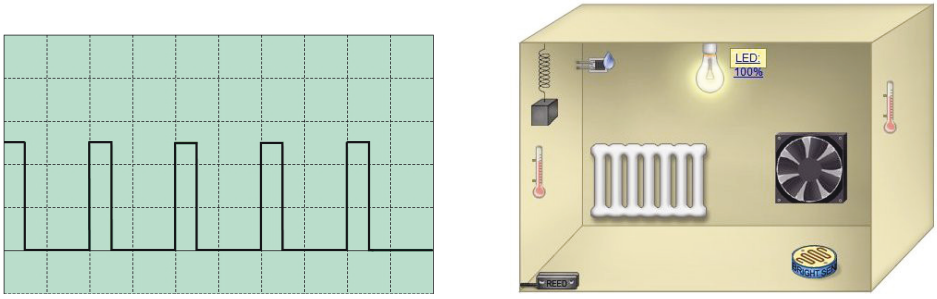
Abbildung 4.15.: Übersicht – Explorative Learning and Visualization Environment (ELVE) [Jaschke und Büchner, 2013, S. 3]

haben. So können sie Teile der Bibliothek erkunden und Designentscheidungen nachvollziehen. Weiterführende Literatur kann im Quelltext als Kommentar verlinkt werden und ermöglicht einen schnellen Zugriff direkt aus der Aufgabenstellung heraus. Dies schließt den Zugriff auf Passagen der Datenblätter mit ein. Die Studierenden sollen nun die definierten Lücken ausfüllen, um beispielsweise einfache Registerzuweisungen in den Vorbereitungsphasen zu erproben. Das dargestellte Beispiel (siehe Abbildung 4.15) dient den Studierenden als Einführung in ELVE, indem sie zuerst binäre und hexadezimale Werte in die Lücken eintragen, dann übernehmen und letztlich das Ergebnis der Registerzuweisung sowohl in der Registeransicht als auch durch eine leuchtende LED in der Verhaltensansicht auf dem Mikrocontroller visualisiert bekommen.

Registeransicht

In der Registeransicht (siehe Abbildung 4.15, oben/rechts) werden die eingetragenen Werte der Aufgabenstellung in 8- und 16-Bit Blöcken abgebildet. Dies ermög-

licht eine binäre Darstellung der Registerzuweisungen auch wenn die Studierenden die dezimale oder hexadezimale Schreibweise verwenden. Handelt es sich bei den erzeugten Ausgaben an einem *Pin* des Mikrocontrollers um eine Pulsweitenmodulation, so kann das Signal als Graph eines Oszilloskops visualisiert werden (siehe Abbildung 4.16(a)). Um erneut eine Verknüpfung mit dem realen Oszilloskop aus der physikalisch-technischen Einführung herzustellen, können die Studierenden die X- und Y-Achse in den Intervallen des realen Oszilloskops verändern und den Graph anpassen. Hierdurch ist ein zügiges Erkunden des Verhaltens einer Pulsweitenmodulation bei Änderungen der verschiedenen Register des Mikrocontrollers möglich, ohne Quelltext zu *compilieren*, zu *flashen* sowie in der Laborpräsenz zu testen und auszumessen. In diesem Punkt stellt ELVE also mehr als eine Visualisierungsumgebung dar, da das Verhalten des Mikrocontrollers simuliert wird. Dies wird dadurch ermöglicht, dass die Zuweisungen in den Lücken zu Timer-Registern ausgewertet und daraus die Frequenz und der *Duty-Cycle* berechnet werden.



(a) Oszilloskop (b) Modellhaus

Abbildung 4.16.: Ansichten – Explorative Learning and Visualization Environment (ELVE) [Jaschke und Büchner, 2013]

Verhaltensansicht

In der Verhaltensansicht (siehe Abbildung 4.15, unten/rechts) können die Studierenden sich zum einen das Modellhaus mit verschiedenen Werten der Aktoren und Sensoren (siehe Abbildung 4.16(b)), zum anderen das Mikrocontrollerboard anzeigen lassen. Die auf dem Board integrierten LEDs können in der Visualisierung eingeschaltet werden. Außerdem ist es möglich, bedingte Anweisungen in der Quellcodeansicht durch Eingaben zu beeinflussen. Dazu können die Taster via Mausklick betätigt werden. Wichtig im Unterschied zu einer Simulation ist, dass das „Programm“ der Quellcodeansicht nicht periodisch durchlaufen wird, sondern Sensordaten einmal erfasst (durch manuelle Eingabe der Werte), verarbeitet und ausgegeben werden. Die Eingabe der Taster wird für einen Durchlauf in einer Variablen gespeichert und – wie auch die Registerzuweisungen – einmalig ausgewertet.

Erstellung von Aufgaben – Backend

Bei der Gestaltung von ELVE wurde auf eine besonders flexible Art der Aufgabengestaltung Wert gelegt, welche keinerlei Restriktionen hinsichtlich der Visualisierungsmöglichkeiten (grafisch) besitzt beziehungsweise auf ein bestimmtes System (z. B. Mikrocontroller) beschränkt ist und daher zielgruppen- und systemunabhängig eingesetzt werden kann. Die Definition der Aufgaben erfolgt mittels XML. Die Tutoren benötigen keine speziellen Kenntnisse über die Funktionsweise von ELVE, sondern definieren das Verhalten des Modellhauses, Mikrocontrollerboards und Oszilloskops, welches gezeigt werden soll, wenn miteinander verknüpfte Lücken korrekt ausgefüllt wurden. Da hier in der Regel verschiedene Repräsentationen als korrekt gelten (binäre, dezimale oder hexadezimale Schreibweise), muss in der Aufgabenstellung jede dieser Möglichkeiten als *richtig* definiert werden. Durch die Verwendung textbasierter Lücken ist ELVE ferner nicht auf eine bestimmte Programmiersprache begrenzt. Der Aufbau einer Aufgabe in ELVE ist in Listing 4.2 aufgeführt. Man erkennt an diesem Beispiel, dass eine Aufgabe aus einer Aufgabenbeschreibung (filedescription), dem Lückentext (challengecode), einer Definition vertauschbarer Antworten (swapablecontent), einer Verhaltensbeschreibung (actionlist) sowie den verwendeten Tastern (tasterlist) und Sensorwerten (sensorlist) besteht. Eine dadurch definierte Aufgabe wird durch Hinzufügen zum *GIT-Repository*[1] automatisch den Studierenden per online Zugriff bereitgestellt.

```
1  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2    <xsd:element name="challengeXML">
3    <xsd:complexType>
4      <xsd:sequence>
5        <xsd:element ref="filedescription" minOccurs="0"
             maxOccurs="1"/>
6        <xsd:element ref="challengecode"/>
7        <xsd:element ref="swapablecontent" minOccurs="0"
             maxOccurs="1"/>
8        <xsd:element ref="actionlist"/>
9        <xsd:element ref="tasterlist" minOccurs="0" maxOccurs
             ="1"/>
10       <xsd:element ref="sensorlist" minOccurs="0" maxOccurs
             ="1"/>
11     </xsd:sequence>
12   </xsd:complexType>
13 </xsd:element>
```

Listing 4.2: XML-Schema der Aufgaben (reduziert auf Hauptelemente)

Didaktische Bewertung

Durch die Funktionsvielfalt professioneller Software ist eine Orientierung bei der Navigation und Nutzung der verschiedenen Eingabe- und Konfigurationsmasken

[1] Verteilte Versionsverwaltung von Dateien [http://git-scm.com/, Abruf: 12/2014].

für Studierende schwierig. Zur Vorbereitung auf die jeweils nächste Laboreinheit und Nachbereitung der vergangenen Praxisphase war es den Studierenden bislang nicht möglich, ihre Erfahrungen durch Wiederholen, Testen und Erweitern der entwickelten Lösung zu manifestieren. Natürlich ist die Lösungsvielfalt der in ELVE gestellten Aufgaben sehr eingeschränkt, da lediglich Lücken zu füllen sind und Eingaben mittels Taster vorgenommen werden können. Dennoch eignet sich dieser Ansatz insbesondere für Lerneinheiten, in denen die Studierenden erste Erfahrungen mit der Programmierung von Teilen einer Steuerung oder Regelung sammeln. Dazu gehört auch, dass in der Vorbereitungsphase partielle Inhalte aus den Datenblättern herausgesucht, interpretiert und getestet werden können. Innerhalb mehrerer Aufgaben können die Studierenden angeleitet Lösungsvarianten kennenlernen und durch Variation der Registerbelegungen die jeweiligen Vor- und Nachteile erfassen. Wie bereits erwähnt ist die Entwicklung eigener Lösungsansätze sowie deren Überprüfung in praxisnahen Aufgabenstellungen das Ziel in projektorientierten Laborpraktika. In den Präsenzphasen allein ist dies aus Mangel an Zeit nicht möglich. Eine weitere Besonderheit dieses Ansatzes ist, dass mehrere Sichten das aktuelle nach außen sichtbare Verhalten, aber auch einen Einblick in die innere Struktur, die Registerbelegungen, des Systems bieten. Bei der Implementierung einer Pulsweitenmodulation können die Studierenden das Ergebnis überprüfen, ohne über Kompetenzen beim Umgang mit der Laborausstattung zu verfügen. Es ist also möglich Kompetenzen zu fördern, zu deren Erreichen man bisher viele andere Kompetenzen als Voraussetzung benötigte.

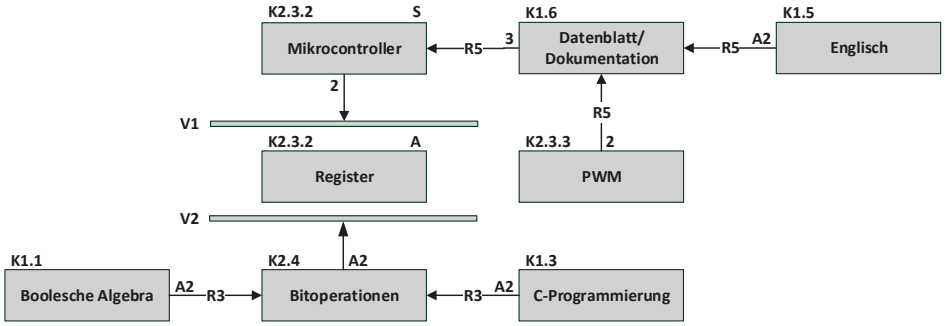


Abbildung 4.17.: Veränderung kognitiver Strukturen durch den Einsatz lernförderlicher Software

Abbildung 4.17 zeigt die zur Lernhürde LH_4 – Register und Bitoperationen – abgeleitete kognitive Struktur. Daran lässt sich verdeutlichen, inwiefern ELVE diese Strukturen verändern kann. Werden Aufgaben gestellt, in denen die Studierenden Register auswählen sollen, kann die Auswirkung auf das System visualisiert werden, ohne dass sie Bitoperationen, Boolesche Algebra oder die C-Programmierung anwenden können. Die kognitive Struktur kann also hier an *V2* abgetrennt wer-

den. Andererseits können Aufgaben zu Bitoperationen gestellt werden, in denen die Register vorgegeben sind. Dann müssen die Studierenden keine Kenntnisse über die verwendete Hardware besitzen, sodass die kognitive Struktur an *V1* aufgetrennt werden kann. Für die Lehr-Lernprozesse heißt das, dass durch den Einsatz lernförderlicher Software die Erarbeitungsreihenfolge, welche sich durch die Relationen von Kompetenzen ergibt, verändert werden kann. Das führt bezogen auf die Zielgruppe mit heterogenen Vorkenntnissen zu einer höheren Flexibilität. Für die weiteren Forschungsarbeiten zum didaktischen System für das Design eingebetteter Systeme bedeutet diese Verzahnung von lernförderlicher Hard- und Software und kognitiven Strukturen, dass diese gemeinsam zu erforschen sind.

4.5. Zusammenfassung und Fazit

Die Arbeiten zur Kompetenzmodellierung und die Identifizierung typischer Lernhürden führten zu einer ersten Ausdifferenzierung der Kompetenzsubdimension *(K2.3) Design* und der Darstellung kognitiver Strukturen. Es wurde gezeigt, dass das selbständige Erarbeiten der Grundlagen ohne eine Orientierung im Fach und der zugehörigen Terminologie nicht möglich ist. Der Lerngegenstand, die Laborausstattung sowie die lernunterstützende Hard- und Software des Entwurfs- und Anwendungspraktikums werden als geeignet angesehen. Mikrocontroller dienen zur Einführung in die Entwicklung eingebetteter Systeme, da nicht das Design vieler, sondern überwiegend die Implementierung einiger Systemkomponenten und -funktionen im Vordergrund steht. Die Hausautomation bietet einen lebensweltnahen Bezug zu eingebetteten Systemen und die Möglichkeit der Integration von Smartphones hin zu cyber-physischen Systemen. Die Laborausstattung ermöglicht die Integration hardwarenaher Versuche und durch die Integration des *Virtual Workspace* auch eine Entwicklungsumgebung für Mikrocontroller in einem *Blended-Learning* Ansatz.

Weiterentwicklung des EAP zur Förderung von Kompetenzen für das Design eingebetteter Systeme nach einem systemorientierten Ansatz

Die Erarbeitungsreihenfolge und die Erarbeitungsstrategien im Entwurfs- und Anwendungspraktikum bedürfen trotz vieler Lernerfolge einer Überarbeitung, was durch die Visualisierung kognitiver Strukturen gezeigt werden konnte. Das Praktikum besteht derzeit aus acht Versuchen, einer Einführungsveranstaltung und einer zweiwöchigen Informationsphase, in der die Studierenden die Grundlagen der Elektrotechnik sowie die Funktionsweise der Laborausstattung weitestgehend selbständig erarbeiten. Innerhalb dieser Informationsphase ist weniger der Zeitumfang als die Art der Erarbeitung als Problem identifiziert worden. Als wesentliche Hürde bei der Förderung von Kompetenzen wurde die Fachsprache benannt, wel-

che systematisch vor den jeweiligen Versuchen zu einem Fachkonzept eingeführt werden muss. Ein rein autodidaktisches Vorgehen ist nicht zielführend. In der einführenden Veranstaltung soll daher nicht nur der Lerngegenstand, das Modellhaus, sondern das Fachgebiet, die Entwicklung eingebetteter Systeme, vorgestellt werden. Dies umfasst auch eine wichtige Kompetenz des empirisch verfeinerten Kompetenzstrukturmodells *(K2.1) Die Studierenden können zielgerichtet und strukturiert beim Entwurf vorgehen.* Den Studierenden muss also verdeutlicht werden, wie der Entwurf in der Praxis erfolgt und warum das Vorgehen im Labor davon abweicht.

