technische universität
dortmund

Ansätze zur Behandlung des Entwurfs eingebetteter Systeme im Informatikunterricht

André Wrede

Algorithm Engineering Report
TR08-3-005
Dez. 2008
ISSN 1864-4503

Fakultät für Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung für das Lehramt an Gymnasien und Gesamtschulen

Ansätze zur Behandlung des Entwurfs eingebetteter Systeme im Informatikunterricht

dem Staatlichen Prüfungsamt Dortmund vorgelegt von

Wrede, André
Zur Hünenburg 61
59823 Arnsberg

Dortmund, 22. September, 2008

Themensteller:
Prof. Dr. Jan Vahrenhold
Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.uni-dortmund.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In dieser schriftlichen Hausarbeit im Rahmen der Ersten Staatsprüfung für das Lehramt an Gymnasien und Gesamtschulen werden verschiedene Ansätze der universitären Lehre des Entwurfs eingebetteter Systeme miteinander verglichen und auf der Basis eines geeigneten Ansatzes der Versuch unternommen, eine in sich abgeschlossene und konsistente Menge von Lerninhalten herzuleiten, die im Rahmen des Informatikunterrichts behandelt werden könnte. Bevor die Struktur dieser Arbeit näher erläutert und begründet wird, sollen zunächst noch einige Ausführungen zur Themenstellung vorgenommen werden. Im Zuge dieser Ausführungen wird die Themenstellung präzisiert, die Relevanz der Arbeit erörtert sowie eine Formulierung der Ziele vorgenommen.

1.1 Ausführungen zur Themenstellung

Der Themenkomplex des Entwurfs eingebetteter Systeme ist nicht für jede Jahrgangsstufe und jede Schulart aufgrund von Voraussetzungen, die die Schülerinnen und Schüler teilweise in der Schule nicht erwerben, uneingeschränkt zu empfehlen. Daher bedarf es einer Einschränkung hinsichtlich der Zielgruppe. Die Untersuchung muss aber auch den Rahmen dieser Arbeit berücksichtigen, so dass auch aus formalen Gründen die Zielgruppe eingeschränkt werden muss.

Im Anschluss daran wird die Relevanz des Themas diskutiert. Es ist nicht offensichtlich, dass im Rahmen einer Staatsarbeit die Möglichkeit untersucht werden soll, Teile des Themenkomplexes der eingebetteten Systeme im Unterricht unterzubringen. Auch aufgrund der Uneinigkeit auf universitärer Ebene, was die Lehre eingebetteter Systeme anbelangt, könnte man geneigt sein, zu dem voreiligen Schluss zu gelangen, dass dieses Thema zu kompliziert und irrelevant für die Schule sei.

Nachdem geklärt wurde, warum diese Arbeit eine Legitimation besitzt, erfolgt die Darstellung dessen, was im Zuge dieser Arbeit erreicht werden soll und was nicht erreicht werden kann. Im Kontext der Gesamtdiskussion um eingebettete Systeme kann diese Arbeit nur einen kleinen Versuch darstellen, die Lehre in diesem Bereich zu verbessern. Daher sind - auch aufgrund der Tatsache, dass eine komplette Vorlesung analysiert werden wird und dies nur punktuell auf einer detaillierten Ebene vonstatten gehen kann - klar formulierte und begrenzte Ziele notwendig. Ebenso wichtig ist die Erläuterung dessen, was im Rahmen dieser Arbeit nicht geleistet werden kann. Hierbei steht vor allem der Unterschied zu einer Entwicklung eines Curriculums im Vordergrund, in dem mehr abverlangt wird, als lediglich eine begründete Aneinanderreihung von Unterrichtsthemen.

1.1.1 Einschränkung des Themas

Das Thema dieser Arbeit ist dahingehend recht allgemein gehalten, als dass von einem nicht näher spezifizierten Informatikunterricht gesprochen wird. Doch muss man sich bewusst werden, dass Informatikunterricht in verschiedenen Schulstufen und verschiedenen Schulformen in verschiedenen Bundesländern stattfindet. Ein Grund, das Thema einzugrenzen, besteht darin, dass eine solche Untersuchung sich immer auch auf bereits bestehende curriculare Vorgaben stützen sollte, da hier die Anforderungen an die Lerninhalte bereits formuliert sind und im Idealfall einen roten Faden vorgeben, der in allen Bereichen des Curriculums ersichtlich wird. Da in Deutschland aber sechzehn verschiedene Bundesländer ihre eigenen Lehrpläne für das Fach Informatik erstellen, ist es nicht möglich, im Rahmen dieser Arbeit die Argumentationen auf alle Bundesländer auszuweiten. Daher beschränken sich die Argumentationen auf ein Bundesland. Dass hierbei die Wahl auf das Land Nordrhein-Westfalen fällt, liegt unter anderem auch daran, dass hier die Informatik in der Schule noch nicht den Stellenwert besitzt, den sie eigentlich innehaben müsste.[1] Diese Untersuchung könnte vor dem Hintergrund der Relevanz dieses Themas (siehe Abschnitt 1.1.2) einen Beitrag dazu leisten, die Informatik auch im schulischen Bereich mehr in den Mittelpunkt zu rücken.

Eine weitere Einschränkung, die vorgenommen werden soll, ist durch die teilweise sehr hohen Voraussetzungen an das Vorwissen der Schülerinnen und Schüler motiviert. Diese inhaltlichen Themen werden noch im Verlauf dieser Arbeit erläutert. Fakt ist, dass der Entwurf eingebetteter Systeme nur in der gymnasialen Oberstufe thematisiert werden sollte. Dass es aber prinzipiell möglich sein dürfte, dieses Thema in der Oberstufe zu verankern, liegt daran, dass durch den Entwurf eines eingebetteten Systems Inhalte und Methodiken vermittelt werden, die im Sinne einer wissenschaftspropädeutischen Ausbilung durchaus erwünscht sind. Hierzu gehören unter anderem fachlich gut vernetztes Grundlagenwissen, Teamarbeit und auch Potential zum fächerübergreifenden Lernen.

1.1.2 Relevanz des Themas

Um dieser Arbeit eine Legitimation zu verleihen, bedarf es einer Erläuterung der Relevanz des Themas. Betrachtet man sich allein die Zahlen, die der Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM) kommuniziert, wird deutlich, dass es sich bei eingebetteten Systemen um einen der wichtigsten Bereiche der verarbeitenden Industrie handelt:

[1] Da dies keine Arbeit über den Stellenwert der Informatik in der Gesellschaft ist, wird darauf verzichtet, den aktuellen Stand der öffentlichen Diskussion darzustellen.

> „Die verarbeitende Industrie erzielt nach einer Studie von Roland Berger Strategy Consultants im Auftrag des BITKOM rund 80 Prozent ihrer Wertschöpfung mit Produkten, die so genannte Embedded Systems enthalten.“ [BITKOM, 2008, S. 1]

Noch deutlicher wird die Relevanz dieses Bereiches durch die Betrachtung komplexer Prozessoren:

> „Wenn man [...] die Anzahl der momentan im Betrieb befindlichen komplexen Prozessoren betrachtet, so wurde geschätzt, dass mehr als 90% dieser Prozessoren in eingebetteten Systemen verwendet werden.“ [Marwedel, 2007, S. 8]

Nun könnte man sich die Frage stellen, warum gerade dieser Bereich Einzug in die gymnasiale Oberstufe finden sollte. Dass dieser Bereich wichtig ist und in Zukunft noch weiter wachsen wird, ist unstrittig:

> „[Es] wird auch der Anteil der Embedded Systems am Gesamtaufwand für Forschung und Entwicklung in Europa steigen: von 9 Prozent im Jahr 2003 auf geschätzte 14 Prozent im Jahr 2009.“ [BITKOM, 2008, S. 1-2]

Aufgrund dieser wachsenden Relevanz erscheint die Lage am deutschen Arbeitsmarkt (Stand: 2007) recht bedrohlich:

> „[Es] sind in der deutschen Wirtschaft insgesamt mindestens 43.000 offene ITK-Stellen vorhanden.“ [BITKOM, 2007, S. 5]

Die Konsequenzen, die sich durch diese offenen Stellen für die deutsche Wirtschaft ergeben, sollen hier nicht ausgeführt werden. Doch bleibt die Frage zu klären, warum sich gerade die Schule hiermit auseinandersetzen sollte. Die Schule hat die Möglichkeit, Schülerinnen und Schüler in einer frühen Phase ihrer Ausbildung für gewisse Themengebiete zu begeistern. Gelänge es durch die Thematisierung von eingebetteten Systemen in Form ihres Entwurfs schon in der Schule, junge Menschen für diesen Bereich zu faszinieren, wären bessere Voraussetzungen gegeben, diesen wichtigen Wachstumsmarkt mit Nachwuchskräften zu versorgen.

Es steht noch die Frage im Raum, warum diese Aufgabe dem Informatikunterricht und nicht beispielsweise der Physik zufallen oder sogar ein eigenes Fach eingeführt werden sollte. Schubert und Schwill [Schubert und Schwill, 2004] halten als eine wichtige Eigenschaft der Informatik fest:

> „[Die Informatik] ist [...] zu einer Grundlagenwissenschaft geworden, deren Methoden und Ergebnisse in nahezu allen anderen Wissenschaften verwendet werden.“ [Schubert und Schwill, 2004, S. 10]

Daher kann die Informatik wie kein anderes Fach einen besonderen Blickwinkel auf den Entwurf eingebetteter Systeme anbieten. Durch die Informatik bietet sich die

Möglichkeit, sich auf grundlegende Methoden zu beschränken und technische Details, die eher auf den Erwerb von Fertigkeiten abzielen, auszublenden. Die technische Umsetzung stünde beispielsweise im Physikunterricht zwangsweise im Vordergrund. Warum diese Konzentration auf grundlegende Methoden geschehen soll, wird unter dem Aspekt der Allgemeinbildung im Verlauf dieser Arbeit näher untersucht.

Aus all diesen Gründen ist es zumindest überdenkenswert, den Entwurf eingebetteter Systeme in der Schule unterrichtlich zu thematisieren. Diese Arbeit erhält hierdurch ihre Legitimation.

1.1.3 Ziele dieser Arbeit

Bisher ist die Ausbildung im Bereich der eingebetteten Systeme den Universitäten vorbehalten. Hierfür gibt es sicher sehr gute Gründe. Betrachtet man sich die Menge der Voraussetzung, die teilweise an Studierende gestellt werden, um sich im Bereich der eingebetteten Systeme vertiefen zu können, erscheint es zunächst utopisch, einen Versuch zu unternehmen, Elemente solcher Vorlesungen in der Schule umsetzen zu wollen. Doch reicht es nicht, einmal auf das Thema zu schauen und dann ein Urteil abzugeben, dass auf keiner Grundlage basiert. Daher ist es der Anspruch dieser Arbeit, Aspekte des Entwurfs eingebetteter Systeme zu betrachten und durch nachvollziehbare Argumentationsketten zu einem begründeten Urteil darüber zu kommen, welche Elemente des Entwurfs eingebetteter Systeme für den Schulunterricht im Sinne der Zielsetzung dieser Arbeit geeignet sind. Dabei steht im Fokus der Untersuchung ein noch zu spezifizierender (siehe dazu 3.1) ganzheitlicher Ansatz. Die Bewertung der Themenbereiche wird also im Hinblick auf ein eigenständiges Unterrichtsmodul ausgerichtet sein, unabhängig davon, ob ein gewisses Thema in einem anderen Kontext geeignet ist, unterrichtet zu werden.

Am Ende dieser Arbeit steht dann entweder eine begründet hergeleitete Menge von Unterrichtsinhalten, die untereinander verknüpft und an einem erkennbaren roten Faden aufgereiht sind oder die Erkenntnis, dass der Entwurf eingebetteter Systeme in der hier intendierten Form nicht umzusetzen ist. Beide Ergebnisse hätten zur Konsequenz, dass eine wissenschaftlich fundierte Aussage über die Frage, ob der Entwurf eingebetteter Systeme als eigenständiges Modul im Informatikunterricht behandelbar erscheint, möglich ist.

1.1.4 Unterschiede zur Curriculumentwicklung

Es ist ausdrücklich nicht das Ziel dieser Arbeit, ein fertiges und praxiserprobtes Modul vorzustellen. Dies könnte auch nicht geleistet werden, da im Rahmen dieser Arbeit, die drei Monate umfasst, kein Unterrichtsmodul über mindestens drei Monate

hergeleitet, erprobt und reflektiert werden kann. Daher sind manche Argumentationen notgedrungen auf Vergleiche zu anderen Themenbereichen, wie beispielsweise der Softwaretechnik, angewiesen. Die Arbeit erhält dadurch einen eher theoretischen Charakter, der aber aufgrund des oben genannten Grundes nicht zu vermeiden ist.

Die Entwicklung einer Unterrichtsreihe soll auch keinen Versuch darstellen, ein ausgereiftes Curriculum als Ergebnis zu haben. Baumann [Baumann, 1990] definiert den Begriff Curriculum wie folgt:

> „Unter einem Curriculum wird ein Lehrplan verstanden, in dem die Ziele des Unterrichts, seine Inhalte und deren Abfolge, sowie Hinweise auf die anzuwendenden Unterrichtsverfahren, die Medien und die Erfolgskontrollen angegeben sind.“ [Baumann, 1990, S. 27]

In dieser Arbeit werden allgemeine Ziele des Unterrichts dazu verwendet, ein Unterrichtsmodul über eingebettete Systeme zu konzipieren. Diese Ziele werden aber nicht auf allgemeiner oder konkreter Ebene wiederholt bzw. spezifiziert. Da diese Arbeit von einer schriftlichen Hausarbeit im Rahmen der zweiten Staatsprüfung abzugrenzen ist, wird auch darauf verzichtet, zu sehr den Fokus auf Medien und Methoden zu legen. Da der Schwerpunkt auf der wissenschaftlich fundierten und begründeten Auswahl von Unterrichtsinhalten liegt, kann diese Arbeit nicht den Anspruch erheben, ein Curriculum für den Entwurf eingebetteter Systeme darzustellen.

1.2 Aufbau der Arbeit

Die Arbeit gliedert sich in zwei inhaltliche Schwerpunkte. Um eine Basis für die weiteren Schlussfolgerungen zu schaffen, werden im ersten Schwerpunkt Ansätze verschiedener Hochschulen miteinander verglichen und hinsichtlich ihrer Eignung, zumindest in Teilen auch in der Schule umgesetzt werden zu können, bewertet. Dabei wird in dieser Phase die universitäre Lehre betrachtet, da kein Curriculum für eingebettete Systeme in der Schule existiert. Um nun so etwas wie ein Modul für eingebettete Systeme herzuleiten, ergibt es daher einen Sinn, die Diskussionen, die auf universitärem Niveau stattfinden, aufzugreifen und für die Gestaltung eines solchen Moduls zu nutzen.

Im Anschluss daran wird auf dieser Basis ein eigener Ansatz hergeleitet. Dieser muss natürlich berücksichtigen, dass nicht alle Aspekte eines universitären Ansatzes in der Schule Erwähnung finden können. Daher werden geeignete Mittel vorgestellt, wie man die Stofffülle geeignet reduzieren kann. Nach der Reduktion erfolgt die eigentliche Bewertung, ob die Intention dieser Arbeit wirklich erreichbar ist. Zum Schluss erfolgt die Reflexion, die bewertet, ob die Ziele dieser Arbeit erreicht wurden

und welcher Beitrag für die Gesamtdiskussion um den Informatikunterricht geleistet wurde.

Aus methodischer Sicht sollen in dieser Arbeit vor allem stets die verschiedenen Sichtweisen auf das entsprechende Thema aufgezeigt werden. Es gibt bei solch einer Arbeit nicht die ultimative Wahrheit. Es können lediglich Argumentationsketten konstruiert werden, bei denen auch immer eine gewisse Sichtweise mitklingt. So gibt es beispielsweise bei der Analyse, welche Lerninhalte geeignet erscheinen, die Sichtweise des Fachs und die Sichtweise der Fachdidaktik, die einige Argumente der fachlichen Sicht entkräften. Es existiert zudem auch immer allgemein die Sicht der Informatik auf das Thema und die Sicht von anderen Fächern. Es ist zentraler Bestandteil dieser Arbeit, dass diese Sichtweisen zumindest angerissen werden und somit zum besseren Verständnis der verschiedenen Argumentationsketten beigetragen werden kann.

2 Bestehende Ansätze in der Hochschullehre

In diesem Kapitel werden verschiedene bestehende Ansätze dafür behandelt, wie der Entwurf eingebetteter Systeme an Hochschulen gelehrt wird. Im Verlauf der Darstellung dieser Herangehensweisen wird deutlich werden, dass basierend auf verschiedenen Grundannahmen die Ausrichtung der Lehre auf ein bestimmtes Lernziel hin massiv divergiert. Es besteht also in der Fachwelt eine rege Diskussion über die Lehre des Entwurfs von eingebetteten Systemen. Daher soll in dieser Arbeit keinerlei Wertung vorgenommen werden, welcher Ansatz für die Hochschullehre der geeignetste ist oder welche Nachteile bestimmte Ansätze mit sich bringen. Die Motivation, sich mit den bestehenden Ansätzen in der Hochschullehre zu beschäftigen, ist, wie in der Einleitung erwähnt, Schlüsse ziehen zu können, welcher Ansatz am geeignetsten erscheint, in der Schule zumindest in Teilen umgesetzt werden zu können.

Zunächst wird der Ansatz der Königlich Technischen Hochschule in Stockholm (KTH) betrachtet. Grimheden [Grimheden und Törngren, 2005] stellt die Ergebnisse seiner didaktischen Analyse darüber vor, wie eingebettete Systeme gelehrt werden sollten. Einen anderen Ansatz präsentiert Sangiovanni-Vincentelli [Sangiovanni-Vincentelli und Pinto, 2005] von der *University of California at Berkeley*. Hier geht es vor allem um die Lehre im Masterstudiengang. Dort wird ein grundlegender Kurs (EECS249) präsentiert, dem weitere vertiefende Kurse folgen können. Zuletzt wird der in Dortmund von Marwedel [Marwedel, 2005] verfolgte Ansatz vorgestellt, bei dem es darauf ankommt, eine breite Übersicht über den Entwurf eingebetteter Systeme zu bieten.

Dabei werden die Lehrpläne gerade dieser Universitäten dargestellt, da sie einen Überblick vermitteln, welche verschiedenen Interpretationen der Lehre des Entwurfs eingebetteter Systeme in der hochschuldidaktischen Diskussion vertreten werden. Diese Auswahl an Ansätzen kann natürlich keinen Anspruch auf Vollständigkeit erheben, da in der konkreten Ausdifferenzierung der Lerninhalte jeder Dozierende eigene Schwerpunkte setzen kann und somit zu einem eigenen Ansatz kommt.

Nach der Darstellung der einzelnen Herangehensweisen an die Lehre des Entwurfs eingebetteter Systeme werden diese anhand verschiedener Kriterien miteinander verglichen. Dabei geht es einerseits um die Aspekte, die der Allgemeinbildung dienlich sind und andererseits um Unterschiede, wie beispielsweise das Selbstverständnis, das hinter den Ansätzen steckt.

Im Anschluss erfolgt die Diskussion darüber, welcher Ansatz nun am geeignetsten erscheint, um an einer allgemeinbildenden Schule als Leitidee umgesetzt zu werden. Dazu wird vor allem eine Definition von Allgemeinbildung verwendet, der die Ansätze gerecht werden sollten.

2.1 Ansatz der KTH in Schweden

Grimheden beschreibt in seinem Aufsatz *„How should embedded systems be taught? Experiences and snapshots from Swedish higher engineering education“* [Grimheden und Törngren, 2005] die Entwicklung des Stellenwertes eingebetteter Systeme an der KTH in Stockholm und präsentiert dann den didaktischen Ansatz, der geprägt sei von den Leitideen der exemplarischen Auswahl (*exemplifying selection*) von Lerninhalten und interaktiver Kommunikation (*interactive communication*) [Grimheden und Törngren, 2005, S. 34].

Den Stellenwert eingebetteter Systeme an der KTH beschreibt Grimheden als historisch gewachsen aus der Zugehörigkeit dieser Systeme - was die Lehre an der KTH anbelangt - zu den Ingenieurswissenschaften (genauer gesagt zum *„department of machine design, mechatronics lab, at the Royal Institute of Technology“* [Grimheden und Törngren, 2005, S. 34]). Daher werde das Fach an der KTH heutzutage im Studiengang Maschinenbau integriert. Dabei werde in der Ausbildung großer Wert darauf gelegt, dass sowohl die gelernten Inhalte als auch die erworbenen Fähigkeiten in der Industrie anwendbar sind:

> *„It is therefore of fundamental interest that the education provided by the universities is deemed relevant and useful by the hiring industry.“* [Grimheden und Törngren, 2005, S. 34-35]

Um dies zu erreichen, soll in der Ausbildung an der Universität eine stärkere Gewichtung der Problemorientierung durch Projekte stattfinden. Eine andere Möglichkeit seien Kurse, die sowohl von Studierenden als auch von Vertretern der Wirtschaft besucht werden, um so Synergieeffekte zu ermöglichen.

Ausgehend von diesen Anforderungen an die universitäre Lehre geht Grimheden den Fragen nach, warum und wie man eingebettete Systeme unterrichten und vor allem, welche Lerninhalte man vermitteln sollte. Zur Legitimation des Faches führt er an:

> *„The legitimacy of a subject is defined as the relation between the educational effort [. . .] and the demands put by the hiring industries on the graduated engineers.“* [Grimheden und Törngren, 2005, S. 35-36]

Es ergebe sich für die eingebetteten Systeme eine funktionale Legitimation, die sich im Erwerb von *„skills and abilities“* [Grimheden und Törngren, 2005, S. 36] ausdrücke, im Gegensatz zu einer formalen Legitimation, die durch den Erwerb von Creditpunkten in spezifischen Fächern charakterisiert sei. Dies sei auch durch die Wirtschaft gestützt:

> *„In Sweden, the hiring industry is primarily interested in functional skills, engineers capable of designing and implementing embedded systems.“* [Grimheden und Törngren, 2005, S. 36]

Die Auswahl der Lerninhalte sei dann eine logische Konsequenz aus der Legitimation. Es komme darauf an, eine exemplarische Auswahl zu schaffen, so dass ein Bereich vertieft studiert wird. Als Beispiel könne sich der Autor einen Kurs vorstellen, in dem ein Mikrocontroller intensiv behandelt wird, um so vom Speziellen auf das Allgemeine schließen zu können. Auch dieses Vorgehen bei der Auswahl der Lerninhalte sei durch die Wirtschaft gestützt:

> *„[...] the companies want engineers with in-depth knowledge and funktional skills rather than formal knowledge.“* [Grimheden und Törngren, 2005, S. 36]

Die Zielsetzung dieses Ansatzes, was Lerninhalte und die Art und Weise, wie diese Lehrinhalte vermittelt werden sollen, anbelangt, ist in Abb. 1 illustriert.

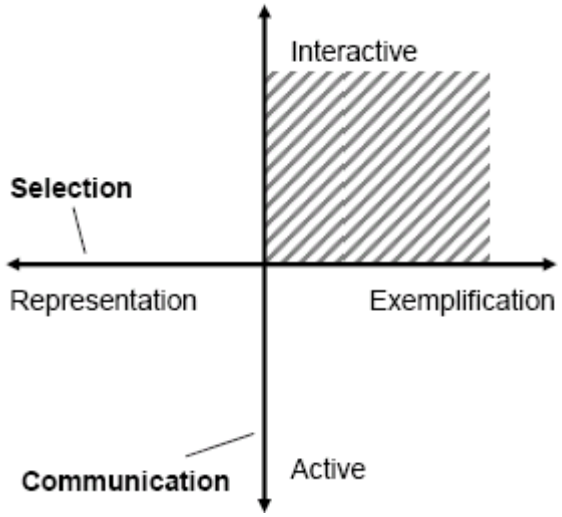

Abbildung 1: Zielsetzung des Ansatzes der KTH

Quelle: [Grimheden und Törngren, 2005, S. 36]

Die Abbildung verdeutlicht, dass die Lerninhalte so abgestimmt sein müssen, dass eine interaktive Kommunikation möglich ist. Daher wird Wert auf vertiefende Lerninhalte gesetzt.

2.2 Ansatz der *University of California at Berkeley*

Sangiovanni-Vincentelli beschreibt in dem Aufsatz *„Embedded system education: a new paradigm for engineering schools?“* [Sangiovanni-Vincentelli und Pinto, 2005]

den über zehn Jahre gereiften Lehrplan zu eingebetteten Systemen an der *University of California at Berkeley* und stellt in diesem Zuge entsprechende Kurse in den Bachelor- und Masterstudiengängen vor. Der Grundsatz in der Ausbildung bestehe dabei in der Verknüpfung der physikalischen Sichtweise eines Systems mit der Sichtweise der Informatik.

Im Bachelorstudiengang sei dafür insbesondere der Kurs „*Structure and interpretation of signals and systems (EECS20N)*“ vorgesehen. In diesem Kurs, der typischerweise im zweiten Studienjahr belegt werde, erfolge die Zusammenführung der verschiedenen Sichtweisen durch die Verbindung zwischen imperativer und deklarativer Beschreibung sowie der Behandlung von Zustandsautomaten und der Frequenzbereichsanalyse für den Entwurf und zur Analyse von Signalen und Systemen (siehe dazu die Literatur [Sangiovanni-Vincentelli und Pinto, 2005, S. 12]). Dass das Programm dieses Kurses so gestaltet ist, resultiere aus folgender Tatsache:

> *„Traditionally undergraduate EE courses have focused on continuous time and detailed modeling of the physical phenomena [...], while undergraduate CS courses have focused on discrete representation and computational abstractions.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 11]

So lernten Studierende früh die jeweils anderen Gebiete kennen. In den Übungen zu dieser Vorlesung kommt verstärkt das Modellierungs- und Simulations-Werkzeug MATLAB/Simulink zum Einsatz.

> *„Using Matlab and Simulink [...], the students have an early exposure to hybrid systems [...] as an important domain for embedded systems.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 13]

Im Masterstudiengang wird vor allem der Kurs „*Design of embedded systems: models, validation and synthesis (EECS249)*“ vorgestellt.

> *„The idea of the course is to emphasize the commonality among the variety of application fields and to use a design methodology as the unified framework where the topics of the course are embedded.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 6]

So werde versucht, Grundlagen für weiterführende Kurse zu schaffen. Der Kurs selbst untergliedert sich in vier Blöcke[2]:

1. *„Introduction and Methodology“*:
   Zu Beginn des Kurses werden einige Beispiele präsentiert und besprochen. Dabei wird aber auch Wert auf die Methodik gelegt:

[2]Die Überschriften der einzelnen Blöcke wurden aus der Originalliteratur [Sangiovanni-Vincentelli und Pinto, 2005] aufgrund unpassender deutscher Übersetzung übernommen.

> *„[. . .] the course is intended to solve 'the embedded system design problem' rather thar particular instances of it.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 6]

2. *„Function“*:
   In diesem Block werden verschiedene Spezifikationssprachen behandelt sowie Berechnungsmodelle und Kommunikationsarten besprochen.

3. *„Architecture and Mapping“*:
   Dieser Abschnitt wird als der möglicherweise wichtigste im gesamten Kurs angesehen, da es hier darauf ankomme, die Studierenden zu befähigen, das Verhalten eines Systems optimal abzubilden:

   > *„Mapping functions to architectures is possibly the most relevant aspect of the Platform based design methodology taught in this class.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 7]

   In diesem Abschnitt wird auch das Thema *scheduling* besprochen.

4. *„Synthesis and Verification“*:
   Gegen Ende des Semesters erfolgen Vorlesungen zur Verifikation, sowie zur Beurteilung von Software und zum *„software synthesis problem“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 7], in dessen Zuge verschiedene Wege präsentiert werden, Code automatisch aus Modellen zu generieren.

Neben zu erledigenden Hausaufgaben wartet auf die Studierenden ein Abschlussprojekt, bei dem sie aus der Industrie und der Wirtschaft unterstützt werden, so dass die Projekte erfolgreich abgeschlossen werden können. Dabei seien einige Projekte durchaus sehr erfolgreich gewesen:

> *„Two other projects that had success to the point of being published in primary conferences were related to chip architectures.“* [Sangiovanni-Vincentelli und Pinto, 2005, S. 7]

Im Masterstudiengang folgen dann weitere fortgeschrittene Kurse, deren Darstellung an dieser Stelle im Hinblick auf eine Analyse unter dem Aspekt einer Implementierung im Informatikunterricht in allgemeinbildenden Schulen nicht hilfreich erscheint.

2.3 Ansatz der Technischen Universität Dortmund

Marwedel skizziert in seinem Artikel *„Towards laying common grounds for embedded system design education“* [Marwedel, 2005], welche Ausrichtung die Lehre im Bereich der eingebetteten Systeme an der TU Dortmund besitzt und beschreibt die Leitlinien und Lehrinhalte des entsprechenden Wahlpflichtkurses:

> *„[Courses] focusing either mostly on hardware [...] or mostly on software [...] will not be sufficient.“* [Marwedel, 2005, S. 25]

Vielmehr komme es auf einen breiten Überblick über den Entwurf eingebetteter Systeme an. Der Dortmunder Ansatz entspricht der Sichtweise, dass zunächst gute Grundlagen geschaffen werden müssen, um sich im weiteren Verlauf der Ausbildung bzw. des Berufslebens zu spezialisieren:

> *„Yet, it seems that fundamental bases are really difficult to acquire during continuous training if they haven't been initially learned, and we can think we must focus on them.“* [ARTIST, 2003, S. 14]

Die Vorlesung richtet sich an Diplom-Studierende zu Beginn des Hauptstudiums sowie an Bachelor-Studierende.