Dekonstruktion als Einführung nach einem systemorientierten Ansatz

Es kann davon ausgegangen werden, dass Studierende in den ersten Semestern, welche noch keine Erfahrung mit der Entwicklung von Soft- und Hardwaresystemen haben, ebenso wenige Kompetenzen in den Bereichen Modelle, Architekturen, Technologien, Randbedingungen, Zeitverhalten und Tools für das Design eingebetteter System besitzen (*K2.3*). Der Umfang zur Förderung von Designkompetenzen ist demnach zu vielschichtig für eine einzelne Lehrveranstaltung. Die Lösungsvielfalt innerhalb des Projektes Hausautomation ist groß, weshalb den Studierenden Implementierungsdetails bereits in Form von Bibliotheken vorgegeben wurden. Die Hardware des Systems kann von den Studierenden nicht verändert werden. Ein rein konstruktivistisches Vorgehen kann aufgrund dieser Lösungsvielfalt bei gleichzeitig fehlendem Verständnis für die Systemstruktur und Komponenten nicht zum Erfolg führen (vgl. Kapitel 3). [Hampel et al., 1999] schlagen vor, ein gegebenes System durch die Analyse der Design- und Implementierungsentscheidungen zu erkunden, statt der Durchführung einer selbständigen Neuimplementierung. Durch ein systematisches Vorgehen innerhalb dieser Dekonstruktionsphasen lernen die Studierenden verschiedene, teilweise durch die Technologie der Systemkomponenten vorgegebene, Lösungsalternativen kennen und diese zu vergleichen. Ein ständiger Wechsel von Phasen der Dekonstruktion und Konstruktion soll dabei Fehlentscheidungen umfassen, ohne zu einer Versuch-Irrtum-Strategie überzugehen. Entscheidend im Entwurfs- und Anwendungspraktikum ist, dass nicht Sprachkonstrukte aus den Bibliotheken unreflektiert übernommen werden, sondern Basisoperationen zur Nutzung dieser selbständig angewandt werden und die Auswirkung auf die Hardware nachvollzogen wird. Eine ausführliche Dokumentation der Bibliotheken und ihrer Verbindungen ist daher entscheidend und muss im Gegensatz zur bisherigen Praxis grafisch aufbereitet werden. Um ein Kopieren der vorgegebenen Algorithmen zu verhindern, ist davon abzusehen eine Musterlösung vollständig freizugeben. Vielmehr sollte die Struktur der Musterimplementierung, deren zugrundeliegenden Überlegungen und ausgewählte Aspekte Teil der Lehrveranstaltung werden (Stufe *Komponenten+Parameter*). Darüber hinaus sind sodann Teilaufgaben zu stellen, welche die vorhandenen Komponenten mehr als bislang um weitere Teilaspekte ergänzen.

Als geeigneter Einstieg wird die Betrachtung des physikalischen Prozesses und die Funktionsweise der Sensoren angesehen, bei denen noch nicht die elektrotechnischen Grundlagen (Schaltungen), sondern die physikalischen Prinzipien im Vordergrund stehen. Warum ändert sich der Widerstand eines resistiven Temperatursensors? Wie ist der Verlauf von Widerstandswert zu Temperatur? Mit diesen Fragestellungen kann die Vorgehensweise bei der Analyse von Datenblättern verdeutlicht und der physikalische Prozess somit Teil der Aufgabenstellung werden. Hier sollten die Studierenden wichtige Passagen identifizieren und Bauteile anhand der analysierten Parameter vergleichen können. Ausgehend von den Datenblättern lassen sich die wesentlichen Parameter elektrotechnischer Bauteile bestimmen, welche die Einführung in die Grundlagen der Elektrotechnik motivieren. Über die in der Hausautomation verwendeten Bauteile hinaus wird empfohlen Datenblätter alternativer Bauteile zu betrachten, um bei der Analyse auch die Unterschiede von Bauteilen und deren Auswirkungen auf das System zu verdeutlichen. Dies ist auch durch die Definition von Randbedingungen möglich, welche nur bei Verwendung bestimmter Komponenten eingehalten werden können.

Ferner empfiehlt der Autor die Ergänzung einer Vorlesungseinheit als Orientierungshilfe im Fach und Einführung der wichtigsten Fachtermini, bevor die Studierenden im Selbststudium die Steuerung des Modellhauses dekonstruieren. Aufgrund der geringen Vorkenntnisse der Studierenden muss diesen die Möglichkeit gegeben werden sich im Fachgebiet zu orientieren, ohne auf eine autodidaktische Einführung angewiesen zu sein. Die weitere Entwicklung lernunterstützender Software kann dazu dienen weitere Lernhürden zu überwinden. Eine tutorengeleitete Einführung in das Laborequipment wäre sehr aufwändig. Eine didaktisch reduzierte Einführung mittels E-Learning und einer Visualisierung analog zu ELVE scheint hier Erfolg versprechend.

Erst anschließend sollten die Studierenden das System erkunden, indem sie die Bauteile des Modellhauses identifizieren und – sofern möglich – die zugehörigen Datenblätter sammeln sowie einen Schaltplan anfertigen. Dieser Schaltplan könnte dann mit dem Musterschaltplan verglichen und im Plenum diskutiert werden. Ein Beispielprogramm ermöglicht bereits eine Regelung der Prozesse, in der die Sollwerte von den Studierenden mittels Smartphone vorgegeben werden können. In den Vor- und Nachbereitungsphasen sollten die elektrotechnischen Grundlagen auf die real existierenden Schaltungen und Bauteile beschränkt werden. Das System, also das Zusammenwirken aller Komponenten, soll von den Studierenden durch Beobachtung des nach außen sichtbaren Verhaltens erkundet werden, was analoge und digitale Signale einschließt. Im Anschluss an die Erkundung des Systems und Messen der elektrischen Größen kann das Blockdiagramm des Mikrocontrollers als Ausgangspunkt für die Dekonstruktion einzelner Bibliotheken der Regelung verwendet werden (siehe Abbildung A.1). ELVE wird dann als lernunterstützende Software zur Einführung in die hardwarenahe Programmierung genutzt.

Umstrukturierung des Entwurfs- und Anwendungspraktikums

Die vorgegebene Sequenzierung des Entwurfs- und Anwendungspraktikums erlaubt durch die Aufgaben in einer Art Tutorium eine gute Orientierung im Lernprozess. Diese Strukturierung bietet allerdings keine Möglichkeiten, eigene Lösungsalternativen in ausreichendem Maße zu überprüfen. Der enge Zeitrahmen verhindert eine ausführliche Exploration des Systems. Sinnvoller ist es, die Vorbereitungsphasen zur Hypothesenbildung und die Präsenzphasen zu deren Prüfung zu nutzen, was ein strukturiertes Vorgehen impliziert. Gezeigt wurde, dass die aktuell vorherrschende Sequenzierung nicht geeignet ist, da verschiedene Kompetenzfacetten zugleich Vorbedingung für ein Verständnis und die Anwendung der Fachkonzepte darstellen. Das heißt, die starre Struktur des Entwurfs- und Anwendungspraktikums sollte aufgebrochen und den Studierenden lediglich betreute Laborzeiten angeboten werden. Ausgehend von den entwickelten kognitiven Strukturen kann das Erkunden der Funktionsweise eingebetteter Systeme anhand der Wirkprinzipien von Sensoren und Aktoren sowie die Dekonstruktion des Modellhauses als geeigneter Zugang verstanden werden. Abschließend ist eine mögliche Strukturierung aufgeführt.

- Architekturen eingebetteter Systeme
- Elektrotechnische Grundlagen
- Physikalisch-technische Einführung (Dekonstruktion der Hardware und Messen elektrischer Größen)
  - Sensoren zur Erfassung der physikalischen Realität
  - Aktoren als nach außen sichtbares Verhalten
  - Analoge und digitale Signale
- Design und Implementierung (Dekonstruktion der Software im technischen Prozess)
  - Konzepte hardwarenaher Programmierung
  - Digitale Verarbeitung analoger Signale
  - Ansteuerung von Aktoren
  - Regelung des peripheren technischen Prozesses

Diese Auflistung ist nicht als strikte Erarbeitungsreihenfolge sondern als Gliederung von Themenkomplexen zu verstehen. Demnach wird empfohlen die Hard- und Software des Systems zu dekonstruieren und anschließend eigene Lösungen zu erarbeiten. Dieser Ansatz eignet sich jedoch nicht zur Einführung in die Architekturen eingebetteter Systeme sowie die elektrotechnischen Grundlagen. Daher ist es ratsam, diese zu Beginn des Praktikums in klassischen Vorlesungs- und Übungsphasen vorzustellen, wobei sich hier auf die für das Entwurfs- und Anwen-

dungspraktikum notwendigen Grundlagen beschränkt werden sollte. So wird der Umfang kognitiver Strukturen auf das notwendige Maß reduziert, was gleichzeitig die Anzahl möglicher Lernhürden verringert.

5. Zusammenfassung, Fazit und Ausblick

5.1. Zusammenfassung

Die Relevanz eingebetteter Systeme für die Gesellschaft, Wirtschaft und Forschung ist unbestritten. Trotz dieser Erkenntnis existiert ein Forschungsmangel zur kompetenzorientierten Hochschuldidaktik der technischen Informatik, welche eine adäquate Ausbildung künftiger Entwickler eingebetteter Systeme unterstützt. Im Forschungsprojekt des Autors wurde ein Forschungsbeitrag zur Hochschuldidaktik der technischen Informatik, in der Studierende der Informatik (Systementwickler) die Zielgruppe bildeten, geleistet. Ein Ziel war es, eine theoretisch fundierte Strukturierung von Lehr-Lernprozessen der technischen Informatik – im Speziellen zu eingebetteten Systemen – zu ermöglichen, weshalb Kompetenzen künftiger Entwickler hinsichtlich ihrer Niveaustufen und Vernetzung untersucht wurden.

Diese Arbeit bietet eine neue Sichtweise auf die Hochschuldidaktik der technischen Informatik, indem von kulturspezifisch geprägten Praxisberichten zu vergleichbaren und theoretisch fundierten Konzepten einer Kompetenzorientierung bei der Entwicklung eingebetteter Systeme übergegangen wird. Grundlage für die Forschungsarbeit des Autors waren die Erkenntnisse aus den Arbeitspaketen des von der Deutschen Forschungsgemeinschaft geförderten Projekts *Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen (KOMINA)*, an denen er maßgeblich beteiligt war. Dazu gehören die normative Ableitung eines Kompetenzstrukturmodells, die empirische Überprüfung des Kompetenzstrukturmodells, die Entwicklung des Entwurfs- und Anwendungspraktikums, dessen Evaluation sowie die Entwicklung lernförderlicher Hard- und Software.

Kognitive Strukturen als Komponente didaktischer Systeme wurden in diesem Forschungsprojekt erforscht. Sie dienten als Basis für die informatikdidaktische Verfeinerung des in KOMINA empirisch evaluierten Kompetenzstrukturmodells für das Entwickeln eingebetteter Mikro- und Nanosysteme. In diesem Beitrag zur Grundlagenforschung zur Hochschuldidaktik der technischen Informatik stand die Diskussion didaktischer Entscheidungen (Diskussionsfunktion didaktischer Systeme) bei der Gestaltung von Lehr-Lernprozessen und Pfaden der Kompetenzaneignung im Vordergrund. Deshalb wurden die Anforderungen an die Darstellung kognitiver Strukturen – Ausdrucksstärke, Übersichtlichkeit und Nachvollziehbarkeit – zu-

gunsten der Diskussionsfunktion angepasst. Zusammenfassend müssen zur Visualisierung kognitiver Strukturen Lernstände, Sequenzen der Kompetenzförderung, Vorbedingungen, kognitive Stufen sowie die Abstraktionsebenen zugleich beschrieben werden. Ferner sind die Arten von Relationen als Begründung der Reihenfolge und gegebenenfalls damit in Verbindung stehende heuristische Strukturen, also Arten der Problemlösung, welche hier gefördert werden sollen, zu bestimmen. Es wurden Ansprüche an die Darstellung kognitiver Strukturen abgeleitet, mit denen es möglich ist den konkreten Lehr-Lernprozess hinsichtlich seiner kognitiven Anforderungen zu untersuchen und zu visualisieren. Eine Orientierung an der Breite von Kompetenzen des Kompetenzstrukturmodells war bei der Gestaltung von Lehr-Lernprozessen nur schwer möglich. Die Beschränkung auf eine Subdimension war unzureichend (z. B. *(K2.3) Design*). Als Erweiterung zur tabellarischen Darstellung des empirisch verfeinerten Kompetenzstrukturmodells wurden die Kompetenzen anhand der verwendeten Ankerbeispiele summarisch beschrieben, was in weiteren Forschungsarbeiten als Kodierleitfaden dient.

Für das Entwurfs- und Anwendungspraktikum wurden acht Experimente entwickelt, welche Kompetenzen des empirisch verfeinerten Kompetenzstrukturmodells unter Berücksichtigung der institutionellen Besonderheiten fördern. Eine weiterführende Analyse der, mithilfe einer vom Autor modifizierten Taxonomie zur Identifikation von Lernhürden, durchgeführten strukturierten Beobachtung im Entwurfs- und Anwendungspraktikum führte zu der Bestimmung von vier wesentlichen Barrieren der Kompetenzaneignung – Dokumentationen und Datenblätter zur Analyse von Funktionsprinzipien, Vorgehensweise zur Strukturierung des Entwicklungsprozesses, Laborausstattung als Mittel zur Erzeugung und Messung elektrischer Größen sowie Register- und Bitoperationen als Konzepte hardwarenaher Programmierung.