Die vorgestellte, und durch Übungen ergänzte, Vorlesung wird bereits seit etwa zehn Jahren gelesen und wurde in dieser Zeit ständig modifiziert. Nach der Einleitung, die u.a. Definitionen, Beispiele und Eigenschaften eingebetteter Systeme beinhaltet, werden verschiedene Themen besprochen, die für den Entwurf eines eingebetteten Systems notwendig sind. Grundlegend dabei ist die Kenntnis von Spezifikationssprachen, der Hardware eingebetteter Systeme sowie eingebetteten Betriebssystemen. Im Anschluss daran erfolgt die Thematisierung der Implementierung der Systeme mittels *hardware/software codesign* sowie die Validation.

Um der Vorlesung folgen zu können, werden einige Voraussetzungen formuliert (vgl. [Marwedel, 2005, S. 27]):

- Grundlegende Programmierkenntnisse
- Ein grundlegendes Verständnis von der Struktur, der Organisation und der Arithmetik innerhalb eines Computers
- Mathematische Grundlagen
- Ein grundlegendes Verständnis von elektronischen Schaltkreisen für den Abschnitt über Hardware
- Grundlagen von Betriebssystemen

Marwedel weist zudem darauf hin, dass in den vergangenen Jahren mit dieser Vorlesung sehr gute Erfahrungen gemacht wurden, gerade auch im Hinblick auf weiterführende Veranstaltungen:

> *„[...] it was realized that the ES course had laid excellent foundations for this more advanced course.“* [Marwedel, 2005, S. 28]

Weiterführende Veranstaltungen sind hierbei u.a. Seminare, in denen die Studierenden Themen selbst erarbeiten und präsentieren müssen.

2.4 Vergleich der Ansätze

Der Vergleich der im Vorfeld dargestellten Ansätze soll keinerlei subjektive Wertung vornehmen, sondern in objektiver Weise Unterschiede und Gemeinsamkeiten gegenüberstellen. Objektiv bedeutet hierbei auch, dass die Argumentationen für oder gegen verschiedene Sichtweisen angemerkt werden. Die Ansätze werden dazu bzgl. verschiedener Kriterien betrachtet, die im Anschluss eine Bewertung hinsichtlich der Eignung als Leitidee an einer allgemeinbildenden Schule im Informatikunterricht erleichtern.

Es wird dabei zunächst auf das Selbstverständnis eingegangen, das hinter den Ansätzen steht. Dabei können aus dem Selbstverständnis Schlüsse im Hinblick auf die Eignung als Unterrichtsinhalt in der Schule gezogen werden. Vor allem vor dem Hintergrund der Diskussion über die Einführung des Pflichtfaches Informatik an allgemeinbildenden Schulen ist dieser Punkt mit einem besonderen Augenmerk zu versehen und wird in 2.5.2.1 zur Bewertung näher analysiert.

Ein weiterer Punkt, der für die Schule von Bedeutung sein kann, ist das Suchen von Unterschieden und Gemeinsamkeiten in der Praxisnähe. Die Bedeutung hiervon besteht in der Motivation von Schülerinnen und Schüler für ein Thema, was in 2.5.2.2 näher beleuchtet wird.

2.4.1 Selbstverständnis

Um das Selbstverständnis eines Ansatzes herausarbeiten zu können, ist vor allem die Entwicklung von Bedeutung, die zum *status quo* geführt hat. An der KTH werden eingebettete Systeme im Rahmen des Maschinenbaustudienganges gelehrt, was - wie in Kapitel 2.1 bereits beschrieben - aus der langjährigen Zugehörigkeit eingebetteter Systeme zu den Ingenieurswissenschaften resultiere. Sowohl in Berkeley als auch in Dortmund stellt es sich dagegen so dar, dass man einen Kurs speziell für den Entwurf eingebetteter Systeme über zehn Jahre hinweg entwickelt hat. So ist es verständlich, dass in Berkeley bzw. Dortmund eine breitere Sichtweise auf die Thematik vorherrscht, während an der KTH das Hauptaugenmerk darauf liegt, dass die Studierenden für die Wirtschaft befähigt werden. Daher wird dort auch der konsequente Ansatz verfolgt, nur wenige Gebiete, diese aber vertieft, zu studieren. Dass hierbei von Grimheden gerade Mikrocontroller als Beispiel angeführt werden, deutet wieder auf die enge Verbundenheit zu den Ingenieurswissenschaften hin. Dieses Beschränken auf spezifische Probleme wird von Marwedel kritisiert:

> *„Traditionally, text books on embedded system design have focused on the very specific problem of interfacing computers to physical environments [...]. However, this view of embedded systems is much too restricted.“* [Marwedel, 2005, S. 25]

In der Tat ist es so, dass in Berkeley bzw. Dortmund explizit auch die Informatikstudierenden das Fach Eingebettete Systeme belegen können. In Berkeley steht dabei die Zusammenführung von verschiedenen Sichtweisen im Vordergrund. Deutlich wird dies anhand des kurz vorgestellten Kurses (EECS20N). Dieser frühe Kontakt mit den Arbeitsweisen der jeweils anderen Disziplin und auch die zahlreichen Vertiefungsmöglichkeiten, die sich aus dem Kurs (EECS249) ergeben, zeigen eine Sichtweise auf eingebettete Systeme, die die Grenzen zwischen Informatik und Elektrotechnik zu verwischen versucht.

In Dortmund liegt der Schwerpunkt auf der Sicht der Informatik auf den Entwurf eingebetteter Systeme.[3] Dies bestätigt allein schon die Tatsache, dass die meisten der etwa einhundert Hörerinnen und Hörer der Vorlesung Studierende des Diplomstudienganges sind.

Der Ansatz der KTH in Schweden ist vom Selbstverständnis her also den Ingenieurswissenschaften zuzuordnen, während die *University of California at Berkeley* versucht, die Sichtweise der Elektrotechnik mit der der Informatik in Einklang zu bringen. Dortmund fährt einen sehr stark informatikbezogenen Ansatz.

2.4.2 Praxisnähe

Auch in diesem Punkt unterscheidet sich der Ansatz der KTH deutlich von den Ansätzen aus Berkeley und Dortmund. An der KTH steht die Befähigung der Studierenden für die Wirtschaft im Vordergrund. Die Lehrinhalte sind also bewusst auf sehr konkreter Ebene gehalten. Dies wird auch durch die Kurse deutlich, die zur Hälfte von Vertretern der Wirtschaft und zur Hälfte von Studierenden besucht werden. Das gesamte Konzept ist daraufhin ausgerichtet, dass man die Anforderungen der Wirtschaft an die universitäre Lehre bedient. Durch diese Orientierung an der Wirtschaft ist die Ausbildung an der KTH mit einer sehr hohen Praxisnähe verbunden.

In Dortmund hingegen kann aufgrund der Breite des Stoffes nicht in jedem Bereich ein Beispiel aus der Praxis behandelt werden, für das man zu diesem Zeitpunkt der Ausbildung auch noch nicht über die nötigen Kenntnisse verfügt. Dafür sind in Dortmund weiterführende Kurse geplant. Deutlich wird das an den Übungen zu der vorgestellten Vorlesung:

> *„Due to the broad coverage of topics, the lab cannot and should not offer hands-on experience to tools in all of the [...] six areas. We propose to use a mixture of theoretical assignments and a limited set of tools to be used.“* [Marwedel, 2005, S. 26]

[3] Jedes Jahr im Sommersemester wird eine inhaltlich reduzierte Version der in Kapitel 2.3 vorgestellten Vorlesung gehalten. Diese wird auch von Studierenden der Robotik besucht.

Es resultiert daher für den Ansatz aus Dortmund eine geringere Praxisnähe, da reale Probleme meist nicht lediglich mit Grundlagenwissen adäquat gelöst werden können.

Eine Mischung aus beiden Ansätzen liefert wieder der Ansatz aus Berkeley. Zwar wird in dem schwerpunktmäßig vorgestellten Kurs (EECS249) ein recht breiter Überblick über den Entwurf geliefert, doch wird auch versucht, durch das Abschlussprojekt viel Praxisnähe zu gewährleisten. Dies wird vor allem durch die Vertreter der Wirtschaft realisiert, die einige Projekte unterstützen.

Es bleibt also festzuhalten, dass der Praxisbezug an der KTH und in Berkeley gegenüber dem Ansatz aus Dortmund höher ist. Eine Bewertung, ob es für die universitäre Lehre Vor- oder Nachteile mit sich bringt, wird an dieser Stelle nicht vorgenommen.

2.5 Bewertung der Ansätze

Die vorgestellten Ansätze beschreiben die Herangehensweise an die Ausbildung in eingebetteten Systemen der Universitäten, doch ist nicht jeder Ansatz unbedingt für allgemeinbildende Schulen zu empfehlen. Im vorangegangenen Unterkapitel wurde teilweise schon angemerkt, welche Faktoren Einfluss haben können. Daher erfolgt die Bewertung nun einerseits anhand des Allgemeinbildungsbegriffs und andererseits anhand obiger Kriterien. Zum Schluss wird eine zusammenfassende Bewertung der Ansätze sowie eine Schlussfolgerung vorgenommen, auf der Basis welchen Ansatzes die Untersuchungen im Rahmen dieser Arbeit fortgeführt werden.

2.5.1 Bewertung hinsichtlich des Allgemeinbildungsbegriffs

Da sich diese Arbeit ausschließlich mit allgemeinbildenden Schulen befasst, ist es folglich konsequent, wenn die Bewertung der Ansätze auf der Basis einer Definition von Allgemeinbildung vonstatten geht. Hubwieser zitiert einen Allgemeinbildungsbegriff nach Bussmann und Heymann:

> „Vorbereitung auf zukünftige Lebenssituationen. D.h. allgemein bildende Schulen sollen Qualifikationen vermitteln,
>
> a) die zur Bewältigung realer und auf absehbare Zeit in unserer Gesellschaft verbreiteter Lebenssituationen beitragen,
>
> b) die nicht auf die Ausübung eines bestimmten Berufes hin ausgerichtet sind,
>
> c) von denen anzunehmen ist, dass sie nicht gleichsam automatisch, nebenher von jedem Heranwachsenden erworben werden und
>
> d) die durch eine gewisse Universität, also Anwendbarkeit in sehr verschiedenen Situationen gekennzeichnet sind.“ [Hubwieser, 2007, S. 57]

Anhand dieser Definition werden nun also die Ansätze hinsichtlich einer Eignung als Leitidee an einer allgemeinbildenden Schule im Informatikunterricht bewertet. Dabei ist unbestreitbar, dass Punkt c) der Definition von keinem Ansatz erfüllt wird, da nicht davon ausgegangen werden kann, dass der Prozess des Entwurfs von eingebetteten Systemen von Heranwachsenden nebenher durchlaufen wird und sie somit die damit verbundenen Kompetenzen automatisch erwerben.

Punkt b) ist dabei schon schwerer zu bewerten. Der Ansatz der KTH erfüllt dieses Kriterium sicher nicht. Die Ausbildung ist dort auf die Wirtschaft hin ausgerichtet und verliert somit bzgl. dieses Punktes einen allgemeinbildenden Anspruch. Berkeley und Dortmund hingegen legen mit ihren Grundlagenkursen die Voraussetzungen für weiterführende Kurse. Es ist klar, dass die Richtung Informatik bzw. Elektrotechnik des späteren Berufs der Studierenden vorgegeben ist. Sie werden aber nicht auf einen bestimmten Beruf, beispielsweise als Designer eingebetteter Systeme, festgelegt. Natürlich kann man auch argumentieren, dass diese beiden Ansätze auf die Ausübung des Berufes des Informatikers bzw. Ingenieurs vorbereiten. Dies ist aber eher darin begründet, dass an der Universität eine Spezialisierung auf ein bestimmtes Fachgebiet stattfindet. Hinsichtlich des allgemeinbildenden Charakters für die Schule bieten die beiden Ansätze aus Berkeley und Dortmund somit eine breitere Grundlage.

Bezieht man die Praxisnähe auf den Punkt a) der Definition, so fällt auf, dass der Ansatz der KTH zu speziell ist. Geht man vom nichtspezifischen Transfer Schwills aus, könnte man geneigt sein, den Ansatz der KTH zu bevorzugen, da hier - wie in Abb. 2 dargestellt - durch die Behandlung von beispielhaften Lerngegenständen über eine Metaebene das Gelernte auf neue Situationen übertragen werden soll:

> *„The basic idea with an exemplifying selection is that the knowledge and skills learned from this selection could be generalized and applied to similar problems and areas.“*
> [Grimheden und Törngren, 2005, S. 36]

Es sollen aber Situationen, die in der Gesellschaft verbreitet auftreten, durch den Ansatz abgedeckt werden. Doch wirkt eine Behandlung weniger spezieller Themenbereiche und eine Ausrichtung hin auf Fertigkeiten dieser Anforderung entgegen. Ein zu verfolgender Ansatz muss daher den Kompetenzerwerb schwerpunktmäßig fördern. Betrachtet man die Ansätze aus Berkeley und Dortmund, fällt auf, dass diese Fokussierung auf Prinzipien gegeben ist. So werden beispielsweise nicht nur einige wenige Tools zur Spezifikation eingebetteter Systeme vertieft behandelt, sondern Anforderungsanalysen durchgeführt, welche Eigenschaften Spezifikationssprachen haben können. Dieser breite Überblick bereitet die Studierenden darauf vor, Designentscheidungen begründet treffen zu können.

Abbildung 2: Nichtspezifischer Transfer nach Schwill

Quelle: [Schwill, 1993, S. 20]

Auch Teil d) der Definition spricht im Bezug auf einen allgemeinbildenden Charakter nicht für den Ansatz der KTH, was die Struktur und die Lerninhalte anbelangt. Die Lehrinhalte sind zu speziell, als dass sie in verschiedenen Bereichen anwendbar sind. Modrow schreibt hierzu:

> „Nun sind sonst Ingenieurwissenschaften mit gutem Grund an allgemeinbildenden Schulen nicht vertreten, weil ihre Inhalte und Methoden zu speziell sind.“ [Modrow, 1991, S. 17]

Zieht man zur Bewertung nur die Inhalte heran, so folgt, dass der Ansatz der KTH dem Kriterium der Allgemeinbildung nicht standhält. Betrachtet man jedoch die methodische Ebene, lassen sich sehr wohl Punkte finden, die auf den ersten Blick für die Schule erstrebenswert wären. Dies bezieht sich vor allem auf den Punkt der *„interactive communication“* [Grimheden und Törngren, 2005, S. 36]. Hier wird der Frage nachgegangen, wie eingebettete Systeme gelehrt werden sollten:

> *„[. . .] in an interactive communication the action by the teacher is dependant on the current status and knowledge level of the individual student or student team.“* [Grimheden und Törngren, 2005, S. 36]

Dieses Eingehen auf den Lernstand der Schülerinnen und Schüler ist ein Aspekt, der bei einem ganzheitlichen Ansatz bedacht werden sollte.