Zur Ableitung von Empfehlungen zu Lehr-Lernprozessen und Laborpraktika sowie der Entwicklung kognitiver Strukturen als Forschungsgrundlage eines didaktischen Systems für das Design eingebetteter Systeme wurde zunächst die Subdimension *(K2.3) Design* ausdifferenziert zu *(K2.3.1) Modelle*, *(K2.3.2) Architekturen*, *(K2.3.3)* Technologien, *(K2.3.4) Randbedingungen*, *(K2.3.5) Zeitverhalten* und *(K2.3.6) Tools*. Die Analyse zweier Befragungen von Erstsemesterstudierenden der Universität Siegen und der Universität Erlangen-Nürnberg gab Aufschluss über implizites Mikrosystemverständnis und die Vorerfahrungen mit eingebetteten Systemen in der Sekundarstufe II. Um zu einer Empfehlung für die Gestaltung von Lehr-Lernprozessen zu gelangen, ist eine kognitive Struktur je zu erlangender Kompetenzfacette abzuleiten, wobei sich in dieser Arbeit zunächst auf Kompetenzen mit Verbindung zu den identifizierten Lernhürden beschränkt wurde. Durch die Ableitung der vier kognitiven Strukturen und der Anwendung der modifizierten Taxonomie konnte eine theoretisch fundierte Empfehlung zur Weiterentwicklung des Entwurfs- und Anwendungspraktikums gegeben werden.

Zur Überwindung der beschriebenen Lernhürden wurde die Plattform *Explorative*

Learning and Visualization Environment in der Hauptverantwortung des Autors am Lehrstuhl Didaktik der Informatik und E-Learning der Universität Siegen entwickelt, welche es erlaubt Aufgaben in einer Art Lückentext zu stellen, in denen die Studierenden verschiedene Registerkonfigurationen testen und die Auswirkungen auf das System beobachten können. Dieser Ansatz eignet sich insbesondere für Lerneinheiten, in denen die Studierenden erste Erfahrungen mit der Programmierung von Teilen einer Steuerung oder Regelung sammeln. Es wurde gezeigt, dass eine abgeleitete kognitive Struktur durch den Einsatz lernförderlicher Software verändert werden kann, was eine größere Flexibilität bei der Gestaltung von Lehrveranstaltungen ermöglicht.

Abschließend wurde eine Empfehlung zur Weiterentwicklung des Entwurfs- und Anwendungspraktikums nach einem systemorientierten Ansatz gegeben. Ausgehend von den entwickelten kognitiven Strukturen wurde das Erkunden der Funktionsweise eingebetteter Systeme anhand der Wirkprinzipien von Sensoren und Aktoren sowie der Dekonstruktion des Modellhauses als geeigneter Zugang verstanden. Als erster Schritt zu einer Binnendifferenzierung wird vorgeschlagen, von einer starren Praktikumsstruktur zu einer individuellen Erkundung des Systems und Phasen der Hypothesenprüfung im Labor überzugehen.

5.2. Fazit

Im Forschungsprojekt des Autors wurden drei Forschungsziele (F1-F3) erreicht, welche Lehr-Lernprozesse in Laborpraktika zu eingebetteten Systemen und die Kompetenzforschung in der Hochschuldidaktik der technischen Informatik fundieren. Durch diese Arbeit wird ferner die Sichtweise auf die Informatik erweitert. Danach ist die Entwicklung eingebetteter Systeme nicht Teilmenge eines Curriculums der technischen Informatik. Vielmehr eignet sich die Verwendung cyber-physischer Systeme zur Förderung von Kompetenzen der allgemeinen Informatik. Im Folgenden werden die Forschungsziele mit dem – vorrangig vom Autor geschaffenen – wissenschaftlichen Erkenntnisgewinn aufgeführt.

F1 Überführung des Kompetenzstrukturmodells in kognitive Strukturen eines didaktischen Systems für das Design eingebetteter Systeme

Es wurde gezeigt, dass das empirisch verfeinerte Kompetenzstrukturmodell eine hinreichende Strukturierung von Kompetenzen im Sinne eines Kodierleitfadens für weitere qualitative Inhaltsanalysen bietet, jedoch keine direkte Ableitung möglicher Lehr-Lernprozesse erlaubt. Für anschließende Forschungsarbeiten wurde das Modell summarisch beschrieben sowie die Subdimension *(K2.3) Design* unter einer Erweiterung des Quellenkorpus ausdifferenziert. Diese Strukturierung diente als Grundlage für die Entwicklung kognitiver Strukturen als Komponente eines künftigen didaktischen Systems für das Design eingebetteter Systeme. Im Kon-

zept der didaktischen Systeme werden Fachkonzepte und deren Relationen in Wissensstrukturen definiert. Es wurde verdeutlicht, dass deren Darstellung nicht den Anforderungen an Ausdrucksstärke und Übersichtlichkeit genügte, sodass die Einführung einer alternativen Form der Visualisierung, hin zu kognitiven Strukturen mit Verknüpfungen zum Kompetenzstrukturmodell, erfolgte. Dadurch wird Forschern zur Hochschuldidaktik der technischen Informatik ein Werkzeug zur Ableitung von Pfaden der Kompetenzaneignung bereitgestellt, was die Diskussion von Lehr-Lernprozessen nach aktuellem Forschungsstand ermöglicht.

F2 Vorwissen von Studierenden und Identifikation typischer Lernhürden in Laborpraktika

Das empirisch verfeinerte Kompetenzstrukturmodell wurde exemplarisch im *Entwurfs- und Anwendungspraktikum* umgesetzt. Dazu wurden Laborversuche sowie lernunterstützende Hard- und Software entwickelt. In der Phase der formativen Evaluation wurde eine strukturierte Beobachtung durchgeführt. Durch die Analyse mithilfe der vom Autor modifizierten Taxonomie zur Vergleichbarkeit informatikdidaktischer Publikationen wurden vier typische Lernhürden identifiziert. Wesentliche Barrieren der Kompetenzaneignung waren die Dokumentationen und Datenblätter zur Analyse von Funktionsprinzipien, Vorgehensweise zur Strukturierung des Entwicklungsprozesses, Laborausstattung als Mittel zur Erzeugung und Messung elektrischer Größen sowie Register- und Bitoperationen als Konzepte hardwarenaher Programmierung. Es wurde eine Fehlvorstellung von den zu erwartenden Vorkenntnissen der Studierenden aufgedeckt. Nach der Analyse von Erstsemesterbefragungen und den Beschreibungen vorhergehender Module wurden die Anforderungen an die Studierenden im Entwurfs- und Anwendungspraktikum angepasst und damit die Notwendigkeit für ein verfeinertes Laborkonzept fundiert. Dazu hat der Autor unter Berücksichtigung der Ergebnisse zu den drei Forschungszielen ein Vorgehen nach systemorientierten Ansatz und Phasen der Dekonstruktion vorgeschlagen.

F3 Überwinden typischer Lernhürden durch die Entwicklung lernförderlicher Software

Die Entwicklung von Laborversuchen nach einem systemorientierten Ansatz wurde begründet. Hier steht das Zusammenspiel von Hard- und Software im Vordergrund. Die digitale Verarbeitung analoger Signale als physikalische Realität sowie die Architekturen eingebetteter Systeme waren den Studierenden der Probandengruppe zumeist unbekannt. Daher werden im Laborkonzept die Phasen Abtastung des Eingangssignals sowie Verarbeitung und Rückführung in den Prozess vorrangig betrachtet. Identifizierte Lernhürden zu Registerzuweisungen, Bitoperationen und Messen elektrischer Größen wurden durch die Entwicklung der webbasierten Lernsoftware *Explorative Learning and Visualization Environment* überwunden und innerhalb der Arbeit als Beispiel für lernförderliche Software für ein didaktisches System integriert. Mittels deren Einsatz wird die zur Lernhürde abgeleitete kogni-

tive Struktur verändert. Dadurch wird deutlich, dass die Verbindung lernförderlicher Hard- und Software sowie kognitive Strukturen gemeinsam in didaktischen Systemen zu erforschen sind.

5.3. Ausblick und offene Fragen

Wie sind Aufgaben zur Erfolgskontrolle zu gestalten, welche die Beobachtung des Niveaus einzelner Kompetenzen erlauben?

Das Niveau der intendierten Kompetenzen zu jedem Laborversuch wurde beschrieben. Bislang erfolgte die Evaluation der Laborversuche auf Basis intersubjektiv vergleichbarer Beobachtungen, die, trotz ihrer Vorstrukturierung, keine empirisch gesicherte Aussage über den Kompetenzerwerb zulassen. Kognitive Prozesse, Kommunikationsdetails und Fehlvorstellungen bleiben dem Beobachter teilweise verborgen. Es stellt sich also die Frage, wie Kompetenzen gemessen werden können. In weiteren Arbeiten zur Kompetenzorientierung in der Informatik werden Instrumentarien in Form von Testaufgaben zur Operationalisierung von Kompetenzfacetten entwickelt. Erfahrungen aus deren Validierung lassen Implikationen für die Testentwicklung für die technische Informatik erwarten.

Wie können Konstruktionsphasen binnendifferenziert gestaltet werden, um Studierende in heterogenen Gruppen gleichermaßen zu motivieren und zu fördern?

In diesem Forschungsprojekt wurde die Besonderheit heterogener Studierendengruppen bezogen auf Motivation, Vorgehen und Lernerfolg hervorgehoben. Dabei soll keinesfalls der Eindruck entstehen, dass die Heterogenität als Hindernis für eine erfolgreiche Kompetenzförderung betrachtet wird. Vielmehr sind damit – aus institutioneller Sicht – Herausforderungen verbunden, Lehr-Lernprozesse ebenso differenziert zu gestalten. Laborpraktika sind per se eher statisch gestaltete Lehrveranstaltungen, die nur schwer ad hoc aufgrund unerwarteter Vorkenntnisse der Studierenden angepasst werden können. Es bleibt zu überprüfen, wie Vorbereitungsphasen, Laborphasen und Nachbereitungsphasen binnendifferenziert gestaltet werden können, um auch die Heterogenität der Studierenden mehr als Chance zu nutzen, denn als Problem aufzufassen.

Anwendung des Konzeptes didaktischer Systeme auf Kompetenzen und Konzepte für Studierende im Masterstudium

Die Analyse zur Herleitung kognitiver Strukturen für ein didaktisches System für das Design eingebetteter Systeme basierte auf Empfehlungen und Modulbeschreibungen zur technischen Informatik in Bachelorstudiengängen. Bei einer Spezialisierung auf eingebettete Systeme sowie im Masterstudium Informatik sind höhere kognitive Stufen und weitere Fachkonzepte zu betrachten. Eine Analyse funda-

mentaler Ideen zu eingebetteten Systemen (vgl. [Büchner, 2014]) lässt hier eine Selektion wichtiger und geeigneter Konzepte für diese Zielgruppen erwarten. Darauf aufbauend ist in weiteren Arbeiten eine Entwicklung des didaktischen Systems für das Design eingebetteter Systeme möglich.

Kompetenzen zur Entwicklung eingebetteter Nanosysteme

Die Erforschung der Kompetenzentwicklung mit eingebetteten Nanosystemen steht noch aus. Hier werden Ergebnisse erwartet, welche den notwendigen Grad der Auflösung eines Systems als Black-Box für eine erfolgreiche Kompetenzförderung bestimmen. Erste Arbeiten hierzu beschränken sich auf eine konzeptionelle Ebene (vgl. [Kleinert et al., 2012]), welche durch die Entwicklung geeigneter Laborversuche und der Ableitung kognitiver Strukturen fortgeführt werden kann.

A. Anhang

A.1. Summarische Beschreibung der Kompetenzsubdimensionen des normativen Kompetenzstrukturmodells

K1: Kompetenzen als Voraussetzung

Einführende Lehrveranstaltungen zu eingebetteten Systemen sind im Curriculum informatorischer Studiengänge meist in höheren Semestern des Bachelorstudiums beziehungsweise frühen Semestern des Masterstudiums vorgesehen. Voraussetzung für die Förderung von Entwicklerkompetenzen sind zuvor innerhalb der Sekundarstufe II an allgemeinbildenden Schulen und den Grundlagenveranstaltungen an der Hochschule zu erwerben. Die Struktur dieser als notwendig erachteten Kompetenzen wurde bislang nicht erforscht und hängt unmittelbar mit den intendierten Kompetenzen konkreter Lehr-Lernprozesse, der Zielgruppe sowie institutioneller Profilbildungen ab. Im normativ entwickelten Kompetenzstrukturmodell bilden diese als Voraussetzung bezeichneten Kompetenzen die erste Kompetenzdimension (K1).

K1.1: Mathematik Die [Artist Education Group, 2003] fordert in ihrer Empfehlung eine grundlegende wissenschaftliche Ausbildung in den Bereichen Mathematik und Physik. Studierende der Informatik hätten oftmals nur geringe Grundkenntnisse in nicht-diskreter Mathematik, was die Einführung von Modulen mit Fokus auf Differential- und Integralrechnung begründet. Für die Entwicklung eingebetteter Systeme sind diese insbesondere relevant, da die physikalischen Größen des umgebenden Systems kontinuierlich sind. Für die Überführung in digitale Systeme sind Verfahren wie das *Nyquist-Shannon-Abtasttheorem* zu verstehen. Für die Beschreibung und den Entwurf von Digitalrechnern sind ferner mathematische Hilfsmittel anzuwenden [RWTH Aachen, 2011].

K1.2: Physik Die Einbettung in eine Umgebung impliziert, dass eingebettete Systeme eine physikalische Schnittstelle besitzen. Aktoren wandeln elektrische Signale in eine andere physikalische Größe. Sensoren erfassen physikalische Größen und wandeln diese in elektrische Signale. Sensoren und Aktoren basieren auf verschiedenen physikalischen Wirkprinzipien (mechanisch, ther-

misch, optisch, chemisch, akustisch, induktiv etc.). Auch die Entwicklung eingebetteter Software erfordert die Verbindung physikalischer und informatorischer Grundlagen [Sztipanovits et al., 2005]. Darüber hinaus sind Kenntnisse über physikalische Prinzipien, welche der Funktionsweise elektronischer Rechner zugrunde liegen, sowie die technischen Randbedingungen und Grenzen zu bewerten [Universität Siegen, 2011], [RWTH Aachen, 2011].

K1.3: Informatik Eingebettete Systeme besitzen dedizierte Hard- und Softwarekomponenten, unter anderem Betriebssysteme für eingebettete Echtzeitsysteme oder in Hardware umgesetzte Algorithmen. Die Verarbeitung von Informationen durch Algorithmen oder die Nutzung von informatischen Modellen für das Design von Systemen sind zwei Repräsentanten der Informatik, welche Voraussetzungen für die Entwicklung eingebetteter Systeme sind. Basistechnologien und Grundlagen zur Verarbeitung von Daten mithilfe von Rechnern bestehend aus Hardware und Systemsoftware müssen praxisorientiert eingesetzt werden können [FAU Erlangen-Nürnberg, 2011], [GI, 2011].

K1.4: Elektrotechnik Eingebettete Systeme und deren Peripherie sind aus elektrotechnischen Komponenten aufgebaut. Das Verständnis der Charakteristika und die Verwendung dieser Bauteile sind notwendig, um deren Auswirkungen auf das System zu verstehen. Da die Kommunikation zwischen Komponenten und Systemen über elektrische Signale erfolgt, gehören Effekte und Randbedingungen der physikalischen Verbindung dazu. Methoden für die Berechnung von elektrischen Netzwerken und Zusammenhänge des elektrostatischen Feldes sowie weitere wichtige schaltungstechnische Hintergründe von digitalen Systemen sind anzuwenden [Universität Siegen, 2011] [FAU Erlangen-Nürnberg, 2011].

K1.5: Materialwissenschaften Materialien mit besonderen Eigenschaften, welche zur Herstellung eingebetteter Systeme verwendet werden (z. B. Halbleiter) haben auch einen Einfluss auf das Verhalten des Systems in verschieden Einsatzszenarien. Das Verstehen der Funktionsweise von elektrotechnischen Bauteile wie der LED, dem Transistor oder einem Widerstand ist ohne die Betrachtung der Materialeigenschaften nicht möglich. Materialeigenschaften sollen dabei als Folge des atomaren Aufbaus verstanden werden [Universität Siegen, 2011].

K1.6: Englisch Literatur zu eingebetteten Systemen existiert überwiegend in englischer Sprache. Neben den Beschreibungen elektrotechnischer Komponenten in Datenblättern sind Präsentationen im Geschäftsumfeld und Dokumentationen in englischer Sprache zu verfassen und zu verstehen [GI, 2005], [TH Karlsruhe, 2011].

K1.7: Wissenschaftliches Arbeiten Kompetenzen im wissenschaftlichen Arbeiten, wie das Verstehen, die Analyse und das Erstellen von Dokumentationen sind Voraussetzung für jede Ingenieurs- und Wissenschaftsdisziplin. Darun-

ter fällt auch die Abstraktion (Datenerhebung, Hypothesenformulierung und -überprüfung, Experimente, Analyse) [ACM/IEEE, 2001] sowie die Präsentation im Geschäftsumfeld [TH Karlsruhe, 2011].

K1.8: Lernorganisation Lernorganisation meint die strukturierte Planung und Durchführung der Kompetenzförderung durch Problemlösungen und der Ergebnispräsentation [Universität Siegen, 2011], [RWTH Aachen, 2011].

K2: Kompetenzen zur Entwicklung eingebetteter Systeme

Eine Strukturierung der Entwicklungskompetenzen nach Entwicklungsschritten eines Vorgehensmodells beim Entwurf eingebetteter Systeme ist insofern sinnvoll, als das die Gliederung in die Schritte Anforderungsanalyse, Systemdesign, Implementierung sowie Optimierung und Test allgemeingültig bei der Entwicklung von Informatiksystemen und nicht auf kulturspezifische Vorgehensmodelle beschränkt ist und Demnach können flexibel weitere Subdimensionen aufgenommen werden.

K2.1: Organisation des Entwicklungsprozesses Modelle beschreiben das strukturierte Vorgehen in Entwicklungsprozessen und modularisieren so die Abläufe mit Prozessbausteinen [Friedrich, 2008]. Die Wahl eines geeigneten Modells hängt vom konkreten Projekt und kulturspezifischen Sichtweisen ab. In Kapitel 2 wurden Entwurfsstile und Ebenen der Abstraktion vorgestellt. [Marwedel, 2011] beschreibt darüber hinaus ein vereinfachtes generisches Modell, dass wie das Brezel-Modell, V-Modell etc. für ein konkretes Projekt angepasst werden muss. Unabhängig von den konkreten Varianten dieser Vorgehensmodelle sind Kompetenzen zur Strukturierung des Entwicklungsprozesses notwendig. Dabei sollen die Studierenden das Vorgehen hin zu kreativen selbständigen Lösungen zielgerichtet planen, durchführen und dokumentieren [Universität Siegen, 2011], [TU München, 2011].

K2.2: Anforderungsanalyse In der Anforderungsanalyse werden die funktionalen und nicht-funktionalen Anforderungen an das System aus Hardware, Software und Vernetzungskomponenten ermittelt. Nicht-funktionale Anforderungen wie Temperaturstabilität, Größe und Leistungsaufnahme sind als besondere Randbedingungen des Entwurfs eingebetteter Systeme zu erläutern [TH Karlsruhe, 2011] [ACM/IEEE, 2008]. Dies beinhaltet die Organisation von Digitalrechnern und deren Risiken und Einsatz in besonderen Umgebungen im Unterschied zu Desktop-Systemen [GI, 2011]. Zur Spezifikation dieser Anforderungen sind verschiedene Spezifikationstechniken miteinander zu vergleichen und die geeignete auszuwählen [GI, 2011].

K2.3: Systemdesign Beim Systemdesign wird unter Berücksichtigung der zuvor abgeleiteten funktionalen und nicht-funktionalen Anforderungen mittels formaler Methoden das System unter Verwendung domänenspezifischer Entwicklungswerkzeuge modelliert [FAU Erlangen-Nürnberg, 2011]. Studierende müssen in der Lage sein die wechselseitigen Einflüsse zu analysieren und einen

Kompromiss zu treffen. Dazu sind die Kenntnisse verschiedener Technologien und Konzepte eingebetteter Systeme notwendig, welche beim Entwurf digitaler Schaltungen, Software- und Hardwarearchitekturen anzuwenden sind [TH Karlsruhe, 2011], [RWTH Aachen, 2011].

K2.4: Implementierung Bei gegebenem Design sind Kompetenzen zur Überführung der Modelle in eine funktionierende Hard- und Software erforderlich. Abhängig von der verwendeten Architektur sind Komponenten des Systems z. B. in einer Programmiersprache in Software, in einer Hardwarebeschreibungssprache oder auf niederer Abstraktionsebene auf Schaltungsebene und Boolescher Algebra zu implementieren. Hier sind Konzepte der maschinennahen Programmierung wie Assembler und Grundschaltungen der Elektronik zu verstehen [Universität Siegen, 2011], [TU München, 2011], [TH Karlsruhe, 2011].

K2.5: Optimierung und Test Zur Optimierung und Test des entwickelten Systems müssen die Studierenden die Schaltungen ausmessen und simulieren können [Universität Siegen, 2011]. Ferner müssen sie eingebettete Systeme nach unterschiedlichen Kriterien beurteilen (Größe, Stromverbrauch, Echtzeitfähigkeit, Stückpreis etc.) [FAU Erlangen-Nürnberg, 2011] [TH Karlsruhe, 2011].

K3: Ebenen übergreifende Kompetenzen

Ebenen übergreifendes Entwickeln meint die Fähigkeiten und Fertigkeiten die notwendig sind, den Entwicklungsprozess als interdisziplinär und zusammenwirken von Konzepten unterschiedlicher Abstraktionsebene zu begreifen. Eine ganzheitliche Entwicklung von eingebetteten Systemen auf nur einer Ebene kann nicht zielführend sein.

K3.1: Top-Down Das Top-Down-Vorgehen ist in der Informatik vorherrschend. Eingebettete Systeme zeichnen sich durch ihre speziellen Einsatzgebiete aus. Top-Down ist dann vorzugehen, wenn nicht die Entwicklung dedizierter Hardware im Vordergrund steht und auf Basisbauteile und existierende Plattformen zurückgegriffen werden kann. Dann müssen die Studierenden die Funktionsweise dieser Plattformen (Digitalrechner) kennen und verstehen, wie die Entwicklungen auf den höheren Abstraktionsebenen sich auf die darunterliegenden auswirken (Software → Hardware) [TU München, 2011], [Universität Siegen, 2011].

K3.2: Bottom-Up Basisbauteile und Funktionsgruppen sowie nanoskaline Strukturen werden Bottom-Up entwickelt. Zusammenhänge zwischen Elektrotechnik und Informatik werden hier besonders deutlich, da Hardwarekonzepte Auswirkungen auf die Software haben [TH Karlsruhe, 2011].

K3.3: Jojo Der Jojo-Ansatz beinhaltet hier auch das Meet-in-the-Middle Vorge-

hen, in welchen keine Entwurfsrichtung vorherrschend ist und sowohl Bottom-Up als auch Top-Down vorgegangen wird. Damit werden diese verknüpft, was insbesondere Kompetenzen auf einer Metaebene bedingt, in denen eine Entwurfsrichtung anhand der gegebenen Problemstellung gewählt und angewandt wird.

K4: Nicht-kognitive Kompetenzen

Die Strukturierung der nicht-kognitiven Kompetenzen wurde aus dem Projekt MoKoM übernommen [Nelles et al., 2010].

K4.1: Einstellungen Die Einstellungen umfassen die Wahrnehmung eines eingebetteten Systems in seiner Umwelt mit den verbundenen Erwartungshaltungen sowie die Fähigkeiten die besonderen qualitativen Anforderungen einzuschätzen und in der Entwicklung mit adäquater Sorgfalt zu berücksichtigen [Nelles et al., 2010] [RWTH Aachen, 2011].

K4.2: Sozial-kommunikativ Sozial-kommunikative Fähigkeiten sowie Kooperationsbereitschaft sind wichtig, um in Kleingruppen, Teams und Projekten gemeinsam zu arbeiten, was die Kommunikation im beruflichen Umfeld mit Ingenieuren und Elektrotechnikern einschließt [Universität Siegen, 2011] [RWTH Aachen, 2011]. Empathie, also die Fähigkeit die Perspektive zu wechseln, soll Entwickler befähigen, das eingebettete System und seine Entwicklung aus Sicht des Benutzers, Entwicklers und Auftraggebers zu betrachten [Nelles et al., 2010].

K4.3: Motivational und Volitional Das facettenreiche Fachgebiet erfordert aufgrund der sich schnell ändernden Technologien ständige Anpassung der persönlichen Einstellungen, Fähigkeiten und Fertigkeiten. Dazu ist Offenheit für neue Ideen und Anforderungen sowie Lernbereitschaft und Bereitschaft zum Engagement zu fördern [Nelles et al., 2010]. Wissen muss immer wieder in komplexen Handlungssituationen neu geschaffen und erfolgreich angewandt werden.

A.2. Blockdiagramm des Prozessors XMEGA A1

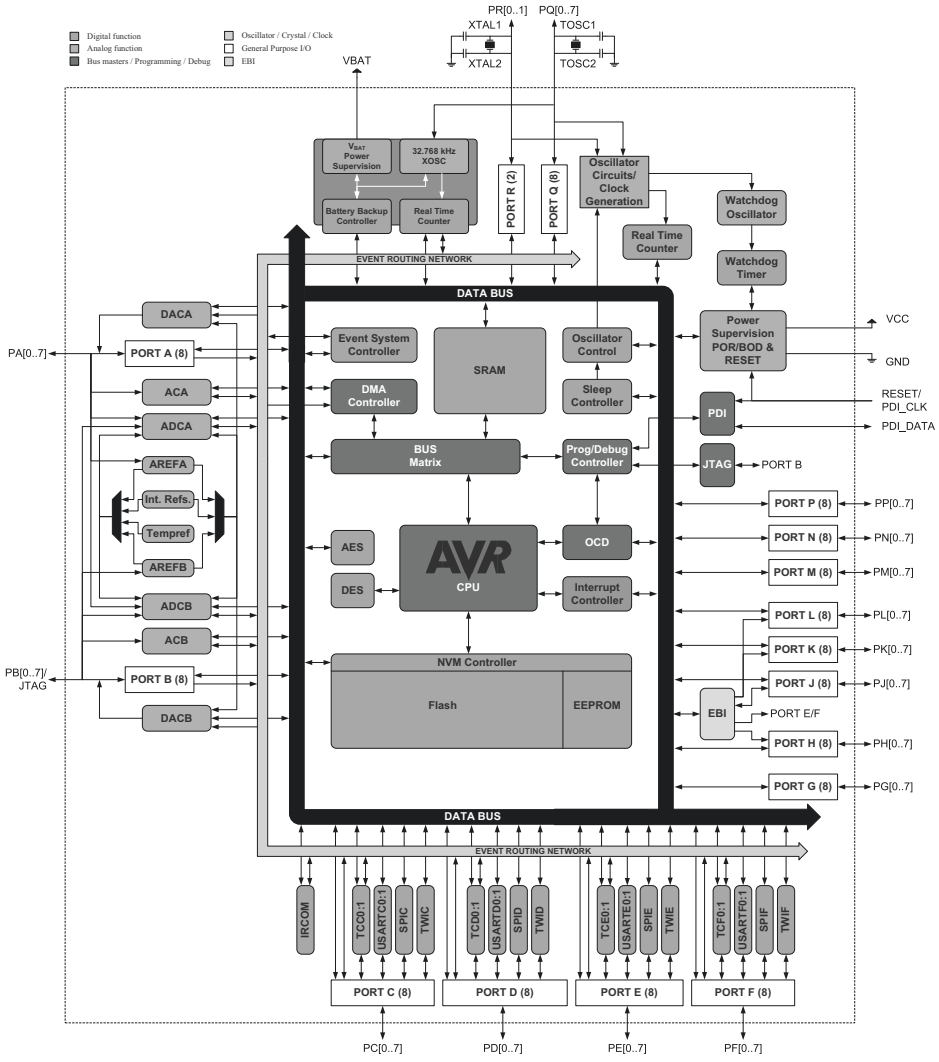


Abbildung A.1.: Blockdiagramm des Prozessors XMEGA A1 [Atmel Corporation, 2013]