Inhaltlich halten die Ansätze aus Berkeley und Dortmund dem Kriterium der Anwendbarkeit in verschiedenen Situationen auch nicht stand. Dafür sind die Lerninhalte auch hier zu speziell. Diese kann man aber durchaus einer Überprüfung auf Anwendbarkeit in verschiedenen Bereichen der Informatik unterziehen. Dies wird

auch noch im weiten Verlauf dieser Arbeit geschehen, wenn die Lerninhalte bzgl. der von Hubwieser [Hubwieser, 2007] abgeleiteten Kriterien untersucht werden.

Ein Blick auf die Methodik zeigt aber den gravierenden Unterschied zwischen den Ansätzen aus Berkeley und Dortmund. In Unterkapitel 2.4.2 wurde schon auf die unterschiedlichen Grade der Praxisnähe eingegangen. Bewertet man diese nun vor dem Hintergrund der Umsetzung in der Schule, könnte man den Ansatz aus Berkeley dem Dortmunder Modell vorziehen, da der problemorientierte Ansatz aus Berkeley durch das Abschlussprojekt sehr gut geeignet für den Informatikunterricht scheint. Doch es ist Intention des Abschlussprojektes, ein funktionierendes Produkt zu entwerfen, das am Ende des Semesters präsentiert werden soll. Daher kann diese Absicht des Ansatzes so nicht in der Schule umgesetzt werden. Humbert schreibt dazu:

> „So entsteht der Eindruck, dass 'Projektarbeit' im Informatikunterricht im Wesentlichen dem Ziel verpflichtet ist, ein funktionierendes Softwaresystem zu erstellen. Diese Sichtweise auf Fachinhalte und auf eine Vorgehensweise reicht nicht aus, den allgemeinbildenden Zielen im Schulfach Informatik gerecht zu werden.“ [Humbert, 2005, S. 77]

Möchte man diesen Gedanken, den Entwurf eingebetteter Systeme durch ein Projekt zu vertiefen, fortführen, so ist dabei folgendes zu bedenken:

> „Ziel eines Unterrichtsprojekts ist nicht ein kommerziell verwertbares Produkt, sondern die Förderung von Selbstständigkeit, Zusammenarbeit im Team und Kritikfähigkeit.“ [Hartmann *et al.*, 2006, S. 100]

Daraus folgt, dass ein Ansatz zunächst gute Grundlagen schaffen muss, damit diese in Form der Projektmethode - sofern man sie als Unterrichtsmethode in Erwägung zieht - erweitert werden können und dabei die oben genannten Fähigkeiten geschult werden.

Da die Methodik des Ansatzes aus Berkeley nur bedingt in der Schule umsetzbar ist, muss man sich, um dem Anspruch der Allgemeinbildung gerecht zu werden, auf grundlegende Inhalte und dazu passende Methoden beschränken. Aufgrund der inhaltlichen Nähe ist daher kein Vorteil für einen der Ansätze aus Berkeley bzw. Dortmund auszumachen.

2.5.2 Bewertung hinsichtlich vorgestellter Kriterien

In Kapitel 2.4 wurden die Kriterien Selbstverständnis und Praxisnähe genauer beleuchtet, um die Ansätze diesbezüglich zu bewerten. Daher wird nun das Selbstverständnis der Ansätze daraufhin analysiert, ob es für den Informatikunterricht

geeignet erscheint. Im Anschluss wird das Motivierungspotential der Praxisnähe der Ansätze bewertet.

2.5.2.1 Selbstverständnis Wie in Unterkapitel 2.4.1 herausgearbeitet wurde, bestehen teilweise gravierende Unterschiede im Selbstverständnis der Ansätze. Um sich nun begründet für einen Ansatz zu entscheiden, muss man auch in Betracht ziehen, welches Erscheinungsbild diese Wahl für die Informatik in der Schule bedeutet und wie sie die Sicht der anderen Fächer auf die Informatik beeinflusst. Verbände wie die Gesellschaft für Informatik (GI) setzen sich seit langem für den Stand der Informatik in der Schule ein:

> „Das deutsche Bildungswesen muss endlich dem Wandel zur Informations- und Wissensgesellschaft Rechnung tragen. Dies erfordert [...] ein eigenständiges Fach." [GI und BITKOM, 2007]

Vor allem im Unterricht der gymnasialen Oberstufe wird hierbei die Eigenständigkeit des Faches angestrebt:

> „In der Sekundarstufe II ist der bisherige geringe Stellenwert des Faches Informatik umgehend zu korrigieren. Es muss künftig mit gleichem Gewicht wie die anderen Fächer in der Sekundarstufe II etabliert und in der Abiturprüfung gleichberechtigt zu den Naturwissenschaften eingebracht und als Prüfungsfach gewählt werden können." [GI, 2000, S. 7]

Es wäre nicht ratsam, einen Ansatz auszuwählen, der anderen Fächern eine Argumentation dahingehend nahe legt, dass die Informatik lediglich als eine Art Zubringer für diese Fächer erachtet wird. Es ist zwar fächerübergreifender Unterricht wünschenswert, doch sollte die Informatik dabei immer ein eigenständiges Gesicht und somit ihre Daseinsberechtigung gegenüber anderen Fächern wahren.

Der Ansatz der KTH in Stockholm ist aus diesem Grund nicht geeignet, als Leitidee an einer allgemeinbildenden Schule im Informatikunterricht umgesetzt zu werden. Die starke Verschränkung mit anderen Fächern schon im Ansatz ist der Diskussion um die Eigenständigkeit des Faches Informatik nicht dienlich. Es kommen aber schon eher die Ansätze aus Berkeley und Dortmund in Frage, wenngleich es auch hier zu differenzieren gilt. Es hat sich im Lauf der Untersuchung in Unterkapitel 2.4.1 herausgestellt, dass der Ansatz aus Berkeley den Versuch startet, die Sichtweisen der Elektrotechnik mit denen der Informatik in Einklang zu bringen. Auch dieser Ansatz wäre das falsche Signal für die Rechtfertigung des Faches Informatik in der Sekundarstufe II. Es erscheint daher der Ansatz aus Dortmund, der sehr informatiknah ist, als geeignet für den Informatikunterricht in der gymnasialen Oberstufe.

2.5.2.2 Praxisnähe In Unterkapitel 2.4.2 wurde die Praxisnähe der einzelnen Ansätze herausgestellt. Es ist immer von Vorteil, wenn die Themen selbst interessant sind, die Schülerinnen und Schüler also aus intrinsischer Motivation heraus agieren. Hubwieser bezeichnet Motivierung sogar als „das vordringlichste Ziel didaktischen Handelns“ [Hubwieser, 2007, S. 15]. Inwieweit die durch große Praxisnähe erzeugte Motivation ansatzspezifisch ist, kann nicht abschließend bewertet werden. Dies liegt vor allem daran, dass die an der KTH gepflegte Praxisnähe an der Schule in der Regel nicht umgesetzt werden kann, weil es Schulen an Kooperationspartnern aus der Wirtschaft fehlt. Auch ist zu bezweifeln, dass nach Abschluss von Projekten das Ergebnis von Kooperationspartnern verwertet werden könnte, da hier die inhaltliche Komponente wieder mit eingeht. Die große Praxisnähe der Ansätze aus Berkeley und der KTH in Stockholm kann demnach nicht als motivierend für Schülerinnen und Schüler an allgemeinbildenden Schulen angesehen werden. Der Ansatz aus Dortmund bietet durch die Breite der Themen, die behandelt werden, viele Ansatzpunkte, Bezüge zur Praxis herzustellen und somit den Schülerinnen und Schülern Motivationsgelegenheiten anzubieten.

2.5.3 Zusammenfassende Bewertung

Ziel dieses Kapitels war es, begründet einen Ansatz auszuwählen, der als Leitidee an einer allgemeinbilden Schule umgesetzt werden kann. Die Königlich Technische Hochschule in Stockholm bietet aber wenige Gründe, sich für dieses Modell zu entscheiden. Aufgrund der sehr hohen Spezialisierung und der starken Verschränkung mit anderen Fächern ist dieser Ansatz für die Ziele dieser Arbeit nicht geeignet.

Die Ansätze aus Berkeley und Dortmund sind zwar ähnlich, unterscheiden sich aber in einem wesentlichen Punkt. Vom Selbstverständnis her ist der Ansatz aus Dortmund dem aus Berkeley vorzuziehen, da hier eine starke informatische Sichtweise auf die Thematik zu Grunde liegt. Hinsichtlich der Breite der Themen sind beide Vorgehensweisen weitestgehend kompatibel mit dem Allgemeinbildungsbegriff. Die größere Praxisnähe des Ansatzes aus Berkeley fällt - wie in 2.5.2.2 erläutert - nicht weiter ins Gewicht.

Somit folgt, dass der Ansatz aus Dortmund am geeignetsten erscheint, als Basis für die Untersuchungen in dieser Arbeit zu dienen. Im folgenden Kapitel wird daher die Möglichkeit untersucht, den Entwurf eingebetteter Systeme als eigenständiges Modul in der Sekundarstufe II unterrichtlich zu behandeln.

2.6 Zusammenfassung

In diesem Kapitel wurden drei verschiedene Ansätze vorgestellt, wie eingebettete Systeme bzw. ihr Entwurf an Hochschulen gelehrt werden. Zunächst wurde der Ansatz der Königlich Technischen Hochschule in Stockholm vorgestellt, der darauf setzt, dass die Ausbildung der Studierenden nah an den Interessen der Wirtschaft ausgerichtet ist. Die Lehrinhalte sind zwar in der Breite recht dünn, gehen dafür aber nach dem Motto *„teaching 'everything of something' rather than 'something of everything'“* [Grimheden und Törngren, 2005, S. 34] sehr stark in die Tiefe.

Die Ansätze aus Dortmund und Berkeley sind hingegen recht ähnlich, was die Struktur der vorgestellten Vorlesungen betrifft. Hier wird versucht, einen Schwerpunkt auf die Sichtweise der Informatik auf das Design eingebetteter Systeme zu legen. Die jeweiligen Ausdifferenzierungen der Kurse unterscheiden sich dabei in der Praxisnähe.

Die Bewertung der Ansätze ist anhand des Allgemeinbildungsbegriffs sowie des Selbstverständnisses und der Praxisnähe der jeweiligen Ansätze erfolgt. Dabei erscheint der Ansatz aus Dortmund am geeignetsten, ihn als Grundlage für den weiteren Verlauf dieser Arbeit zu verwenden, da mit diesem Ansatz die Informatik potentiell eine weitere Rechtfertigung erhält, als eigenständiges und gleichwertiges Fach angesehen zu werden.

3 Ein ganzheitlicher Ansatz

Nachdem im letzten Kapitel begründet ein Ansatz, der als Grundlage für die weiteren Untersuchungen in dieser Arbeit dienen soll, ausgewählt wurde, wird nun eine Menge von Unterrichtsinhalten bestimmt, die zu einem ganzheitlichen Ansatz führen sollen. Dazu bedarf es zunächst einer Erläuterung, was mit einem ganzheitlichen Ansatz gemeint ist. Diese Begriffsdefinition sowie die Mittel, zu solch einem ganzheitlichen Ansatz zu kommen, werden im ersten Unterkapitel thematisiert.

Im Anschluss an die Kennzeichnung folgt der eigentliche Schwerpunkt dieser Arbeit. Ausgehend von der Buchvorlage [Marwedel, 2007] werden einzelne Themen näher beleuchtet und durch fachliche und didaktische Argumente hinsichtlich des Ziels dieser Arbeit analysiert.

Im letzten Unterkapitel erfolgt dann der Versuch einer Strukturierung der Menge der Lerninhalte sowie einer Bewertung. Im Zuge dieser Bewertung wird am Ende begründet Stellung dazu bezogen, ob auf der Basis der Ergebnisse dieser Arbeit ein ganzheitlicher Ansatz über den Entwurf eingebetteter Systeme möglich erscheint.

3.1 Kennzeichnung eines ganzheitlichen Ansatzes

Ein ganzheitlicher Ansatz muss darauf abzielen, den Schülerinnen und Schülern einen breiten Überblick über den Entwurf eingebetteter Systeme zu vermitteln. Dies bedeutet, dass eingebettete Systeme nicht als Beispiel verwendet werden, um andere Bereiche der Informatik zu veranschaulichen, sondern ein eigenständiges Unterrichtsmodul in der gymnasialen Oberstufe bilden, das eine wirkliche Alternative zu bestehenden Modulen wie „Netzwerkanwendungen“, „Relationale Datenbanken“ oder „Endliche Automaten und formale Sprachen“ (vgl. [MSW, 2008]) schafft.

Inhaltlich darf man sich nicht der Illusion hingeben, man könne in der Schule die Themen in der Tiefe behandeln, wie es an Universitäten üblich ist. Daher muss der Stoff einer didaktischen Reduktion unterzogen werden. Was sich hinter dem Begriff der didaktischen Reduktion prinzipiell verbirgt, wird ein Thema dieses Unterkapitels sein. Um begründet das Mittel der didaktischen Reduktion anwenden zu können, bedarf es einiger Kriterien für die Auswahl von Lerninhalten. Diese von Hubwieser aufgestellten Kriterien werden im Anschluss thematisiert.

3.1.1 Didaktische Reduktion von Lerninhalten

Mit einer vollständigen unterrichtlichen Behandlung des Entwurfs eingebetteter Systeme kann natürlich nicht gemeint sein, dass alle Facetten und Ausprägungen eines Ansatzes, der an Universitäten gelehrt wird, identisch auf die Schule abgebildet

werden. Vielmehr ist es erforderlich, dass man sich auf die inhaltlichen und methodischen Kernpunkte beschränkt, die den Anforderungen an Allgemeinbildung (siehe dazu auch Kapitel 2.5.1) genügen:

> „[Es] kann sich der schulisch aufzuarbeitende Teil der Informatik nicht allein der Systematik der universitären Bezugsdisziplin unterwerfen. Bei der rasanten Eigendynamik dieses Leitfaches wird jeder Versuch fehlschlagen, die zentralen fachlichen Inhalte überwiegend mit Anleihen aus dem Wissenschaftsbereich zu beschreiben.“ [MSW, 1999a, S. 5-6]

Es ist notwendig, dass die Lerninhalte einer didaktischen Reduktion unterzogen werden. Rösler und Schmidkunz [Rösler und Schmidkunz, 1996] schreiben hierzu:

> „[Die didaktische Reduktion ist ein] Vorgang, bei dem [...] komplexe Sachverhalte für eine bestimmte Lerngruppe auf das jeweils Wesentliche zurückgeführt [...] [wird], um Überschaubarkeit und Begreifbarkeit für den Lernenden zu erreichen.“ [Rösler und Schmidkunz, 1996, S. 4]

Sie stellen in ihrem Artikel desweiteren die „drei Möglichkeiten der didaktischen Vereinfachung“ [Rösler und Schmidkunz, 1996, S. 4] nach Hering (siehe hierzu auch die Literatur [Rösler und Schmidkunz, 1996, S. 5]) vor, auf die in Kapitel 3.2 zurückgegriffen wird:

1. Die Anzahl der merkmalträchtigen Einzelheiten einer Aussage wird vermindert: In diesem mehrstufigen Prozess wird eine vereinfachte Aussage durch Weglassen bzw. Vereinfachen von Teilaussagen gewonnen, wobei die vereinfachte Aussage immer noch den Gültigkeitsumfang der ursprünglichen Aussage - allerdings auf einem allgemeineren Niveau - inne hat.