A.3. Expertenbefragung – Verfeinerung des NKSM

Bewerten Sie folgende Kompetenzen (ein Kreuz/Zeile) mit:
a) besonders wichtig b) eher wichtig c) eher unwichtig d) besonders unwichtig

K1	Kompetenzen als Voraussetzung (Basis)	a)	b)	c)	d)
Die Studierenden bringen Vorkenntnisse mit aus den Bereichen					
K1.1	Mathematik	☐	☐	☐	☐
K1.2	Physik	☐	☐	☐	☐
K1.3	Informatik	☐	☐	☐	☐
	Sie besitzen Kenntnisse der relevanten Speichertechnologien	☐	☐	☐	☐
	Sie verstehen die verschiedenen Darstellungsformen von Zahlen und Alphabeten in Rechnern	☐	☐	☐	☐
	Sie besitzen fundierte theoretische und praxisorientierte Kenntnisse über die Grundlagen der Verarbeitung von Daten mit Hilfe von Rechnern	☐	☐	☐	☐
	Sie kennen die Prinzipien einschlägiger Basistechnologien im Bereich der Hardware und Systemsoftware und können sie einsetzen	☐	☐	☐	☐
K1.4	Elektrotechnik	☐	☐	☐	☐
K1.5	Materialwissenschaften	☐	☐	☐	☐
K1.6	Englisch	☐	☐	☐	☐
K1.7	Wissenschaftliches Arbeiten	☐	☐	☐	☐
K1.8	Lernorganisation	☐	☐	☐	☐

K2	Entwicklungskompetenzen (EMNS)	a)	b)	c)	d)
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K2.1	Organisation des Entwicklungsprozesses				
	Sie können selbstständige und kreative Lösungen vorgegebener Aufgaben gestalten	☐	☐	☐	☐
	Sie können Projekte planen, durchführen und dokumentieren	☐	☐	☐	☐
	Sie erlernen zielgerichtete und strukturierte Vorgehensweisen beim Entwurf	☐	☐	☐	☐
K2.2	Anforderungsanalyse				
	Sie kennen die besonderen Randbedingungen des Entwurfs eingebetteter Systeme	☐	☐	☐	☐
	Sie kennen die Verbindung und Organisation von Komponenten in Digitalrechnern	☐	☐	☐	☐
	Sie sind in der Lage, Risiken und Zuverlässigkeit in besonderen Bereichen beurteilen zu können	☐	☐	☐	☐
	Sie können Spezifikationstechniken für Eingebettete- und Realzeitsysteme miteinander vergleichen und geeignete Techniken auswählen	☐	☐	☐	☐
K2.3	Systementwurf				
	Sie sind in der Lage, eigene Schaltungen zu entwickeln	☐	☐	☐	☐
	Sie erwerben elementare Kenntnisse und grundlegendes Verständnis über die wichtigsten Technologien und über die wichtigsten Konzepte, die beim Entwurf und der Analyse von rechnergestützten Systemen benötigt werden	☐	☐	☐	☐
	Sie sind in der Lage, unbekannte Schaltungen zu analysieren und zu verstehen	☐	☐	☐	☐
	Sie verstehen den Aufbau und die Funktion aller wichtigen Grundschaltungen und Rechenwerke	☐	☐	☐	☐
	Sie haben ein Verständnis von Rechnersystemen als geschichtete abstrakte Maschinen	☐	☐	☐	☐
	Sie erwerben fundierte Kenntnisse für den Entwurf Eingebetteter Systeme unter Einsatz formaler Methoden und rechnergestützter Entwurfsverfahren	☐	☐	☐	☐
K2.4	Realisierung				
	Sie können einen Prozessor mit einem „Field Programmable Gate Array (FPGA)“ in Hardware umsetzen	☐	☐	☐	☐
	Sie kennen und beherrschen moderne Softwaretechnik für Eingebettete Systeme	☐	☐	☐	☐
	Sie können die Schaltalgebra als mathematisches Modell, die Registertransfersprachen zur Beschreibung von Steuerwerken und die Programmiermethodik auf der Mikroprogrammebene anwenden	☐	☐	☐	☐
	Sie können Schaltungen mittels einer Beschreibungssprache entwerfen	☐	☐	☐	☐
	Sie erwerben fundierte Kenntnisse über die Implementierung eingebetteter Systeme unter Einsatz formaler Methoden und rechnergestützter Entwurfsverfahren	☐	☐	☐	☐
	They appreciate the concept of an instruction set architecture, ISA, and the nature of a machine-level instruction in terms of its functionality and use of resources (registers and	☐	☐	☐	☐

Abbildung A.2.: Befragung Universitätsprofessoren Seite 1

	memory)	
K2.5	Optimierung und Test	
	Sie können wesentliche Bauelemente und einfache Grundschaltungen der Elektronik ausmessen und simulieren	☐ ☐ ☐ ☐
	Sie können eigene Hardware nach unterschiedlichen Kriterien optimieren	☐ ☐ ☐ ☐
	Sie sind in der Lage, Eingebettete Systeme nach unterschiedlichen Kriterien zu beurteilen	☐ ☐ ☐ ☐

K3	**Kompetenz zur ebenen übergreifenden Entwicklung von EMNS**	**a)**	**b)**	**c)**	**d)**
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K3.1	Top-Down				
	Sie verstehen den Maschinenbefehlszyklus auf Basis der Vorgänge in der Hardware auf Registertransferebene	☐	☐	☐	☐
	Sie erlernen den Umgang mit der Programmiersprache C und deren Zusammenspiel mit der Hardware. Dabei werden grundlegende Funktionalitäten und das Zusammenspiel der Basiskomponenten eines Betriebssystems mit dem Schwerpunkt auf effiziente Ressourcenverwaltung vermittelt	☐	☐	☐	☐
K3.2	Bottom-Up				
	Sie sind in der Lage, den Zusammenhang zwischen Hardware-Konzepten und den Auswirkungen auf die Software zu verstehen, um effiziente Programme erstellen und anwenden zu können sowie einen Rechner aus Grundkomponenten aufbauen zu können	☐	☐	☐	☐
	Sie sind in der Lage, die durch Nanotechnologien entstehenden Probleme bei der Zuverlässigkeit nanoelektronischer Bauelemente zu verstehen	☐	☐	☐	☐
	Sie können selbst-organisierende Prinzipien, z.B. Selbst-Adaptivität durch Rekonfiguierung, verstehen und anwenden, um nanoelektronische Systeme zuverlässiger zu machen	☐	☐	☐	☐
	Sie verstehen die physikalischen Grundprinzipien neuer Basis-Nanobauelemente für die IT, wie Nanoröhrchen, Nanocrossbars oder Quantenzellularautomaten	☐	☐	☐	☐
K3.3	Jojo-Design				
	Sie erkennen die Zusammenhänge der Grundlagen der Elektrotechnik mit Anwendungen der Informatik	☐	☐	☐	☐

K4	**Nicht-kognitive Kompetenzen**	**a)**	**b)**	**c)**	**d)**
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K4.1	Einstellung				
	Sie besitzen Wahrnehmung / Antizipation des EMNS im Kontext	☐	☐	☐	☐
	Sie besitzen Sensibilität für besondere qualitative Anforderungen	☐	☐	☐	☐
K4.2	Sozial-kommunikativ				
	Sie arbeiten in Kleingruppen, Tutorengruppen und Teamarbeit in Projektgruppen	☐	☐	☐	☐
	Sie können im beruflichen Umfeld mit Ingenieuren und Elektrotechnikern als Informatikanwender kompetent kommunizieren	☐	☐	☐	☐
	Sie besitzen Empathie (Perspektivwechsel: Benutzer, Entwickler, Auftraggeber)	☐	☐	☐	☐
K4.3	Motivational und volitional				
	Sie besitzen die Offenheit für neue Ideen / Anforderungen	☐	☐	☐	☐
	Sie zeigen Lernbereitschaft	☐	☐	☐	☐
	Sie zeigen Bereitschaft zum Engagement	☐	☐	☐	☐

Kommentare zu den aufgeführten Kompetenzen bzw. Ergänzung fehlender Kompetenzen

Abbildung A.3.: Befragung Universitätsprofessoren Seite 2

Bewerten Sie folgende Kompetenzen (ein Kreuz/Zeile) mit:
a) besonders wichtig b) eher wichtig c) eher unwichtig d) besonders unwichtig

K1	Kompetenzen als Voraussetzung (Basis)	a)	b)	c)	d)
Die Studierenden bringen Vorkenntnisse mit aus den Bereichen					
K1.1	Mathematik	☐	☐	☐	☐
K1.2	Physik	☐	☐	☐	☐
K1.3	Informatik	☐	☐	☐	☐
K1.4	Elektrotechnik	☐	☐	☐	☐
K1.5	Materialwissenschaften (insbesondere der atomare Aufbau)	☐	☐	☐	☐
K1.6	Englisch	☐	☐	☐	☐
K1.7	Wissenschaftliches Arbeiten	☐	☐	☐	☐
K1.8	Lernorganisation	☐	☐	☐	☐

K2	Entwicklungskompetenzen (EMNS)	a)	b)	c)	d)
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K2.1	Organisation des Entwicklungsprozesses				
	Sie können selbstständige und kreative Lösungen vorgegebener Aufgaben gestalten	☐	☐	☐	☐
	Sie können Projekte planen, durchführen und dokumentieren	☐	☐	☐	☐
	Sie erlernen zielgerichtete und strukturierte Vorgehensweisen beim Entwurf	☐	☐	☐	☐
K2.2	Anforderungsanalyse				
	Sie kennen die besonderen Randbedingungen des Entwurfs eingebetteter Systeme	☐	☐	☐	☐
	Sie kennen die Verbindung und Organisation von Komponenten in Digitalrechnern	☐	☐	☐	☐
	Discuss the concept of parallel processing and the relationship between parallelism and performance	☐	☐	☐	☐
	Sie sind in der Lage, Risiken und Zuverlässigkeit in besonderen Bereichen beurteilen zu können	☐	☐	☐	☐
	Understand how performance can be increased by incorporating multiple processors on a single chip	☐	☐	☐	☐
	Sie können Spezifikationstechniken für Eingebettete- und Realzeitsysteme miteinander vergleichen und geeignete Techniken auswählen	☐	☐	☐	☐
K2.3	Systementwurf				
	Sie sind in der Lage, eigene Schaltungen zu entwickeln	☐	☐	☐	☐
	Sie erwerben elementare Kenntnisse und grundlegendes Verständnis über die wichtigsten Technologien und über die wichtigsten Konzepte, die beim Entwurf und der Analyse von rechnergestützten Systemen benötigt werden	☐	☐	☐	☐
	Sie sind in der Lage, unbekannte Schaltungen zu analysieren und zu verstehen	☐	☐	☐	☐
	Sie verstehen den Aufbau und die Funktion aller wichtigen Grundschaltungen und Rechenwerke	☐	☐	☐	☐
	Sie haben ein Verständnis von Rechnersystemen als geschichtete abstrakte Maschinen	☐	☐	☐	☐
	Sie erwerben fundierte Kenntnisse für den Entwurf Eingebetteter Systeme unter Einsatz formaler Methoden und rechnergestützter Entwurfsverfahren	☐	☐	☐	☐
K2.4	Realisierung				
	Sie können einen Prozessor mit einem „Field Programmable Gate Array (FPGA)“ in Hardware umsetzen	☐	☐	☐	☐
	Sie kennen und beherrschen moderne Softwaretechnik für Eingebettete Systeme	☐	☐	☐	☐
	Sie können die Schaltalgebra als mathematisches Modell, die Registertransfersprachen zur Beschreibung von Steuerwerken und die Programmiermethodik auf der Mikroprogrammebene anwenden	☐	☐	☐	☐
	Sie können Schaltungen mittels einer Beschreibungssprache entwerfen	☐	☐	☐	☐
	Sie erwerben fundierte Kenntnisse über die Implementierung eingebetteter Systeme unter Einsatz formaler Methoden und rechnergestützter Entwurfsverfahren	☐	☐	☐	☐
	They appreciate the concept of an instruction set architecture, ISA, and the nature of a machine-level instruction in terms of its functionality and use of resources (registers and memory)	☐	☐	☐	☐

Abbildung A.4.: Befragung Fachhochschulprofessoren Seite 1

K2.5	Optimierung und Test	
	Sie können wesentliche Bauelemente und einfache Grundschaltungen der Elektronik ausmessen und simulieren	☐ ☐ ☐ ☐
	Sie können eigene Hardware nach unterschiedlichen Kriterien optimieren	☐ ☐ ☐ ☐
	Sie sind in der Lage, Eingebettete Systeme nach unterschiedlichen Kriterien zu beurteilen	☐ ☐ ☐ ☐