2. Teilaussagen können in den Vordergrund gestellt werden: Bei dieser Akzentuierung des Wesentlichen bestehe allerdings die Gefahr, dass ein falscher Eindruck der Ursprungsaussage entstehe.

3. Teilaussagen können durch einen Oberbegriff ersetzt werden: Diese Zusammenfassung lässt Details weg und ermöglicht somit eine Konzentrierung auf die wesentlichen Aspekte, die die Teilaussagen gemeinsam haben.

Somit sind die Arten der Reduzierung beschrieben. Es fehlt nun aber noch ein Kriterienkatalog, nach dem die Teilaussagen ausgewählt werden, die weggelassen, in den Vordergrund gerückt oder generalisiert werden. Dies geschieht im nächsten Abschnitt.

3.1.2 Didaktische Auswahlkriterien für Lerninhalte

In diesem Abschnitt werden die vier Kriterien für die Auswahl von Lerninhalten vorgestellt, die Hubwieser [Hubwieser, 2007] aus den Kriterien zur Identifizierung von fundamentalen Ideen eines Faches ableitet. Hubwieser fordert

- die allgemeine Bedeutung,
- die Lebensdauer,
- die Vermittelbarkeit und
- die exemplarische Auswahl und Einflechtung

von auszuwählenden Lerninhalten ein (vgl. [Hubwieser, 2007, S. 83f]). Die einzelnen Kriterien werden im Folgenden näher erläutert. Dabei ist es auch wichtig, die Kriterien unter dem Aspekt der speziellen Anforderungen an ein Modul über eingebettete Systeme zu bewerten.

3.1.2.1 Allgemeine Bedeutung Hubwieser unterteilt die Breite der Anwendungsbereiche im Bezug auf Informatiksysteme in vier Klassen (vgl. dazu Tabelle 4.2 in Hubwiesers Buch [Hubwieser, 2007, S. 83]) ein. Dabei ist bei der Auswahl der Lerninhalte eines Moduls über eingebettete Systeme darauf zu achten, dass man sich nicht nur auf ein konkretes System, wie beispielsweise einen bestimmten Mikrocontroller (vgl. Ansatz der KTH (2.1)), beschränkt, sondern die Prinzipien in den Vordergrund rückt, die womöglich auch eine Verwendung „außerhalb des Bereiches der EDV möglich“ [Hubwieser, 2007, S. 83] machen.

3.1.2.2 Lebensdauer Die Forderung nach Lebensdauer beschreibt Hubwieser als eine Möglichkeit innerhalb von Allgemeinheitsklassen zu differenzieren, da diese mit der Forderung nach allgemeiner Bedeutung korreliere. Es dürften keine Fachinhalte gelehrt werden, deren Bedeutung für die Informatik - insbesondere hinsichtlich einer Langfristigkeit der Bedeutung - noch nicht geklärt sei.

3.1.2.3 Vermittelbarkeit Vermittelbarkeit erscheint als eine so grundlegende Forderung, dass sie auf den ersten Blick keiner weiteren Erläuterung bedarf. Doch fällt dieser Forderung vor dem Hintergrund des Ziels dieser Arbeit eine besondere Bedeutung zu. Da im Ansatz aus Dortmund selbst Voraussetzungen formuliert werden, die notwendig seien, um der Vorlesung folgen zu können, müssen bei der

Auswahl von Lerninhalten diese Voraussetzungen ebenfalls bedacht werden. Sollten sie in der entsprechenden Jahrgangsstufe nicht gegeben sein, so kann dies dazu führen, dass bestimmte Themen außer Acht gelassen werden müssen.

3.1.2.4 Exemplarische Auswahl und Einflechtung Die exemplarische Auswahl wird der zeitlichen Beschränkung der Schule gerecht. Es bestehe „die Notwendigkeit einer Beschränkung auf einige wenige repräsentative Probleme“ [Hubwieser, 2007, S. 84]. Auch bei den eingebetteten Systemen wird es darauf ankommen, für übergeordnete Ideen und Konzepte geeignete Lerninhalte herauszufiltern.

Die Einflechtung des Stoffs, die Hubwieser als Lerninhalte, die „nebenbei, im Zuge der Beschäftigung mit anderen Themen“ [Hubwieser, 2007, S. 85] vermittelt werden können, bezeichnet, erscheint für diese Arbeit nicht geeignet. Es geht darum, einen ganzheitlichen Ansatz für den Entwurf eigebetteter Systeme zu konzipieren. Daher ist eine Verteilung der Lerninhalte in anderen Teildisziplinen nicht wünschenwert. Dies kommt schon eher in Frage, wenn ein ganzheitlicher Ansatz nicht geeignet erscheint, gewisse Teile des Entwurfs eingebetteter Systeme aber dennoch daraufhin untersucht werden sollen, ob sie in anderen Modulen integriert werden können.

3.2 Herleitung einer Menge von Unterrichtsinhalten

In diesem Unterkapitel wird aufbauend auf dem Ansatz der TU Dortmund eine Menge von Unterrichtsinhalten hergeleitet. Daher dient als Grundlage das Buch von Marwedel [Marwedel, 2007], das auch das von ihm ausgewiesene Begleitmaterial zu der in Kapitel 2.3 beschriebenen Vorlesung ist. Somit ist sichergestellt, dass ein von diesem Buch didaktisch reduzierter Ansatz auch aus fachlicher Sicht das Kriterium der Ganzheitlichkeit erfüllt. Das Buch ist in sechs Kapitel gegliedert:

1. Einleitung
2. Spezifikationssprachen
3. Hardware eingebetteter Systeme
4. Eingebettete Betriebssysteme, *Middleware* und Scheduling
5. Implementierung eingebetteter Systeme: Hardware-/Software-Codesign
6. Evaluierung und Validierung

Die Grobstruktur des Ansatzes wird sich an diese sechs Kapitel anlehnen, um einen breiten Überblick über den Entwurf eingebetteter Systeme zu gewährleisten. Es empfiehlt sich daher, das Buch parallel zu dieser Arbeit zu lesen. Es ist nicht Ziel dieser Arbeit, größtenteils zu reproduzieren, es kommt vielmehr auf die eigentlichen Argumentationsstränge an, die zu der Empfehlung oder Nicht-Empfehlung der Aufnahme des entsprechenden Lerngegenstands in die Menge der möglichen Unterrichtsinhalte führen.

Im Folgenden werden die einzelnen Kapitel der didaktischen Reduktion nach Hering (siehe dazu Kapitel 3.1.1) unterzogen. Das Mittel, die entsprechenden Teilaussagen auszuwählen, liefert der Kriterienkatalog Hubwiesers. Auch Anknüpfungspunkte, die sich im Sinne eines Spiralcurriculums anbieten, werden ausgewiesen. Es ist sicher nicht möglich, alle Argumente für und gegen jeden Lerninhalt im Rahmen dieser Arbeit aufzulisten und gegeneinander abzuwägen. Vielmehr sei nochmals darauf hingewiesen, dass es das Ziel ist, dass am Ende der Untersuchung eine Menge von potentiellen Unterrichtsinhalten aufgelistet wird, die auf einem nachvollziehbar begründeten Fundament stehen. Dabei ist dieses Unterkapitel als erster Durchgang zu verstehen. Am Ende steht nicht eine fertige, strukturierte und praxiserprobte Unterrichtseinheit, sondern eine Basis für weitere Schlussfolgerungen.

3.2.1 Eigenschaften eingebetteter Systeme

Im Einleitungskapitel sind als wichtigste Aussagen die Definition von eingebetteten Systemen, ihre Eigenschaften sowie ihre Anwendungsgebiete zu identifizieren. Gerade zu Beginn einer neuen Unterrichtsreihe kommt es darauf an, die Schülerinnen und Schüler für das bevorstehende Thema zu motivieren. Daher bietet sich als Anknüpfungspunkt im Sinne eines spiral curricularen Vorgehens der Bereich „Informatik und Gesellschaft“ (I&G) an. Die Thematisierung von Anwendungsgebieten von eingebetteten Systemen verdeutlicht die immer weiter steigende Allgegenwärtigkeit der Informatik in unserer Umgebung. Bezieht man sich auf Hubwiesers Kriterien, erreicht man hierdurch eine allgemeinere Stufe, da Beispiele für eingebettete Systeme dann nicht mehr ohne Kontext behandelt werden, sondern durch Diskussionen eine Kompetenzschulung stattfindet:

> „Ansätze zur Einschätzung der Möglichkeiten und Grenzen des Computereinsatzes und die Thematisierung seiner gesellschaftlichen Auswirkungen regen auf der Grundlage eines soliden Fachwissens die Urteils- und Kritikfähigkeit an.“ [MSW, 1999a, S. 8]

Um die Schülerinnen und Schüler nun zu befähigen, sich ein kritisches Urteil bilden zu können, muss daher das entsprechende Fachwissen thematisiert werden. Daher

sollten auch die Definition und Eigenschaften eingebetteter Systeme in die Menge der möglichen Unterrichtsinhalte aufgenommen werden. Gerade vor dem Hintergrund einer Diskussion über die Anwendungsbereiche von eingebetteten Systemen ist in der Oberstufe die Vermittelbarkeit von Eigenschaften sowie der Definition gegeben.

In diesem Kapitel sollte man sich auf Anwendungsgebiete, die Definition sowie einige Eigenschaften, die sich aus den Anwendungsgebieten ergeben, beschränken. Diese Reduzierung stellt sicher, dass man sich nicht in Einzelheiten verliert. Zwar werden einige Eigenschaften in den Vordergrund gerückt und dadurch andere nicht besprochen[4], doch wird in der Definition auch ausdrücklich darauf hingewiesen, dass eingebettete Systeme in der Regel nicht alle der beschriebenen Eigenschaften erfüllen (vgl. dazu auch die Literatur [Marwedel, 2007, S. 5]).

3.2.2 Spezifikationssprachen

Am Anfang des Entwurfs eines eingebetteten Systems steht die Spezifikation. Im Ansatz aus Dortmund werden viele verschiedene Spezifikationssprachen sowie generelle Anforderungen und Eigenschaften in verschiedener Tiefe behandelt. Über allem stehen verschiedene Berechnungsmodelle bzw. Kommunikationsarten. Es ist sicher nicht sinnvoll, den Stoff in der Tiefe im Schulunterricht zu behandeln, wie er im Buch vorgestellt ist. Vielmehr sollte man sich darauf beschränken, verschiedene Berechnungsmodelle und Kommunikationsarten vorzustellen und exemplarisch wenige Spezifikationssprachen als Repräsentanten einer Klasse zu besprechen.

Die Hervorhebung dieses Aspekts erleichtert zudem die Bewertung hinsichtlich der allgemeinen Bedeutung. Diese besteht vor allem darin, dass verschiedene Paradigmen kennengelernt werden. Diese Ermunterung zu Paradigmenwechseln innerhalb der Oberstufe ist auch im Lehrplan von Nordrhein-Westfalen verankert:

> „Für ein Unterrichtskonzept über eine Dauer von drei Jahren ist ein Paradigmenwechsel wünschenwert, damit die Schülerinnen und Schüler zumindest exemplarisch erfahren, dass bestimmte Problemklassen durch grundlegend andersartige Denkansätze mit neuen Werkzeugen vielfach effizienter gelöst werden können.“ [MSW, 1999a, S. 27]

Zwar bezieht sich der hier angesprochene Paradigmenwechsel auf mehrere verschiedene Sprachkonzepte wie dem imperativen, objektorientierten, wissensbasierten oder funktionalen Ansatz, doch bleibt die Argumentation im Kontext der Berechnungsmodelle die gleiche.

Es bleibt nun also noch zu klären, welche Repräsentanten der Berechnungsmodelle und Kommunikationsarten in den Vordergrund gerückt werden sollen. Anhand der

[4]Eine allzu detailreiche Thematisierung von Effizienz eingebetteter Systeme erscheint in diesem frühen Stadium aufgrund der Gefahr, sich in Einzelheiten zu verlieren, nicht sinnvoll.

Themen im Buch werden nun einige Punkte diskutiert. Viele der nur kurz vorgestellten Sprachen finden dabei in dieser Arbeit keine Berücksichtigung, da sie entweder den hier behandelten Sprachen ähnlich sind oder selbst im Buch nur kurz erwähnt werden. Im Sinne eines breiten Ansatzes soll daher nicht zu sehr ins Detail gegangen werden.

3.2.2.1 *StateCharts* Um auch hier Anknüpfungspunkte zu bieten, könnte man geneigt sein, bekannte Diagrammtypen von UML für die Speziafikation zu verwenden. Marwedel weist aber darauf hin, dass die „nicht genau definierte Semantik von UML“ [Marwedel, 2007, S. 54] Probleme bereiten könnte und UML daher mit einer ausführbaren Sprache kombiniert werden müsse. Von UML können aber Zustandsdiagramme verwendet werden, die eine Variante der *StateCharts* sind.

StateCharts sind über gemeinsamen Speicher kommunizierende endliche Automaten und sind damit der erste Repräsentant einer Klasse von Berechnungsmodellen. Je nachdem, wie intensiv UML in den vorangegangenen Jahrgangsstufen behandelt wurde, stellen sie eine Verknüpfung zu bereits Gelerntem dar, gehen aber über reine Reproduktion hinaus, da die spezielle Semantik von *StateCharts*, die *StateMate*-Semantik, für deterministisches Verhalten sorgt. Die für eingebettete Systeme so wichtige Modellierung von Zeit geschieht durch spezielle *Timer*.