K3	**Kompetenz zur ebenen übergreifenden Entwicklung von EMNS**	**a)**	**b)**	**c)**	**d)**
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K3.1	Top-Down				
	Sie verstehen den Maschinenbefehlszyklus auf Basis der Vorgänge in der Hardware auf Registertransferebene	☐	☐	☐	☐
	Sie erlernen den Umgang mit der Programmiersprache C und deren Zusammenspiel mit der Hardware. Dabei werden grundlegende Funktionalitäten und das Zusammenspiel der Basiskomponenten eines Betriebssystems mit dem Schwerpunkt auf effiziente Ressourcenverwaltung vermittelt	☐	☐	☐	☐
K3.2	Bottom-Up				
	Sie sind in der Lage, den Zusammenhang zwischen Hardware-Konzepten und den Auswirkungen auf die Software zu verstehen, um effiziente Programme erstellen und anwenden zu können sowie einen Rechner aus Grundkomponenten aufbauen zu können	☐	☐	☐	☐
	Sie sind in der Lage, die durch Nanotechnologien entstehenden Probleme bei der Zuverlässigkeit nanoelektronischer Bauelemente zu verstehen	☐	☐	☐	☐
	Sie können selbst-organisierende Prinzipien, z.B. Selbst-Adaptivität durch Rekonfiguierung, verstehen und anwenden, um nanoelektronische Systeme zuverlässiger zu machen	☐	☐	☐	☐
	Sie verstehen die physikalischen Grundprinzipien neuer Basis-Nanobauelemente für die IT, wie Nanoröhrchen, Nanocrossbars oder Quantenzellularautomaten	☐	☐	☐	☐
K3.3	Jojo-Design				
	Sie erkennen die Zusammenhänge der Grundlagen der Elektrotechnik mit Anwendungen der Informatik	☐	☐	☐	☐

K4	**Nicht-kognitive Kompetenzen**	**a)**	**b)**	**c)**	**d)**
Die Studierenden erlangen Kompetenzen in folgenden Bereichen					
K4.1	Einstellung				
	Sie besitzen Wahrnehmung / Antizipation des EMNS im Kontext	☐	☐	☐	☐
	Sie besitzen Sensibilität für besondere qualitative Anforderungen	☐	☐	☐	☐
K4.2	Sozial-kommunikativ				
	Sie arbeiten in Kleingruppen, Tutorengruppen und Teamarbeit in Projektgruppen	☐	☐	☐	☐
	Sie können im beruflichen Umfeld mit Ingenieuren und Elektrotechnikern als Informatikanwender kompetent kommunizieren	☐	☐	☐	☐
	Sie besitzen Empathie (Perspektivwechsel: Benutzer, Entwickler, Auftraggeber)	☐	☐	☐	☐
K4.3	Motivational und volitional				
	Sie besitzen die Offenheit für neue Ideen / Anforderungen	☐	☐	☐	☐
	Sie zeigen Lernbereitschaft	☐	☐	☐	☐
	Sie zeigen Bereitschaft zum Engagement	☐	☐	☐	☐

Kommentare zu den aufgeführten Kompetenzen bzw. Ergänzung fehlender Kompetenzen

Abbildung A.5.: Befragung Fachhochschulprofessoren Seite 2

A.4. Befragung von Erstsemesterstudierenden

DFG-Projekt „Kompetenzentwicklung mit Eingebetteten Mikro- und Nanosystemen“
Befragung von Erstsemesterstudierenden zu Eingebetteten Mikro- und Nanosystemen

1.) Welches Anwendungsfach/Nebenfach haben Sie gewählt? _____

Haben Sie zuvor eine informationstechnische Ausbildung abgeschlossen? Ja ☐ Nein ☐

2.) Woher kennen Sie Informatik?

☐	☐	☐	☐	☐
Grundkurs (3 Std.)	Grundkurs (2 Std.)	Leistungskurs	andere Quelle	ohne

3.) Welche der folgenden programmierbaren Hardwaresysteme oder Geräte kennen Sie?

	Unbekannt	Kenne ich	Nutze ich	Besitze ich
Lego Mindstorms NXT	☐	☐	☐	☐
Fischertechnik Robo Mobile Kit	☐	☐	☐	☐
Mikrocontroller (z.B. Atmel AVR)	☐	☐	☐	☐
FPGAs (z.B. Xilininx, Altera, Lattice, …)	☐	☐	☐	☐

4.) Welche der folgenden Programmiersprachen kennen Sie?

	Unbekannt	Kenne ich	Nutze ich	Beherrsche ich
C (oder C++, …)	☐	☐	☐	☐
JAVA	☐	☐	☐	☐
VHDL (oder Verilog)	☐	☐	☐	☐
BASIC (Visual Basic, …)	☐	☐	☐	☐
LOGO	☐	☐	☐	☐
Prolog	☐	☐	☐	☐

Definition:
Eingebettete Systeme (ES) sind dedizierte, intelligente Systeme, bestehend aus Software-, Hardware- und Vernetzungskomponenten zur reaktiven Steuerung analoger, peripherer technischer Prozesse oder Umgebungen, in die sie eingebettet werden.

5.) Bewerten Sie folgende Aussagen:

	Trifft zu	Trifft nicht zu
Wussten Sie schon was Eingebettete Systeme (ES) sind?	☐	☐
Benutzen Sie bewusst und täglich ES?	☐	☐
Kommen Sie täglich mit ES in Kontakt?	☐	☐
- Wenn zutreffend, nennen Sie bitte ein/zwei dieser ES!	_____	_____
Wurden ES bereits in der Schule behandelt?	☐	☐
Besitzt ihre Schule programmierbare ES?	☐	☐
Habe Sie bereits ES programmiert/entwickelt?	☐	☐

Definition:
Mikrosysteme bestehen aus Strukturen/Komponenten in einer Größenordnung von 0,1 µm - 100 µm (1 µm = 0,000.001 m). Nanosysteme um Faktor 1000 geringer (0,0001 µm - 0,1 µm).

6.) Ordnen Sie folgende Produkte in keine, eine oder mehrere Kategorien ein:

	Eingebettete Systeme	Nanotechnik	Mikrotechnik
Lego Mindstorm NXT	☐	☐	☐
Handy	☐	☐	☐
3-Fach Steckdose	☐	☐	☐
Spielekonsole	☐	☐	☐
Waschmaschine	☐	☐	☐

7.) Nennen Sie bis zu drei (noch nicht genannte) Produkte, die Sie der Nanotechnik zuordnen können.

_____ _____ _____

Wir bedanken uns für ihre Zeit zur Beantwortung des Fragebogens und wünschen Ihnen ein erfolgreiches Studium!

Abbildung A.6.: Befragung Erstsemesterstudierende (Sommersemester 2011 - Universität Siegen)

DFG-Projekt „Kompetenzentwicklung mit eingebetteten Mikro- und Nanosystemen“

Befragung von Erstsemesterstudierenden zu eingebetteten Mikro- und Nanosystemen

1. Welchen Studiengang haben Sie gewählt?

☐ Informatik ☐ anderen: _____

2. Haben Sie zuvor eine informationstechnische Ausbildung abgeschlossen?

☐ Ja ☐ Nein

3. Haben Sie Vorkenntnisse in Informatik?

☐	☐	☐	☐
ja, aus Grundkurs	ja, aus Leistungskurs	ja, aus anderer Quelle	nein, keine Vorkenntnisse

4. Welche der folgenden programmierbaren Hardware-Systeme oder Geräte kennen Sie?

	unbekannt	kenne ich	nutze ich	besitze ich
Lego Mindstorm NXT	☐	☐	☐	☐
Fischertechnik Robo Mobile Kit	☐	☐	☐	☐
Mikrocontroller (z.B. Atmel AVR)	☐	☐	☐	☐
FPGA (z.B. Xilinx, Altera)	☐	☐	☐	☐

5. Welche der folgenden Programmiersprachen kennen Sie?

	unbekannt	kenne ich	nutze ich	beherrsche ich
Basic (z.B. Visual Basic)	☐	☐	☐	☐
Java	☐	☐	☐	☐
C/C++	☐	☐	☐	☐
VHDL (oder Verilog)	☐	☐	☐	☐
LOGO	☐	☐	☐	☐
Prolog	☐	☐	☐	☐

6. Eingebettete Mikro- und Nanosysteme

Eingebettete Systeme (ES) sind digitale Rechner, die weitgehend unsichtbar für den Benutzer ihren Dienst in vielen Anwendungsbereichen und Geräten verrichten, z.B. die Steuerung der Airbags in Kraftfahrzeugen. Hergestellt werden ES zur Zeit mit „klassischen“ Chips auf Halbleiterbasis. Die zunehmende Miniaturisierung führt zu Nanosystemen mit Strukturen in Größenordnungen weniger Moleküle.

	ja	nein
Wussten Sie schon vorher was ES sind?	☐	☐
Benutzen Sie bewusst und täglich ES?	☐	☐
Kommen Sie täglich mit ES in Kontakt?	☐	☐
Wenn ja, nennen Sie bitte ein/zwei dieser ES!	1. _____	2. _____
Wurden ES bereits in Ihrer Schule behandelt?	☐	☐
Besitzt Ihre Schule programmierbare ES?	☐	☐
Haben Sie bereits ES programmiert/entwickelt?	☐	☐

7. Sind die folgenden Produkte eingebettete Systeme? Sind sie in Mikro- und/oder Nanotechnik realisiert?

	eingebettetes System	Mikrotechnik	Nanotechnik
Lego Mindstorm NXT	☐	☐	☐
Handy	☐	☐	☐
Dreifachsteckdose	☐	☐	☐
Spielekonsole	☐	☐	☐
Waschmaschine	☐	☐	☐

8. Nennen Sie bis zu drei noch nicht genannte Produkte, die Sie der Nanotechnik zuordnen!

_____ _____ _____

Wir bedanken uns für Ihre Zeit zur Beantwortung des Fragebogens und wünschen Ihnen ein erfolgreiches Studium!

Abbildung A.7.: Befragung Erstsemesterstudierende (Sommersemester 2011 - Universität Erlangen-Nürnberg)

Literaturverzeichnis

[acatech, 2011] acatech (2011). *Cyber-physical systems: Innovationsmotor für Mobilität, Gesundheit, Energie und Produktion*. Deutsche Akademie der Technikwissenschaften, Acatech Position. Springer Verlag, Berlin, Heidelberg.

[ACM/AIS/AITP, 2002] ACM/AIS/AITP (2002). *IS2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems*. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2001] ACM/IEEE (2001). *Computing Curricula 2001: Computer Science*. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2004a] ACM/IEEE (2004a). *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering: A Report in the Computing Curricula Series*. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2004b] ACM/IEEE (2004b). *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2005] ACM/IEEE (2005). *Computing Curricula 2005: The Overview Report*. Computing Curricula Series. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2008] ACM/IEEE (2008). *Computer Science Curriculum 2008: An Interim Revision of CS 2001*. Association for Computing Machinery (ACM), New York, USA.

[ACM/IEEE, 2013] ACM/IEEE (2013). *Computer Science Curriculum 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery (ACM), New York, USA.

[AK DQR, 2013] AK DQR (2013). Deutscher Qualifikationsrahmen für lebenslanges Lernen (DQR). http://www.dqr.de/media/content/Der_Deutsche_Qualifikationsrahmen_fue_lebenslanges_Lernen.pdf, (Abruf: 12/2014).

[Anderson et al., 2000] Anderson, Lorin W. ; Krathwohl, David R. ; Airasian, Peter W. ; Cruikshank, Kathleen A. ; Mayer, Richard E. ; Pintrich, Paul R. ; Raths, James und Wittrock, Merlin C. (2000). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Ab-*

ridged Edition. Allyn & Bacon, 2. Auflage.

[Artist Education Group, 2003] Artist Education Group (2003). Guidelines for a Graduate Curiculum on Embedded Software and Systems. http://www.artist-embedded.org/docs/Publications/Education.pdf, (Abruf: 12/2014).

[Atmel Corporation, 2013] Atmel Corporation (2013). 8-bit Atmel XMEGA A Microcontroller: XMEGA A MANUAL. http://www.atmel.com/Images/doc8077.pdf, (Abruf: 12/2014).

[Ausubel et al., 1980] Ausubel, David P. ; Novak, Joseph D. ; Hanesian, Helen und Vontin, Walther (1980). *Psychologie des Unterrichts*. Beltz, Weinheim [u.a.], 2. Auflage.

[Baumann, 1993] Baumann, Rüdiger (1993). Ziele und Inhalte des Informatikunterrichts. *Zentralblatt für Didaktik der Mathematik*, 1993 (Nr. 25): Seiten 9–19.

[Baumann, 1996] Baumann, Rüdiger (1996). *Didaktik der Informatik*. Ernst Klett Verlag GmbH, Stuttgart, 2. Auflage.

[Bender, 2005] Bender, Klaus (2005). *Embedded Systems: Qualitätsorientierte Entwickung*. Springer, Berlin, 1. Auflage.

[Berns et al., 2010] Berns, Karsten ; Schürmann, Bernd und Trapp, Mario (2010). *Eingebettete Systeme - Systemgrundlagen und Entwicklung eingebetteter Software*. Vieweg+Teubner Verlag.

[Bibliographisches Institut GmbH, 2014] Bibliographisches Institut GmbH (2014). Duden. http://www.duden.de/, (Abruf: 12/2014).

[BITKOM, 2010] BITKOM (2010). *Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland: Anwendungsbeispiele, Zahlen und Trends*. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V., Berlin.

[Blömeke und Zlatkin-Troitschanskaia, 2013] Blömeke, Sigrid und Zlatkin-Troitschanskaia, Olga (2013). *Kompetenzmodellierung und Kompetenzerfassung im Hochschulsektor: Ziele, theoretischer Rahmen, Design und Herausforderungen des BMBF-Forschungsprogramms KoKoHs: (KoKoHs Working Papers, 1)*. Humboldt-Universität & Johannes Gutenberg-Universität, Berlin, Mainz.

[Bortz und Döring, 2005] Bortz, Jürgen und Döring, Nicola (2005). *Forschungsmethoden und Evaluation: Für Human- und Sozialwissenschaftler ; mit 70 Tabellen*. Springer-Lehrbuch. Springer, Heidelberg, 3. Auflage.

[Bouldin, 2004] Bouldin, Don (2004). Impacting education using FPGAs. In: *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, Seiten 142–148, Santa Fe, New Mexico.

[Bower, 2008] Bower, Matt (2008). A Taxonomy of Task Types in Computing.

In: *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, Seiten 281–285, New York, USA. Association for Computing Machinery (ACM).

[Brand et al., 2011] Brand, Emily A. ; Honig, William L. und Wojtowicz, Matthew (2011). Intelligent Systems Development in a Non Engineering Curriculum. In: *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, Seiten 48–52, New York, USA. Association for Computing Machinery (ACM).

[Brejcha et al., 2011] Brejcha, Philipp ; Beneder, Roman und Kramer, Michael (2011). New approaches for a distance learning course about embedded systems. In: *Global Engineering Education Conference (EDUCON), 2011 IEEE*, Seiten 903–906, Amman, Jordan.

[Brinda, 2001] Brinda, Torsten (2001). Einfluss fachwissenschaftlicher Erkenntnisse zum objektorientierten Modellieren auf die Gestaltung von Konzepten in der Didaktik der Informatik. In: Keil-Slawik, Reinhard und Magenheim, Johannes (Hrsg.), *Informatikunterricht und Medienbildung*, Band 8 aus *Lecture Notes in Informatics*, Seiten 75–86, Bonn. Gesellschaft für Informatik e.V. (GI).

[Brinda, 2004] Brinda, Torsten (2004). *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II.* Dissertation, Universität Siegen, Siegen.

[Brück et al., 2008] Brück, Rainer ; Freischlad, Stefan und Schmidt, Tilo (2008). Ein innovatives Konzept für ein Entwurfs- und Anwendungspraktikum Mikrosysteme. In: Schubert, Sigrid (Hrsg.), *Bildungskonzepte für Internetworking und eingebettete Mikrosysteme*, Band 8 aus *Reihe Medienwissenschaften*, Seiten 167–182. Universitätsverlag Siegen, Siegen.

[Büchner, 2014] Büchner, Steffen (2014). Empirical and Normative Research on Fundamental Ideas of Embedded System Development. In: *Key Competencies in Informatics and ICT: In Druck*, Seiten 244–245, Potsdam, Deutschland.

[Büchner und Jaschke, 2013] Büchner, Steffen und Jaschke, Steffen (2013). Preparation for Embedded Systems Laboratories The Virtual Workspace Approach. In: IEEE (Hrsg.), *IEEE Global Engineering Education Conference*, Seiten 171–175, Berlin.

[Büchner et al., 2013] Büchner, Steffen ; Jaschke, Steffen und Schubert, Sigrid (2013). Enhancing the Comparability between Didactic Research on Embedded Systems. In: Marwedel, Peter ; Jackson, Jeff und Ricks, Kenneth G. (Hrsg.), *Workshop on Embedded and Cyber-Physical Systems Education: [in Druck].* ACM, Montreal, Kanada.

[Büchner und Schubert, 2014] Büchner, Steffen und Schubert, Sigrid (2014). Criteria-based Research on Fundamentals of Embedded System Development

in Higher Education. In: *The 2014 International Conference on Embedded Systems and Applications (ESA'14): In Druck*, Las Vegas, USA.

[Caspi et al., 2005] Caspi, P. ; Sangiovanni-Vincentelli, A. ; Almeida, L. ; Benveniste, A. ; Bouyssounouse, B. ; Buttazzo, G. ; Crnkovic, I. ; Damm, W. ; Engblom, J. ; Folher, G. ; Garcia-Valls, M. ; Kopetz, H. ; Lakhnech, Y. ; Laroussinie, F. ; Lavagno, L. ; Lipari, G. ; Maraninchi, F. ; Peti, Ph ; La Puente, J. de ; Scaife, N. ; Sifakis, J. ; Simone, R. de ; Torngren, Martin ; Verıssimo, P. ; Wellings, A. J. ; Wilhelm, R. ; Willemse, T. und Yi, W. (2005). Guidelines for a graduate curriculum on embedded software and systems. *ACM Transactions on Embedded Computing Systems*, 4 (Nr. 3): Seiten 587–611.

[Claus und Schwill, 2006] Claus, Volker und Schwill, Andreas (2006). *Duden Informatik A–Z. Fachlexikon für Studium und Praxis.* Dudenverlag, Mannheim [u.a.], 4. Auflage.

[Council of Europe, 2001] Council of Europe (2001). *Common European framework of reference for languages: Learning, teaching, assessment.* Press Syndicate of the University of Cambridge, Cambridge, England.

[Cranach und Frentz, 1969] Cranach, Mario und Frentz, Hans Georg (1969). *Systematische Beobachtung*, Band 7 aus *Handbuch der Psychologie.* Hogrefe, Göttingen.

[DFG, 2006] DFG (2006). Exzellenzinitiative des Bundes und der Länder zur Förderung von Wissenschaft und Forschung an deutschen Hochschulen: Ausschreibung. http://www.dfg.de/download/pdf/dfg_im_profil/reden_stellungnahmen/2006/exin_0610_pressemappe/exin_0610_wr_ausschr_1.pdf, (Abruf: 12/2014).

[Dörner, 1979] Dörner, Dietrich (1979). *Problemlösen als Informationsverarbeitung.* Kohlhammer-Standards Psychologie. Kohlhammer, Stuttgart, 2. Auflage.

[Edelmann, 2000] Edelmann, Walter (2000). *Lernpsychologie.* Beltz, Weinheim [u.a.], 6. Auflage.

[Edelmann und Wittmann, 2012] Edelmann, Walter und Wittmann, Simone (2012). *Lernpsychologie: Mit Online-Materialien.* Beltz, Weinheim [u.a.], 7. Auflage.

[EU, 2008] EU (2008). *Der Europäische Qualifikationsrahmen für lebenslanges Lernen (EQR).* Allgemeine & berufliche Bildung. Amt für Amtliche Veröffentlichungen der Europäischen Gemeinschaften, Luxemburg.

[FAU Erlangen-Nürnberg, 2011] FAU Erlangen-Nürnberg (2011). Studienführer. http://www.studium.informatik.uni-erlangen.de/studium/Studienfuehrer_Ba_informatik.pdf, (Abruf: 01/2011).

[Feisel und Rosa, 2005] Feisel, Lyle D. und Rosa, Albert J. (2005). The Role of the

Laboratory in Undergraduate Engineering Education. *Journal of Engineering Education*, 94 (Nr. 1): Seiten 121–130.

[Freischlad, 2010] Freischlad, Stefan (2010). *Entwicklung und Erprobung des Didaktischen Systems Internetworking im Informatikunterricht.* Dissertation, Universität Siegen, Siegen.

[Friedrich, 2008] Friedrich, Jan (2008). *Das V-Modell XT: Für Projektleiter und QS-Verantwortliche ; kompakt und übersichtlich.* Informatik im Fokus. Springer, Berlin und Heidelberg.

[Fuller et al., 2007] Fuller, Ursula ; Riedesel, Charles ; Thompson, Errol ; Johnson, Colin G. ; Ahoniemi, Tuukka ; Cukierman, Diana ; Hernán-Losada, Isidoro ; Jackova, Jana ; Lahtinen, Essi ; Lewis, Tracy L. und Thompson, Donna McGee (2007). Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin*, 39 (Nr. 4): Seiten 152–170.

[Gajski, 1994] Gajski, Daniel D. (1994). *Specification and design of embedded systems.* PTR Prentice Hall, Englewood Cliffs, USA.

[Gajski und Kuhn, 1983] Gajski, Daniel D. und Kuhn, R.H (1983). Introduction: New VLSI Tools. *Computer Science*, 16 (Nr. 12): Seiten 11–14.

[GI, 2005] GI (2005). *Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen: Empfehlung der Gesellschaft für Informatik e.V., Neuauflage der GI-Standards zur Akkreditierung von Informatik-Studiengängen aus dem Jahr 2000.* Gesellschaft für Informatik e.V. (GI), Bonn.

[GI, 2011] GI (Hrsg.) (2011). *Curriculum Technische Informatik in Bachelor- und Masterstudiengängen Informatik: Empfehlung der Gesellschaft für Informatik e. V.* Gesellschaft für Informatik e.V. (GI), Bonn.

[Greiner et al., 2009] Greiner, Felix ; Tschulena, Guido und Korb, Winfried (2009). Mikro-Nano-Integration - Einsatz von Nanotechnologie in der Mikrosystemtechnik. In: *Schriftenreihe der Aktionslinie Hessen-Nanotech des Hessischen Ministeriums für Wirtschaft Verkehr und Landesentwicklung*, Band 13. HA Hessen Agentur GmbH, Wiesbaden.

[Grimheden und Torngren, 2005] Grimheden, Martin und Torngren, Martin (2005). What is Embedded Systems and How Should It Be Taught? —results from a Didactic Analysis. *ACM Transactions on Embedded Computing Systems*, 4 (Nr. 3): Seiten 633–651.

[Hahn et al., 2004] Hahn, Kai ; Popp, Jens und Wagener, Andreas (2004). Process management and design for MEMS and microelectronics technologies. In: Abbott, Derek ; Eshraghian, Kamran ; Musca, Charles A. ; Pavlidis, Dimitris und Weste, Neil (Hrsg.), *Microelectronics: Design, Technology, and Packaging*, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series,

Seiten 322–330, Perth, Australien. SPIEE Press.

[Hampel et al., 1999] Hampel, Thorsten ; Magenheim, Johannes und Schulte, Carsten (1999). Dekonstruktion von Informatiksystemen als Unterrichtsmethode - Zugang zu objektorientierten Sichtweisen im Informatikunterricht. In: Schwill, Andreas (Hrsg.), *Fachspezifische und fachübergreifende didaktische Konzepte, INFOS 1999, 8. GI-Fachtagung Informatik und Schule, 22.-25. September 1999 in Potsdam*, Band 32. Gesellschaft für Informatik e.V. (GI).

[Hertwig und Brück, 2000] Hertwig, Andre und Brück, Rainer (2000). *Entwurf digitaler Systeme: von den Grundlagen zum Prozessorentwurf mit FPGAs*. Hanser, München.

[Hinostroza et al., 2000] Hinostroza, Enrique ; Rehbein, Lucioe ; Mellar, Harvey und Preston, Christina (2000). Developing educational software: a professional tool perspective. *Education and Information Technologies*, 5 (Nr. 2): Seiten 103–117.

[Informatik, 2004] Informatik, Fakultätentag (2004). Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten.

[ISO, 2010] ISO (2010). *Nanotechnologies - Terminology and definitions for nano-objects - Nanoparticle, nanofibre and nanoplate: DIN ISO/TS 27687:2010-2*. International Organization for Standardization.

[Jank und Meyer, 2005] Jank, Werner und Meyer, Hilbert (2005). *Didaktische Modelle*. Cornelson, Berlin.

[Jaschke, 2013] Jaschke, Steffen (2013). Towards a Didactic System for Embedded System Design in Higher Education. In: IFIP TC3 (Hrsg.), *Tenth IFIP World Conference on Computers in Education*, Seiten 284–293, Torun, Polen.

[Jaschke, 2014a] Jaschke, Steffen (2014a). Mobile Learning Applications for Technical Vocational and Engineering Education. In: *17th International Conference on Interactive Collaborative Learning: In Druck*. Dubai, Vereinigte Arabische Emirate.

[Jaschke, 2014b] Jaschke, Steffen (2014b). Mobile Lernapplikationen in cyberphysischen Arbeitsprozessen der Industrie 4.0. In: *Tagungsband der GTW Herbstkonferenz 2014, Bildung und Arbeitswelt, [in Druck]*, Aachen. Lit-Verlag.

[Jaschke und Büchner, 2013] Jaschke, Steffen und Büchner, Steffen (2013). Explorative Learning and Visualization Environment for Teaching Embedded Systems in Higher Education. In: *IEEE International Conference on Teaching, Assessment and Learning for Engineering*. Kuta, Indonesien.

[Jaschke et al., 2012] Jaschke, Steffen ; Büchner, Steffen ; Schubert, Sigrid ; Schäfer, André und Brück, Rainer (2012). Competence Oriented Embedded Systems

Course For Computer Science Students. In: Marwedel, Peter ; Jackson, Jeff und Ricks, Kenneth G. (Hrsg.), *Workshop on Embedded and Cyber-Physical Systems Education: [in Druck]*. ACM, Tampere, Finnland.

[Jaschke et al., 2011] Jaschke, Steffen ; Schubert, Sigrid ; Schäfer, André ; Brück, Rainer ; Kleinert, Bruno ; Schmidt, Harald und Fey, Dietmar (2011). Competence Research: Teaching Embedded Micro/Nano Systems. In: Jackson, Jeff ; Marwedel, Peter und Ricks, Kenneth G. (Hrsg.), *Workshop on Embedded Systems Education*, Taipei, Taiwan. ACM.

[Kassner und Ricks, 2005] Kassner, Kevin C. und Ricks, Kenneth G. (2005). Hardware/Software Co-design of Embedded Real-time Systems from an Undergraduate Perspective. In: *Proceedings of the 2005 Workshop on Computer Architecture Education: Held in Conjunction with the 32Nd International Symposium on Computer Architecture*, WCAE '05, New York, USA. Association for Computing Machinery (ACM).

[Kerres, 2000] Kerres, Michael (2000). Computerunterstütztes Lernen als Element hybrider Lernarrangements. In: Kammerl, Rudolf (Hrsg.), *Computerunterstütztes Lernen*, Seiten 23–39. Oldenbourg Wissenschaftsverlag, München.

[Kleinert et al., 2012] Kleinert, Bruno ; Schmidt, Harald ; Fey, Dietmar ; Jaschke, Steffen ; Büchner, Steffen ; Schubert, Sigrid ; Schäfer, André und Brück, Rainer (2012). Approaches to teaching future computation technologies and nanosystems. In: *Proceedings of IFIP-Conference*, IFIP, Manchester, England.

[Klieme, 2004] Klieme, Eckhard (2004). Was sind Kompetenzen und wie lassen sie sich messen? *Pädagogik*, (Nr. 6): Seiten 10–13.

[Klieme et al., 2007] Klieme, Eckhard ; Avenarius, Hermann ; Blum, Werner ; Döbrich, Peter ; Gruber, Hans ; Prenzel, Manfred ; Reiss, Kristina ; Riquarts, Kurt ; Rost, Jürgen ; Tenorth, Heinz-Elmer und Vollmer, Helmut J. (2007). Bildungsforschung Band 1 Expertise Zur Entwicklung nationaler Bildungsstandards.

[Klieme und Leutner, 2006] Klieme, Eckhard und Leutner, Detlev (2006). Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen. Beschreibung eines neu eingrichteten Schwerpunktprogramms der DFG. *Zeitschrift für Pädagogik*, 52 (Nr. 6): Seiten 876–903.