Da Zustandsdiagramme im Rahmen von UML in der Oberstufe problemlos behandelt werden können, ist die Vermittelbarkeit trotz der Erweiterungen durch *Timer* und die *StateMate*-Semantik vor dem Hintergrund der im Vorfeld erarbeiteten Eigenschaften eingebetteter Systeme[5] gegeben. Dabei ist zu erwähnen, dass es Programme gibt, die die Spezifikation durch *StateCharts* unterstützen und sogar ausführbar sind, die Spezifikation also hinsichtlich des Verhaltens simuliert werden kann.[6] Ein Beispiel für solch ein Programm ist das Java-basierte Tool Dave.[7]

3.2.2.2 Kahn Prozessnetzwerke Um einen Überblick über Berechnungsmodelle und Kommunikationsarten geben zu können, sollte auch eine zweite Spezifikationssprache behandelt werden, die ein weiteres Berechnungsmodell, das verschieden ist von dem Von-Neumann-Paradigma, und nach Möglichkeit auch eine andere

[5]Das Anforderungsprofil eingebetteter Systeme setzt Determinismus und auch Berücksichtigung der Zeit voraus. Wird dies vorher erarbeitet, erscheinen Modellierung von Zeit und die Forderung nach Determinismus in der Spezifikationssprache als logische Konsequenz, müssen demnach nicht separat als allgemeine Forderung an Spezifikationssprachen mit den Schülerinnen und Schülern erarbeitet werden.

[6]Dabei ist zu beachten, dass diese Simulation nicht eine Simulation im Sinne von Abschnitt 3.2.6 ist. Dort geht es um die Simulation des fertig konstruierten eingebetteten Systems.

[7]Nähere Informationen zu dem Programm sind unter `http://ls12-www.cs.tu-dortmund.de/edu/ES/ES2007-Uebung/dave.html` abrufbar [Stand: 2008-09-15].

Kommunikationsart vorstellt. Betrachtet man Abb. 2.64 aus der angegebenen Literatur [Marwedel, 2007, S. 91], kommen hier vor allem Kahn Prozessnetzwerke (KPN) bzw. das synchrone Datenfluss-Modell (SDF) in Frage.

Für ein KPN spricht die Tatsache, dass bereits ein Lernmodul [Sirocic, 2007] existiert, das von der TU Dortmund in der Lehre verwendet wird. Dieses Lernmodul bietet auch die Möglichkeit, die Modellierung zu simulieren, was einen nicht unwesentlichen Vorteil darstellt. Die Vermittelbarkeit ist gewährleistet, da neben Taskgraphen, die problemlos neu eingeführt werden können, das Verständnis von Warteschlagen gefordert ist, was in der Jahrgangsstufe 12 laut Vorgaben für das Zentralabitur [MSW, 2008] vorgesehen ist. Problematisch ist die theoretisch unendliche Größe dieser Puffer, die Zweifel an der praktischen Relevanz aufkommen lassen könnten. Durch das Lernmodul, das die Puffergröße auf zwanzig beschränkt, können Probleme praktisch gelöst werden, so dass durch kreative Aufgaben ein sinnstiftender Kontext geschaffen werden kann. Zudem können in dem Lernmodul die Anweisungen des Verhaltens der Prozesse in der Programmiersprache Java definiert werden, so dass sich die neuen Lerninhalte wirklich auf die Theorie der Prozessnetzwerke beschränken und die Schülerinnen und Schüler nicht von einer neuen und komplex wirkenden Anweisungssyntax abgeschreckt werden.

Da für das SDF bisher noch keine Lernumgebung vorhanden ist, werden Kahn Prozessnetzwerke vorgezogen. Die Theorie hinter beiden Modellen ist, bezogen auf die Hauptaspekte, die gleiche, so dass die praktischen Vorteile überwiegen.

3.2.2.3 Specification and Description Language Durch die Behandlung von *StateCharts* und Kahn Prozessnetzwerken sind je zwei Berechnungsmodelle und Kommunikationsarten abgedeckt. Zwar steht die *Specification and Description Language* (SDL) für eine Spezifikationssprache, die asynchronen Nachrichtenaustausch beschreibt und kommunizierende endliche Automaten als Berechnungsmodell zu Grunde legt, diese Kombination also neu wäre, doch muss man sich bzgl. der didaktischen Reduktion des Stoffes bewusst machen, welcher Aspekt in den Vordergrund gestellt werden soll. hier kommt es nicht darauf an, dass die Schülerinnen und Schüler einen möglichst breiten Pool an Spezifikationssprachen beherrschen, sondern den übergeordneten Blick auf den Entwurf eingebetteter Systeme kennenlernen. Daher sollte SDL nicht in die Menge der möglichen Unterrichtsinhalte aufgenommen werden, auch wenn mit dieser Sprache beispielsweise ISDN spezifiziert wurde und den Schülerinnen und Schülern somit ein prominentes praktisches Anwendungsgebiet aufgezeigt würde.

3.2.2.4 Java Mit Java wird eine für die Schule bekannte Programmiersprache[8] aus einem völlig neuen Blickwinkel betrachtet. Marwedel weist auf die Vorteile Javas gegenüber den Programmiersprachen C bzw. C++ hin [Marwedel, 2007, S. 63]. Es werden allerdings auch kritische Aspekte wie die problematische Umsetzung der Nebenläufigkeit durch *Threads*, die Größe der Bibliotheken oder die Rechenzeit der automatischen Speicherverwaltung angemerkt. Für die Schule ergäbe sich eine Möglichkeit, ohne dass die Schülerinnen und Schüler extra eine neue Sprache lernen müssten, eingebettete Systeme zu spezifizieren, wenn nicht die damit verbundenen Problematiken entstünden.

Aus fachlicher Sicht sind diese Problematiken die oben erwähnten, aus didaktischer Sicht ergibt sich aber noch ein anderes Problem. Durch die Spezifikation mittels Java wird nicht die gewünschte Erweiterung des Horizonts erreicht. Es besteht die Gefahr, dass eingebettete Systeme als eine Anwendung der Programmierung mit Java verstanden werden und das eigentliche Ziel, eine Klasse von Informatiksystemen zu begreifen, aus den Augen verloren wird. Ist Java nicht die Sprache der Wahl, so empfiehlt es sich auch nicht, nur für die Spezifikation eines eingebetteten Systems einen Java-Kurs abzuhalten. Die Lehre der Syntax und Semantik verbrauchte zu viel Zeit, die noch für die übrigen Kapitel erforderlich ist.

3.2.2.5 VHDL Mit VHDL erhielten die Schülerinnen und Schüler die Möglichkeit, eine andere Klasse von Sprachen kennenzulernen. Bis zur Behandlung von VHDL hätten sie demnach nur Sprachen kennengelernt, die der Entwicklung von Software dienen.[9] Die textuelle Beschreibung von Hardware, gerade was auch Nebenläufigkeit anbelangt, wäre eine neue Sichtweise.

Es besteht dabei aber die nicht unwesentliche Gefahr, dass Syntax und Semantik wie bei üblichen Programmiersprachen erarbeitet werden müssen, um VHDL einsetzen zu können. Dies nimmt einerseits viel Zeit in Anspruch, die dann für die wesentlichen Ziele dieses Unterrichtsmoduls fehlt, andererseits könnte der Entwurf eingebetteter Systeme zu einem Programmierkurs entarten, der nur auf Fertigkeiten ausgerichtet ist, anstatt Kompetenzen zu schulen. Damit wäre das Kriterium der allgemeinen Bedeutung nicht erfüllt.

Für eine unterrichtliche Thematisierung von VHDL spräche allerdings, dass hier-

[8] Voraussetzung ist natürlich, dass die Programmiersprache der Wahl wirklich Java ist. Für das Zentralabitur sind die beiden Sprachen Delphi und Java zugelassen. Einige Probleme wie beispielsweise Zeiger-Arithmetik oder Freigabe von nicht mehr verwendetem Speicher, die in Java nicht vorhanden sind, kommen in Delphi zum Tragen.

[9] Eine vorangegangene Unterrichtseinheit über theoretische Informatik widerspräche dieser Behauptung natürlich. Dies änderte aber nichts daran, dass VHDL für die Schülerinnen und Schüler zu einer neuen Klasse von Sprachen gehört.

durch ein Repräsentant eines bis dahin noch nicht vorgestellten Berechnungsmodells gegeben wäre, das „Diskrete Ereignismodell“. Es bleibt aber auf der konkreten Ebene die Frage, in welcher Form VHDL von den Schülerinnen und Schülern eingesetzt werden kann. Entscheidet man sich dafür, VHDL zu thematisieren, liegt der Schwerpunkt des gesamten Moduls aufgrund des Zeitfaktors automatisch auf diesem Bereich. Dies widerspräche der Ganzheitlichkeit des Ansatzes.

Eine Möglichkeit, VHDL doch in der Schule zu behandeln, wäre ein sich an das Modul anschließendes Projekt, in dessen Rahmen rekonfigurierbare Bausteine programmiert werden. Dabei müsste aber noch die Realisierbarkeit eines solchen Projektes näher untersucht werden, was im Rahmen dieser Arbeit nicht stattfinden soll.

3.2.2.6 Petri-Netze Petri-Netze nehmen eine besondere Rolle ein, da sie durchaus schon in anderen Zusammenhängen in der Schule behandelt werden können und somit der Entwurf eingebetteter Systeme auf etwas Bekanntes zurückgreifen könnte. Schubert und Schwill [Schubert und Schwill, 2004] benennen die Vorteile, die sich durch die unterrichtliche Thematisierung von Petri-Netzen ergeben:

> „[...] mit Petri-Netzen [lassen sich] alle wichtigen Begriffe und Phänomene der parallelen Programmierung [...] anschaulich studieren und beschreiben.“ [Schubert und Schwill, 2004, S. 193]

Hubwieser schlägt dabei vor, Petri-Netze im Rahmen einer Unterrichtseinheit über Rechnerkommunikation zu behandeln (siehe [Hubwieser, 2007, S. 248-252]). Um zu einer Entscheidung zu kommen, ob Petri-Netze im Rahmen eines Moduls über eingebettete Systeme besprochen werden sollen, muss sich Klarheit verschafft werden über die Ziele, die man verfolgt. In der universitären Vorlage werden Petri-Netze vor allem zur „Modellierung kausaler Abhängigkeiten“ [Marwedel, 2007, S. 49] verwendet. Dies wird ausgenutzt, um Aussagen über das Gesamtsystem zu machen und diese auch formal zu beweisen. Für die Schule ergäbe sich daher die Konsequenz, Petri-Netze auch unter diesem und nicht unter dem Gesichtspunkt zu behandeln, den Hubwieser bzw. Schubert und Schwill vorgeschlagen haben. Dabei stellt sich vor allem die Frage der Vermittelbarkeit. Um die sogenannten „S-Invarianten“ [Marwedel, 2007, S. 45] zu finden, werden tiefgreifende Kenntnisse der Mathematik benötigt, die die Schülerinnen und Schüler zu diesem Zeitpunkt im Allgemeinen nicht haben werden. Somit bliebe noch der Aspekt der Modellierung verteilter Systeme, den Schubert und Schwill hervorgehoben haben. Die von ihnen angestrebte anschließende Implementierung von Petri-Netzen führt aber zu sehr aus dem Bereich der eingebetteten Systeme, so dass die abschließende Bewertung von Petri-Netzen dahin-

gehend ausfällt, dass sie nicht im Rahmen des hier entwickelten Unterrichtsmoduls behandelt werden sollten.

3.2.2.7 Anforderungen und Spracheigenschaften Eine Thematisierung von Anforderungen an Spezifikationssprachen oder die Erstellung einer Sammlung von allgemeinen Spracheigenschaften im Unterricht erscheint nicht sinnvoll. Eine vollständige Auflistung könnte die Notwendigkeit hervorrufen, weitere Spezifikationssprachen zu besprechen, da keine der Sprachen, seien es *StateCharts*, Kahn Prozessnetzwerke oder auch Programmiersprachen wie Java, die dem Von-Neumann-Modell folgen, die vollständige Liste abdeckt. Zudem sollte nicht der Aspekt der Sprachentheorie in den Vordergrund gerückt werden. Dies hätte zur Konsequenz, dass weitere Sprachen auch unter dem Gesichtspunkt behandelt werden müssten, den Schülerinnen und Schülern die Unterschiede deutlich zu machen. Auch aufgrund der größtenteils lediglich zu reproduzierenden Inhalte sollte dieser Punkt in den Hintergrund gerückt werden.

3.2.3 Hardware eingebetteter Systeme

Um den Schülerinnen und Schülern deutlich zu machen, wie ein informationsverarbeitendes System in einen Kontext eingebettet wird, ist es vonnöten, die Grundprinzipien dieser Einbettung zu behandeln. Marwedel stellt dabei das Prinzip *Hardware in the loop* (siehe [Marwedel, 2007, S. 96]) vor. Diese Vorgehensweise der Eingabe, Verarbeitung und Ausgabe (EVA-Prinzip) ist so grundlegend, dass sie schon in vielen Bereichen der Informatik kennengelernt wurde und somit Anknüpfungspunkte zu bereits gelerntem Stoff schafft. Im Folgenden werden daher die im Buch angesprochenen Themen auf der Grundlage dieses Prinzips bewertet.

3.2.3.1 Eingabe Um eine Information, die sich aus der Umwelt des eingebetteten Systems ergibt, zu verarbeiten, muss diese Information erst für die informationsverarbeitende Einheit verwertbar gemacht werden. Marwedel beschreibt diesen Vorgang mittels Sensoren, *Sample-and-Hold*-Schaltungen und Analog/Digital (A/D)-Wandlern.

Mit den Sensoren verhält es sich ähnlich wie mit den Eigenschaften eingebetteter Systeme bzw. Anforderungen an Spezifikationssprachen. Der Lerninhalt wäre rein informativ, so dass im Zuge einer Leistungsmessung lediglich der Anforderungsbereich I bedient werden könnte. Eine tiefgreifende physikalische Betrachtung kommt allein schon aufgrund der Tatsache nicht in Betracht, dass in diesem ganzheitlichen Ansatz die informatiknahen Themen in den Vordergrund gerückt werden sollen.

Aus diesem Grund erscheint auch eine Behandlung der *Sample-and-Hold*-Schaltungen sowie der A/D-Wandler nicht sinnvoll. In der Tat ist eine Betrachtung von Kondensatoren und angelegter Spannung wenig hilfreich, um Prinzipien der Informatik zu vermitteln. Auch Baumann fordert, dass „technische Einzelheiten nicht Gegenstand des Unterrichts sein dürfen“ [Baumann, 1990, S. 267]:

> „[Es geht] um die Idee der Formalisierung, der Mechanisierung und der Automatisierung, die in der abendländischen Kulturgeschichte tief verwurzelt sind.“ [Baumann, 1990, S. 266].

Um diesem Anspruch gerecht zu werden, ist es erforderlich, dass die Schülerinnen und Schüler die Prinzipien verstehen, die hinter der Umwandlung von analogen in digitale Signale stehen, dabei aber gleichzeitig inhaltlich nicht zu sehr in die Physik abgewichen wird. Die Diskretisierung des Eingangssignals ist leicht zu motivieren. Für den Prozess der Umwandlung werden bei Marwedel der *Flash* A/D-Wandler und das Prinzip der sukzessiven Approximation vorgestellt. Dabei erscheint gerade das Prinzip der sukzessiven Approximation geeignet, unterrichtlich behandelt zu werden. Da es um das Prinzip als solches geht, ist nicht die Notwendigkeit gegeben, die konkrete technische Realisierung zu besprechen.