[KMK, 2012] KMK (2012). Bildungsstandards für die fortgeführte Fremdsprache (Englisch / Französisch) für die Allgemeine Hochschulreife. http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2012/2012_10_18-Bildungsstandards-Fortgef-FS-Abi.pdf, (Abruf: 12/2014).

[Lee, 2008] Lee, Edward Ashford (2008). Cyber Physical Systems: Design Challenges. In: *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, Seiten 363–369, Orlando, USA. IEEE Computer Society.

[Lee und Seshia, 2010] Lee, Edward Ashford und Seshia, Anjit Arunkumar (2010). An Introductory Textbook on Cyber-Physical Systems. In: Marwedel, Peter ; Jackson, Jeff und Ricks, Kenneth G. (Hrsg.), *Proceedings of the 5th Workshop on Embedded Systems Education*, WESE '10, New York, USA. ACM.

[Lee und Seshia, 2012] Lee, Edward Ashford und Seshia, Anjit Arunkumar (2012). *Introduction to embedded systems: A Cyber-Physical Systems-Approach*. LeeSeshia.org, Berkeley, USA, 1. Auflage.

[Lehner et al., 2010] Lehner, Leopold ; Magenheim, Johannes ; Nelles, Wolfgang ; Rhode, Thomas ; Schaper, Niclas ; Schubert, Sigrid und Stechert, Peer (2010). Informatics Systems and Modelling – Case Studies of Expert Interviews. In: Reynolds, Nicholas und Turcsányi-Szabó, Márta (Hrsg.), *Key Competencies in the Knowledge Society*, Band 324 aus *IFIP Advances in Information and Communication Technology*, Seiten 222–233. Springer, Berlin und Heidelberg.

[Magenheim, 2003] Magenheim, Johannes (2003). Informatik Lernlabor - Systemorientierte Didaktik in der Praxis. In: Hubwieser, Peter (Hrsg.), *Informatische Fachkonzepte im Unterricht, INFOS 2003, 10. GI-Fachtagung Informatik und Schule, 17.-19. September 2003 in Garching bei München*, Band 32 aus *LNI*. Gesellschaft für Informatik e.V. (GI).

[Magenheim et al., 2010] Magenheim, Johannes ; Nelles, Wolfgang ; Rhode, Thomas ; Schaper, Niclas ; Schubert, Sigrid und Stechert, Peer (2010). Competencies for informatics systems and modeling: Results of qualitative content analysis of expert interviews. In: *Education Engineering (EDUCON), 2010 IEEE*, Seiten 513–521, Madrid, Spanien.

[Magenheim et al., 2013] Magenheim, Johannes ; Neugebauer, Jonas ; Stechert, Peer ; Ohrndorf, Laura ; Linck, Barbara ; Schubert, Sigrid ; Nelles, Wolfgang und Schaper, Niclas (2013). Competence Measurement and Informatics Standards in Secondary Education. In: Diethelm, Ira und Mittermeir, Roland T. (Hrsg.), *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages Informatics in Schools – 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 2013) in Oldenburg*, Band 7780 aus *Lecture Notes in Computer Science*, Seiten 159–170. Springer, Berlin.

[Marwedel, 2005] Marwedel, Peter (2005). Towards laying common grounds for embedded system design education. *SIGBED Rev.*, 2005 (Nr. 4): Seiten 25–28.

[Marwedel, 2011] Marwedel, Peter (2011). *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Embedded Systems. Springer Science und Business Media B.V, Dordrecht.

[Marwedel und Engel, 2011] Marwedel, Peter und Engel, Michael (2011). Embedded system design 2.0: rationale behind a textbook revision. In: Jackson, Jeff ; Marwedel, Peter und Ricks, Kenneth G. (Hrsg.), *Workshop on Embedded Sys-*

tems Education, Taipei, Taiwan. ACM.

[Mayring, 2010] Mayring, Philipp (2010). *Qualitative Inhaltsanalyse: Grundlagen und Techniken*. Beltz Pädagogik. Beltz, Weinheim, 11. Auflage.

[Mitsui et al., 2009] Mitsui, Hiroyasu ; Kambe, Hidetoshi und Koizumi, Hisao (2009). Use of Student Experiments for Teaching Embedded Software Development Including HW/SW Co-Design. *IEEE Transactions on Education*, 52 (Nr. 3): Seiten 436–443.

[Muppala, 2011] Muppala, Jogesh K. (2011). Teaching embedded software concepts using Android. In: *Proceedings of the 6th Workshop on Embedded Systems Education*, WESE '11, Seiten 32–37, New York, USA. Association for Computing Machinery (ACM).

[Myrach und Montandon, 2007] Myrach, Thomas und Montandon, Corinne (2007). Blended Learning. In: Thom, Norbert und Zaugg, Robert J. (Hrsg.), *Moderne Personalentwicklung*, Seiten 189–204. Gabler, Wiesbaden.

[Nelles et al., 2010] Nelles, Wolfgang ; Rhode, Thomas und Stechert, Peer (2010). Entwicklung eines Kompetenzrahmenmodells – Informatisches Modellieren und Systemverständnis. *Informatik Spektrum*, 33 (Nr. 1): Seiten 45–53.

[Nooshabadi und Garside, 2006] Nooshabadi, Saeid und Garside, Jim (2006). Modernization of Teaching in Embedded Systems Design, An International Collaborative Project. *IEEE Transactions on Education*, 49 (Nr. 2): Seiten 254–262.

[NRW, 2014] NRW (2014). *Gesetz über die Hochschulen des Landes Nordrhein-Westfalen: HG*.

[OECD, 2005] OECD (20.07.2005). *Definition und Auswahl von Schlüsselkompetenzen: Zusammenfassung*. Organisation for Economic Co-operation and Development (OECD).

[Paetz et al., 2011] Paetz, Nadja-Verena ; Ceylan, Firat ; Fiehn, Janina ; Schworm, Silke und Harteis, Christian (2011). *Kompetenz in der Hochschuldidaktik: Ergebnisse einer Delphi-Studie über die Zukunft der Hochschullehre*. VS Verlag für Sozialwissenschaften, Wiesbaden.

[Philips Semiconductors, 2000] Philips Semiconductors (2000). KTY81-2 series Silicon temperature sensors. http://www.nxp.com/documents/data_sheet/KTY81_SER.pdf, (Abruf: 12/2014).

[Rajkumar et al., 2010] Rajkumar, Ragunathan ; Insup Lee ; Lui Sha und Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. In: *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, Seiten 731–736, Anaheim, USA.

[Reichenbach et al., 2011] Reichenbach, Marc ; Schmidt, Michael ; Pfundt, Benja-

min und Fey, Dietmar (2011). A New Virtual Hardware Laboratory for Remote FPGA Experiments on Real Hardware. In: Arabnia, Hamid R. ; Deligiannidis, Leonidas ; Solo, Ashu M.G. und Bahrami, Azita (Hrsg.), *Proceedings of the 2011 International Conference on E-Learning, E-Business, Enterprise Information Systems, & E-Goverment*, Seiten 17–23, Las Vegas, USA.

[RWTH Aachen, 2011] RWTH Aachen (2011). Vorlesungsverzeichnis. https://www.campus.rwth-aachen.de/, (Abruf: 01/2011).

[Sarik und Kymissis, 2010] Sarik, J. und Kymissis, I. (2010). Lab kits using the Arduino prototyping platform. In: *Frontiers in Education Conference (FIE), 2010 IEEE*, Seiten T3C–1–T3C–5, Washington, USA.

[Schäfer und Brück, 2013] Schäfer, André und Brück, Rainer (2013). Teaching strategies for undergraduate laboratories with students having heterogeneous prior knowledge. In: *Global Engineering Education Conference (EDUCON), 2013 IEEE*, Seiten 112–117, Berlin.

[Schäfer et al., 2012a] Schäfer, André ; Brück, Rainer ; Büchner, Steffen ; Jaschke, Steffen ; Schubert, Sigrid ; Fey, Dietmar ; Kleinert, Bruno und Schmidt, Harald (2012a). The empirically refined competence structure model for embedded micro- and nanosystems. In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, ITiCSE '12, Seiten 57–62, New York, USA. Association for Computing Machinery (ACM).

[Schäfer et al., 2011] Schäfer, André ; Brück, Rainer ; Jaschke, Steffen ; Schubert, Sigrid ; Fey, Dietmar ; Kleinert, Bruno und Schmidt, Harald (2011). A normative competence structure model for embedded micro- and nanosystems development. In: *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, ITiCSE '11, Seite 375, New York, USA. Association for Computing Machinery (ACM).

[Schäfer et al., 2012b] Schäfer, André ; Brück, Rainer ; Kleinert, Bruno ; Fey, Dietmar ; Büchner, Steffen ; Jaschke, Steffen und Schubert, Sigrid (2012b). Competence model for embedded micro-and nanosystems. In: *Collaborative Learning & New Pedagogic Approaches in Engineering Education*, Marrakesch, Marokko. IEEE.

[Schäfer et al., 2012c] Schäfer, André ; Jaschke, Steffen und Büchner, Steffen (2012c). Skript zum Entwurfs- und Anwendungspraktikum.

[Schaumont, 2008] Schaumont, Patrick (2008). Hardware/Software Co-design is a starting point in Embedded Systems Architecture Education. In: Jackson, Jeff ; Marwedel, Peter und Ricks, Kenneth G. (Hrsg.), *Proceedings of the 3th Workshop on Embedded Systems Education*, WESE '08, Atlanta, USA. ACM.

[Schmidt, 2011] Schmidt, Tilo (2011). *Technologiemanagement und anwendungsspezifische Prozessentwicklung in der Mikrosystemtechnik*. Dissertation, Univer-

sität Siegen, Siegen.

[Schubert et al., 2010] Schubert, Sigrid ; Brück, Rainer und Fey, Dietmar (2010). Antrag auf Gewährung einer Sachbeihilfe - Projekt: Kompetenzentwicklung mit Eingebetteten Mikro- und Nanosystemen (KOMINA). http://www.die.informatik.uni-siegen.de/KOMINA/Komina-Antrag.pdf, (Abruf: 12/2014).

[Schubert und Schwill, 2011] Schubert, Sigrid und Schwill, Andreas (2011). *Didaktik der Informatik*. Spektrum Akademischer Verlag, Heidelberg, 2. Auflage.

[Schwidrowski, 2010] Schwidrowski, Kirstin (2010). *Konzeption und Diskussion zu E-Learning für Internetworking in der beruflichen Bildung unter Weiterentwicklung des Didaktischen Systems*. Dissertation, Universität Siegen.

[Spath, 2013] Spath, Dieter (Hrsg.) (2013). *Produktionsarbeit der Zukunft - Industrie 4.0*. Fraunhofer-Verlag, Stuttgart.

[Stechert, 2009] Stechert, Peer (2009). *Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht*. Dissertation, Universität Siegen, Siegen.

[Stechert und Schubert, 2007] Stechert, Peer und Schubert, Sigrid (2007). A strategy to structure the learning process towards understanding of informatics systems. In: Berglund, Anders (Hrsg.), *Proceedings of the 6th Baltic Sea Conference on Computing Education Research*, Seiten 128–131, Koli Calling, Finnland. Association for Computing Machinery (ACM).

[Sztipanovits et al., 2005] Sztipanovits, Janos ; Biswas, Gautam ; Frampton, Ken ; Gokhale, Aniruddha ; Howard, Larry ; Karsai, Gabor ; Koo, T. John ; Koutsoukos, Xenofon und Schmidt, Douglas C. (2005). Introducing Embedded Software and Systems Education and Advanced Learning Technology in an Engineering Curriculum. *ACM Transactions on Embedded Computing Systems*, 4 (Nr. 3): Seiten 549–568.

[TH Karlsruhe, 2011] TH Karlsruhe (2011). Modulhandbuch. http://www.informatik.kit.edu/downloads/studium/mhb_info_bsc_ws1011_de_lang.pdf, (Abruf: 12/2014).

[TU München, 2011] TU München (2011). Modulkatalog. https://drehscheibe.in.tum.de/myintum/kurs_verwaltung/cmall.html, (Abruf: 01/2011).

[Universität Siegen, 2011] Universität Siegen (2011). Modulhandbuch. http://pi.informatik.uni-siegen.de/akkred/module/FB12/, (Abruf: 12/2014).

[Wagener, 2005] Wagener, Andreas (2005). *Fertigungsnahe Entwurfsunterstützung für die Mikrosystemtechnik*. Dissertation, Universität Siegen, Siegen.

[Walker und Thomas, 1985] Walker, Robert A. und Thomas, Donald E. (1985). A Model of Design Representation and Synthesis. In: *Proceedings of the 22Nd*

ACM/IEEE Design Automation Conference, DAC '85, Seiten 453–459, Piscataway, USA. IEEE Press.

[Wei-Feng et al., 2009] Wei-Feng, Yin ; Rong-Gao, Sun und Zhong, Wan (2009). Distributed Remote Laboratory Using Web Services for Embedded System. In: *Proceedings of the 3rd WSEAS International Conference on Circuits, Systems, Signal and Telecommunications*, CISST'09, Seiten 56–59, Stevens Point, USA. World Scientific and Engineering Academy and Society (WSEAS).

[Weicker, 2007] Weicker, Nicole (2007). Zielorientierte Didaktik der Informatik - Kompetenzvermittlung bei engen Zeitvorgaben. In: Schubert, Sigrid (Hrsg.), *Didaktik der Informatik in Theorie und Praxis. INFOS 2007: 12. GI-Fachtagung Informatik und Schule, 19.-21. September 2007 in Siegen*, Band 112 aus *LNI*, Seiten 337–348, Siegen. Gesellschaft für Informatik e.V. (GI).

[Weinert, 2001] Weinert, Franz E. (2001). *Leistungsmessungen in Schulen: Dr. nach Typoskript.* Beltz-Pädagogik. Beltz, Weinheim [u.a.].

[Wild und Möller, 2009] Wild, Elke und Möller, Jens (2009). *Pädagogische Psychologie.* Springer-Lehrbuch. Springer Verlag, Berlin und Heidelberg.