Für Kurse, die ein gutes Fundament aufweisen, was die Physik anbelangt, könnte es sogar sinnvoll sein, an dieser Stelle auch den *Flash* A/D-Wandler zu besprechen. Der Grund, warum gerade an dieser Stelle von der starren Linie abgewichen werden kann, die informatischen Aspekte in den Vordergrund zu rücken, besteht - so paradox es klingen mag - in der intensiveren Behandlung informatischer Aspekte. Denn nur wenn ein zweiter A/D-Wandler besprochen wird, ist die Betrachtung der Komplexität der Wandler motiviert. Die Komplexität nur einer Schaltung zu betrachten, ergibt keinen Sinn, da kein Vergleichswert vorhanden ist, der es den Schülerinnen und Schülern ermöglicht, begründet zu einem Urteil über das erlernte Prinzip zu kommen. So wird durch die O-Notation ein weiterer Anknüpfungspunkt an vorangegangene Unterrichtsmodule geschaffen.

3.2.3.2 Verarbeitung In diesem Abschnitt geht es um das Herz eingebetteter Systeme, die informationsverarbeitenden Einheiten und den Speicher. In der Buchvorlage werden drei Einheiten vorgestellt (siehe [Marwedel, 2007, S. 108-128]):

- Anwendungsspezifische integrierte Schaltkreise (*Application-Specific Integrated Circuits*, (ASICs))
- Rekonfigurierbare Logik
- Prozessoren

Eine unterrichtliche Thematisierung bedeutete, dass die Schülerinnen und Schüler die Merkmale und Eigenschaften dieser drei Typen lernen müssten. Gerade bei den Merkmalen von Prozessoren stellt sich dann die Frage, ob diese Tiefe im Stoff zielführend ist. Die Antwort dürfte vor dem Hintergrund des allgemeinbildenden Anspruchs an die in dieser Arbeit betrachtete Schulform nicht schwer fallen. Die Forderung nach Effizienz von Prozessoren ist noch vermittelbar, doch bringt die konkrete Realisierung - beispielsweise das Vorhandensein eines zweiten Befehlssatzes einiger ARM-Prozessoren - keine Erkenntnisse, die von allgemeiner Bedeutung für die Informatik wären. Hierbei handelt es sich vielmehr um spezielle Techniken.

Ebenso verhält es sich mit dem Speicher. Zwar ist die Forderung nach Energie-Effizienz[10] einsichtig, doch ist an eine intensive Behandlung von *Scratchpad*-Speichern in der Schule aufgrund der enormen Komplexität nicht zu denken.

Diese Tatsachen liefern nicht gerade gute Argumente für eine Behandlung des Themenbereiches Verarbeitung. Doch ist dieser Bereich der informationsverarbeitenden Einheiten so zentral für eingebettete Systeme, dass er in einem ganzheitlichen Ansatz vorkommen sollte. Eine Möglichkeit hierzu eröffnet Marwedel:

> *„[the] industry has difficulty finding adequately trained engineers, fully aware of design choices.“* [Marwedel, 2005, S. 25]

Dies ist ein Hinweis, sich darauf zu konzentrieren, begründet Entwurfsentscheidungen treffen zu können. Für die Schule ergibt sich daher die Möglichkeit, die Schülerinnen und Schüler zu befähigen, anhand von Anforderungen, die an das eingebettete System gestellt werden, auf der Basis der Eigenschaften der drei oben genannten informationsverarbeitenden Einheiten - insbesondere dem Kostenfaktor[11] - zu entscheiden, welche dieser Einheiten am besten geeignet erscheint, bei der Realisierung des geforderten Systems verwendet zu werden. Dies bedeutet insbesondere, dass die Unterschiede zwischen den Einheiten thematisiert werden. Die konkrete Realisierung, ebenso Themen wie Codegrößen-Effizienz, spielen keine Rolle.

3.2.3.3 Ausgabe Für die Ausgabe ist es zunächst notwendig, dass das digitale Ergebnis in ein analoges Signal umgewandelt wird, bevor es über Aktuatoren umgesetzt wird. Für den Unterricht bedeutet dies, dass dieser Prozess nachvollzogen werden müsste. Doch ergibt sich dabei ein Problem, das durch die von Marwedel formulierten Voraussetzungen entsteht, die nötig sind, um die Vorlesung verstehen

[10] Bei diesem Thema bietet sich auch eine weitere gesellschaftliche Diskussion an. Es kann im Unterricht der Frage nachgegangen werden, welchen Beitrag die Informatik - vor allem durch Energie-Effizienz - zum Umweltschutz leisten kann.

[11] Natürlich ist keine exakte Berechnung der Kosten gemeint. Auch hier geht es um eine übergeordente Sichtweise.

zu können (siehe [Marwedel, 2005, S. 27]). Unter anderem sollten die Kirchhoffschen Gesetze sowie Operationsverstärker in einem Physikkurs behandelt worden sein, um den Digital/Analog (D/A)-Wandler zu verstehen. Nun kann aber nicht davon ausgegangen werden, dass dieses Vorwissen bei allen Schülerinnen und Schülern vorhanden ist, gerade wenn bedacht wird, dass die Informatik nun als gleichwertiges Fach zu Naturwissenschaften angesehen wird. Im Gegensatz zu den verschiedenen A/D-Wandlern führte eine Thematisierung des D/A-Wandlers auch nicht auf weitere Fragestellungen der Informatik. Hierbei bestünde das Lernziel darin, zu verstehen, wie ein D/A-Wandler funktioniert. Dies ist aber eher dem Bereich der Physik zuzuordnen, der Informatik brächte es keinen Mehrwert.

Wie auch Marwedel formuliert „ist [es] unmöglich, einen Überblick über alle verfügbaren Aktuatoren zu geben“ [Marwedel, 2007, S. 133]. Demnach ist auch eine reine Auflistung im Unterricht nicht zielführend.

Für den Abschnitt der Ausgabe ergibt sich daher für die Schule aufgrund der aufgeführten Argumente eine Lücke. Es bleibt also der Bewertung (siehe 3.3) vorbehalten, welche Schlüsse hieraus gezogen werden können.

3.2.4 Eingebettete Betriebssysteme und Scheduling

Um einen breiten Überblick über eingebettete Systeme zu erhalten, ist es unerässlich, dass nicht nur die Hardware, sondern auch die Software genauer beleuchtet wird. Die im Buch präsentierten Bemerkungen zu *Middleware* werden im Rahmen dieser Untersuchung keine Rolle spielen, da sich auf die wesentlichen Prinzipien des Entwurfs eingebetteter Systeme konzentriert werden soll und daher Themen wie Echtzeit-Datenbanken oder spezielle Softwarepakete in zu detailierten Informationen mündeten, die für allgemeinbildende Schulen nicht von Bedeutung sind. Im Folgenden werden daher die Möglichkeiten untersucht, Echtzeitbetriebssysteme (*Real-Time Operating Systems*, (RTOS)) sowie Scheduling-Algorithmen im Unterricht zu behandeln.

3.2.4.1 Echtzeitbetriebsysteme In der Buchvorlage werden Echtzeitbetriebssysteme dahingehend behandelt, dass Eigenschaften und Anforderungen aufgelistet und verschiedene Möglichkeiten präsentiert werden, wie diese Systeme aufgebaut sein können. Die Behandlung von Betriebssystemen ist zwar durch den Lehrplan abgedeckt [MSW, 1999a, S. 58], doch betrifft dies die Standardbetriebssysteme und nicht die speziellen Echtzeitbetriebssysteme, die teilweise sehr unterschiedliche Anforderungen im Vergleich zu Standardsystemen besitzen.

Es stellt sich daher die Frage, ob diese Behandlung der verschiedenen Aus-

prägungen von Eigenschaften sowie den speziellen Voraussetzungen, die zudem immer auch vom Kontext der Anwendung abhängig sind, zielführend im Sinne eines ganzheitlichen Ansatzes ist. Vielmehr sollte man in Anlehnung an die didaktische Reduktion den Aspekt des Schedulings als wichtige Aufgabe von Echtzeitbetriebssystemen in den Vordergrund rücken. Dies könnte zwar dazu verleiten, dass die Fokussierung auf diesen Bereich eine Argumentation dahingehend nahelegt, dass zu spezielle Kenntnisse vermittelt werden, doch bietet das Scheduling-Problem positive Möglichkeiten, die im nächsten Unterkapitel näher erläutert werden und somit diese Reduktion rechtfertigen.

3.2.4.2 Scheduling-Algorithmen Es ergibt sich aus den im Unterricht besprochenen Beispielen und den daraus hergeleiteten Anforderungen, dass es Situationen gibt, in denen Berechnungen vom Computer - also gewisse Tasks - bis zu einem gewissen Zeitpunkt abgeschlossen sein müssen. Durch einfache Beispiele kann die Notwendigkeit von Scheduling-Algorithmen problemlos motiviert werden.[12] Eine detailierte Behandlung von Schranken von Ausführungszeiten wäre aber nicht zielführend, da das Hauptaugenmerk auf Scheduling-Verfahren liegen soll.

Es bietet sich inhaltlich an, bei der Behandlung der Scheduling-Algorithmen eine ähnliche Sichtweise zu verwenden, wie bei den Spezifikationssprachen. Es sollten die verschiedenen Algorithmen klassifiziert und dann punktuell Repräsentanten von einigen Klassen im Unterricht besprochen werden. Der Unterschied zum Vorgehen bei den Spezifikationssprachen liegt aber darin, dass bei den Scheduling-Algorithmen die Klassifikation anhand der Anforderungen vonstatten geht. Es müssen also die Eigenschaften erarbeitet werden, die eine Klassifikation ermöglichen. Dieser Punkt erschien bei den Spezifikationssprachen überflüssig.

Die im Buch vorgestellten Scheduling-Algorithmen bieten sich alle an, um in der Schule behandelt zu werden. Dies gilt auch für *Latest Deadline First* (LDF) als Vertreter für Schedulingverfahren bei aperiodischen abhängigen Tasks, da die verwendete Datenstruktur des Stapels zur Obligatorik des Lehrplans gehört (siehe [MSW, 2008]). Um einen breiten Überblick zu gewährleisten, sich aber nicht zu sehr in den Scheduling-Verfahren zu verlieren, sollte man sich im Unterricht auf zwei bis drei Verfahren beschränken, die in ihren Merkmalen so unterschiedlich sind, dass alle Eigenschaften[13] durch diese Algorithmen abgedeckt werden.

Ein methodischer Kommentar soll an dieser Stelle jedoch nicht gegeben werden, da es sich in der Praxis beweisen muss, ob zuerst die Eigenschaften erarbeitet werden

[12] Denkbar wären Kontrollanlagen in Atomkraftwerken.

[13] Gemeint sind natürlich die Eigenschaften periodisch/aperiodisch, präemptiv/nicht-präemptiv und statisch/dynamisch (siehe auch Abbildung 3).

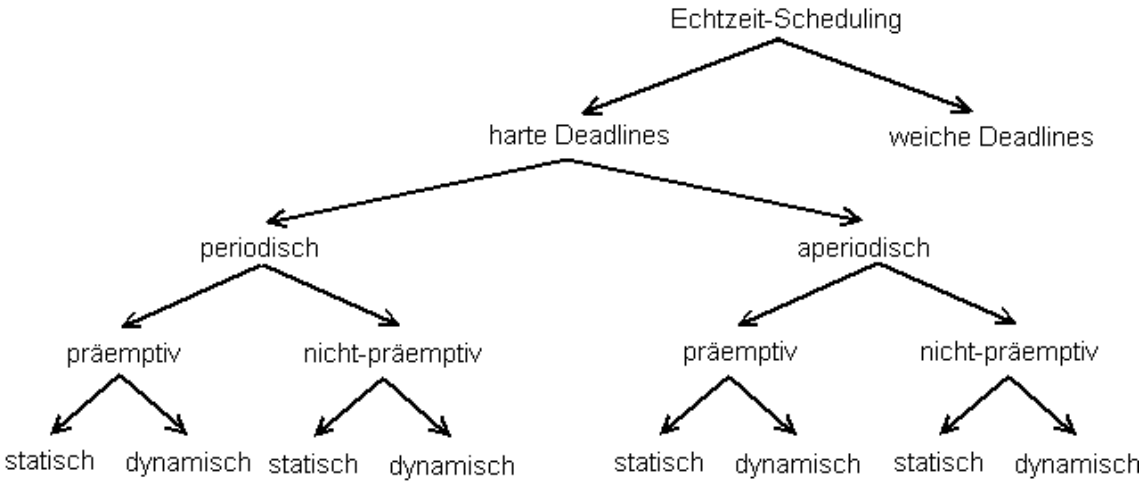

Abbildung 3: Klassen von Scheduling-Algorithmen

Quelle: [Marwedel, 2007, S. 138]

und anschließend eine Klassifikation vorgenommen wird oder ob zuerst Algorithmen vorgestellt werden, die dann anschließend auf ihre Eigenschaften hin analysiert werden. Abbildung 3 verdeutlicht die Übersicht über die Klassen von Scheduling-Algorithmen, wie sie im Unterricht erarbeitet werden sollten.

Im Lehrplan Informatik steht zwar, dass „das Ausführen eines Beweises, zu dem eigenständige Beweisgedanken erforderlich sind“ [MSW, 1999a, S. 86], ein legitimes Mittel ist, im Rahmen des Anforderungsbereiches III eingesetzt zu werden, doch erfordern einige Beweise über die Optimalität von Scheduling-Algorithmen ein grundlegendes Verständnis über Beweismethoden[14], die sowohl im Rahmen des Mathematikunterrichts als auch im Rahmen des Informatikunterrichts nicht Gegenstand des Unterrichts sind. Daraus resultiert die Empfehlung, sich in diesem Kapitel auf die Klassifikation und die Anwendung der Scheduling-Algorithmen zu konzentrieren. Für Kurse, die vor diesem ein Modul über theoretische Informatik absolviert haben, bietet sich hier ein Anknüpfungspunkt durch Komplexität von *Schedulability*-Tests an die theoretische Informatik an.[15]

Die Themen Prioritäts-Umkehr bzw. Prioritätsvererbung bieten die Möglichkeit, den Aspekt des gegenseitigen Ausschlusses, den Schubert und Schwill vorschlagen,

[14]Die Optimalität von *Earliest Due Date* und *Least Laxity* wird über Widerspruchsbeweise gezeigt.

[15]Es handelt sich dabei vielmehr um eine Bemerkung darüber, dass solche Tests oftmals NP-hart sind (siehe [Marwedel, 2007, S. 140-141]). Dies zu zeigen, ist sicher nicht Sinn und Zweck einer Unterrichtseinheit über eingebettete Systeme. Doch zeigt es, dass viele Aspekte der Informatik in dem Entwurf eingebetteter Systeme eine Rolle spielen, was wiederum der Legitimation dieses Unterrichtsmoduls zuträglich ist.

im Rahmen der Petri-Netze zu unterrichten (siehe dazu auch Kapitel 3.2.2.6), zu thematisieren. Diese Themen stellen zwar eine vertiefende Auseinandersetzung mit der Scheduling-Problematik dar, doch ist diese Klasse von Problemen innerhalb der maschinennahen Konzepte auch jetzt schon im Lehrplan verankert:

> „Insbesondere die Synchronisation von in der Maschine verteilt ablaufenden Prozessen und die Zuteilung von Ressourcen [...] an konkurrierende Anforderer lassen sich anhand entsprechender Algorithmen unterrichtlich aufarbeiten.“ [MSW, 1999a, S. 69]

Somit erhält dieses Thema auch von offizieller Seite die Legitimation für den Unterricht, was sich wiederum auf die Gesamtlegitimation eines Moduls über eingebettete Systeme in der gymnasialen Oberstufe auswirkt.

Aus didaktischer Sicht besteht zudem der Vorteil, dass auch zu diesem Thema eine Lernvisualisierung [Sirocic, 2008] existiert. Mit diesem Programm können verschiedene Scheduling-Verfahren simuliert und auch unmittelbar verglichen werden. Genau wie im Lernmodul über Kahn Prozessnetzwerke existiert auch in diesem Programm eine Anleitung, in der die Schülerinnen und Schüler Informationen und Hilfestellungen finden können.

3.2.5 Implementierung eingebetteter Systeme

Dieses Kapitel stellt innerhalb der Kurssequenz den zentralen Aspekt dar. Die Theorie der Hardware, Software und der Spezifikation eines eingebetteten Systems mündet schließlich in der Implementierung. Es werden die Verhaltensweisen der Systeme auf die entsprechende gewählte Plattform abgebildet. Hierin besteht auch die Schwierigkeit, wenn man versucht, diesen Teil des Entwurfs in der Schule unterrichtlich zu behandeln. Um ein funktionierendes eingebettetes System auch praktisch zu implementieren, bedarf es in der Regel weitreichenderer Kenntnisse, als sie in dem gewählten Ansatz, geschweige denn in der hier reduzierten Menge von potentiellen Unterrichtsinhalten vermittelt werden können. Die Auswirkungen auf den gesamten Ansatz sollen an dieser Stelle nicht besprochen werden. Dies passiert in der didaktischen Bewertung in Kapitel 3.3.

Klar ist, dass nicht alle Aspekte der Implementierung besprochen werden können. Somit ist auch hier wieder begründet eine Auswahl von Lerninhalten zu treffen. Das Problem ist, dass alle im Buch vorgestellten Techniken von der gewählten Plattform abhängen und somit eine breite Behandlung dieses Themas unmöglich erscheint. Die didaktische Reduktion nach Hering (siehe Kapitel 3.1.1) schlägt daher vor, unter den Elementen Gemeinsamkeiten zu finden, auf die generalisiert werden kann.

Die Implementierung der Systeme ist nicht nur unter dem Aspekt zu betrachten, dass es funktionsfähig ist. Vielmehr steht im Vordergrund, dass das System so effizient wie möglich implementiert wird. Alle im Buch vorgestellten Techniken haben zum Ziel, dass genau dieser Punkt der effizienten Implementierung erreicht wird, sei es die Optimierung hinsichtlich der Freigabe von nicht benötigten Ressourcen, Optimierungen auf hoher Abstraktionsebene (*High-Level*-Optimierungen), der effizienten Abbildung[16] von Prozessen auf Hardware bzw. Software oder optimierende Compiler.

Es ist daher konsequent, wenn das Hauptaugenmerk in der Schule nicht darauf gelegt wird, dass am Ende der Unterrichtsreihe jede Schülerin und jeder Schüler ein kleines eingebettetes System aus rekonfigurierbarer Logik bzw. Prozessoren erstellt hat, sondern der Gesichtspunkt der Optimierung betont wird. Der Lehrplan liefert hierzu eine gewisse Legitimation:

> „Zu den wesentlichen Arbeitsmethoden des Informatikunterrichts gehören das Testen, Korrigieren und Optimieren selbsterstellter Programme auf dem Rechner.“ [MSW, 1999a, S. 80]

Diese Vorgehensweise gilt aber nicht nur für die Arbeit am Rechner, sondern ist auch auf das Umgehen mit selbst erstellten Lösungen von Problemen zu verallgemeinern.

In dem Kapitel „Implementierung eingebetteter Systeme: Hardware-/Software-Codesign“ (siehe [Marwedel, 2007, S. 169-211]) werden die oben skizzierten Techniken näher erläutert. Im Folgenden werden diese Punkte dahingehend analysiert, ob sie den Kriterien Hubwiesers genügen und somit repräsentativ für die Vielzahl der Optimierungstechniken für die verschiedenen Plattformen sind.

3.2.5.1 Organisation der Nebenläufigkeit auf Taskebene Das Verschmelzen oder Aufteilen von Tasks, um eine bessere Auslastung der Ressourcen zu gewährleisten, erscheint auf den ersten Blick durchaus geeignet, um in der Schule behandelt zu werden. Der Grund hierfür ist die gut vermittelbare Möglichkeit, Ressourcen besser zu nutzen. Dazu muss aber als Voraussetzung ein Grundverständnis von exklusivem Zugriff auf Ressourcen vorhanden sein. Hinzu kommt, dass bei dieser Optimierungstechnik der praktische Nutzen nicht unmittelbar erfahrbar ist, da das System im Allgemeinen im Rahmen der Schule nicht implementiert werden kann. Nähere Ausführungen zu den Auswirkungen dieses Punktes werden in 3.3.2.2 gemacht.

Für komplexe Transformationen wird *FlowC*, eine Erweiterung der Sprache C, verwendet. Daher sind komplexe Transformationen nicht für die Schule geeignet. Es

[16] Hierbei sind auch Kostenfaktoren und nicht nur Schnelligkeit oder Energieeffizienz zu bedenken.

müssten einerseits Elemente der Sprache C neu erlernt werden, die allerdings nicht für das Zentralabitur zugelassen ist, andererseits müsste man sich mit der Erweiterung beschäftigen. Angesichts der geringen Bedeutung innerhalb der Gesamtsequenz ist dieser Zeitaufwand nicht zu rechtfertigen.

3.2.5.2 *High-Level*-Optimierungen Die Optimierungstechniken, die mit dem Begriff der *High-Level*-Optimierungen verbunden sind, sind aus guten Gründen nicht für die Schule geeignet. Zwar wird als eine Möglichkeit der Optimierung der Wechsel der internen Repräsentation von Zahlen vorgestellt, doch reicht es für die gymnasiale Oberstufe, wenn sich, wie im Lehrplan gefordert, darauf beschränkt wird, dass mit den manschineninternen Zahlendarstellungen umgegangen werden kann. Zudem werden Gleitkommazahlen nicht in jedem Informatikkurs behandelt. Diese zunächst nachzuholen, um dann anschließend die Umwandlung, bei der Informationen zwangsweise verloren gehen, zu thematisieren, wäre nicht zielführend.

Auch die Modifikationen am Programmcode durch Schleifentransformationen übersteigt den theoretischen Hintergrund[17], der in der Schule leistbar ist. Es besteht sogar die Gefahr bei einer unterrichtlichen Thematisierung solcher Verfahren, dass sich diese Techniken derart als Fehlvorstellung festsetzen, dass in Aufgaben, die nicht mit eingebetteten Systemen zu tun haben, diese besondere Behandlung der Schleifen mit dem Hinweis auf den optimierenden Charakter verwendet werden, was negative Auswirkungen auf den Programmierstil haben dürfte.

3.2.5.3 Hardware-/Software-Partitionierung In diesem Abschnitt wird ein mehrschrittiges Verfahren zur Optimierung behandelt. Im Zuge der Hardware-/Software-Partitionierung wird das Verhalten des Systems, das in einzelnen Prozessen spezifiziert wurde, auf Hardware bzw. Software abgebildet. Dabei kommt es darauf an, dies möglichst geschickt, also unter Berücksichtigung aller Nebenbedingungen wie Kosten und Laufzeit zu realisieren.

Sähe man das Partitionierungsverfahren in seiner Gänze für die Schule vor, entstünden die Probleme, dass bei der Anwendung Kenntnisse von VHDL, C, spezifischer Hardware und der Hierarchie in Taskgraphen abverlangt würden. Eine Verwendung von Tools, die diese Schritte übernehmen, ist nicht sinnvoll, da es nicht das Ziel des Informatikunterrichts ist, Anwenderwissen, sondern grundlegende Konzepte zu vermitteln.

Somit folgt, dass der einzige Punkt, der in diesem Kapitel eine Rolle spielen könnte, der der ganzzahligen Programmierung ist. Die ganzzahlige Programmierung

[17]So müsste beispielsweise die Speicheranordnung für zweidimensionale Felder in der Programmiersprache der Wahl bekannt sein.

ist eine „Standard-Optimierungstechnik aus dem *Operations Research*“ [Marwedel, 2007, S. 187] und besitzt daher eine gewisse Rechtfertigung, auch im Schulunterricht behandelt zu werden. Dabei bietet sich auch hier wieder ein Anknüpfungspunkt zur theoretischen Informatik, da dieses Problem vollständig ist für die Komplexitätsklasse **NP**. Doch muss bedacht werden, dass auch Marwedel aufgrund der unklaren Kommunikationszeiten davon spricht, dass dies zunächst nur eine Näherung bringt, die durch Modifikation und Iteration verbessert werden muss. Daher ist zu fragen, ob hierbei dann ein kleiner Teil herausgegriffen würde, der die Komplexität des Gesamtprozesses überdeckte.

Grundsätzlich erscheint die Thematisierung des Problems der ganzzahligen Programmierung für die Schule aus oben genanntem Grund geeignet. Es kann davon ausgegangen werden, dass Optimierungsaufgaben bereits im Mathematikunterricht behandelt wurden (siehe dazu den Lehrplan Mathematik [MSW, 1999b, S. 61]), sodass man sich im Informatikunterricht auf die Theorie dieser speziellen Optimierungstechnik konzentrieren könnte. Um nicht einen zu großen Schwerpunkt auf die Berechnung zu legen - was unter Umständen auch sehr viel Zeit in Anspruch nähme - können auch Programme[18] eingesetzt werden, die eine Lösung bestimmen. Es bleibt dabei aber noch zu diskutieren, ob dieses Verfahren so typisch für den Entwurf eingebetteter Systeme ist, dass es eine Rechtfertigung besitzt, in einem Unterrichtsmodul in der Schule eingesetzt werden. Diese Diskussion wird im Rahmen der didaktischen Bewertung in 3.3 geführt.

3.2.5.4 Compiler für eingebettete Systeme Diesem Themenbereich fehlt für die Behandlung in der Schule eine wichtige Voraussetzung. Die Schülerinnen und Schüler müssten sich, bevor sie sich mit Compilern für eingebettete Systeme beschäftigen, allgemeines Wissen über Compiler aneignen. Es kann nicht davon ausgegangen werden, dass diese Thematik im Vorfeld behandelt wurde. Hinzu kommt, dass die im Buch angeführten Beispiele auf Stoff aufbauen, der im Vorfeld schon als eher ungeeignet eingestuft wurde. So wird zwar im Rahmen der besseren Ausnutzung der Speicherhierarchie durch *Scratchpad*-Speicher wieder auf das Problem der ganzzahligen Programmierung reduziert, doch müssten hierfür diese Speicher im Unterricht behandelt worden sein (siehe dazu Kapitel 3.2.3.2). Zudem stellt die Anordnung von Variablen im Speicher eine so detailierte Ebene dar, dass sie nicht vereinbar wäre mit dem angestrebten Überblickswissen, das in diesem Modul in der Schule vermittelt werden soll. Aus diesem Grund sind auch Themen wie Compiler für

[18] Ein kostenloses Programm in englischer Sprache steht zum Download unter `http://sourceforge.net/project/showfiles.php?group_id=145213&package_id=159735&release_id=617581` [Stand: 2008-09-15] zur Verfügung.

spezielle Prozessoren zu streichen. Die allgemeine Bedeutung im Sinne Hubwiesers ist hier nicht gegeben.

3.2.5.5 Energie-Management Auch in diesem Abschnitt ist die Vermittelbarkeit des Stoffes nicht gewährleistet, da schon die Voraussetzungen im Unterricht nicht behandelt werden sollten. Dabei wird wieder einmal das große Problem bei der Implementierung eingebetteter Systeme ersichtlich. Die Optimierung wird anhand von verschiedenen Plattformen erläutert. Es ist somit keine globale für alle Plattformen geltende Optimierungsstrategie vermittelbar. Zwar könnten im Zuge einer Unterrichtseinheit über technische Informatik in der Jahrgangsstufe 11 die verschiedenen Energiesparzustände eines Prozessors besprochen werden, doch erscheint eine Optimierung hinsichtlich des Zeitpunkts eines Zustandswechsels auch für die Jahrgangsstufe 13 als nicht gewinnbringend, zumal die Erkenntnisse nicht unmittelbar ausprobiert und umgesetzt werden können.

3.2.6 Evaluierung und Validierung

Dieser Abschnitt soll nur kurz begründen, warum Evaluierung und Validierung, wie sie in der Buchvorlage präsentiert werden, nicht als Unterrichtsinhalt in einer allgemeinbildenden Schule geeignet sind. Evaluierung steht dafür, dass geprüft wird, ob der resultierende Entwurf den Anforderungen genügt, die zu Beginn an das eingebettete System gestellt wurden. Da in der Schule ohnehin keine komplexen Systeme entworfen werden können und die Anforderungen daher recht überschaubar sind, wird auch keine professionelle Methode wie die Pareto-Optimalität (siehe Literatur [Marwedel, 2007, S. 215-217]) benötigt.

Die Validierung prüft die Korrektheit in dem Sinne, dass der Entwurf auch wirklich das leistet, was von ihm erwartet wird. Bekannt für die Schülerinnen und Schüler ist die Methode des Testens. Bei der Erstellung von Software wird dieses Mittel mehr oder weniger strukturiert[19] durchgeführt. Methoden wie die formale Verifikation würden den ohnehin schon sehr hohen theoretischen Anteil in diesem Modul noch erhöhen, zumal hierzu ein Kurs über Logik vorauszusetzen ist. Da in der Vorlesung, die neben dem oft gegannten Buch als Vorlage dient, der Bereich der Evaluierung und Verifikation auch eher knapp behandelt wird, ist es nur konsequent, dass diese Themen auch in der Schule nicht beachtet werden.

[19]Eine strukturierte Methode in Java bestünde in der Verwendung von JUnit.

3.3 Strukturierung und Bewertung der ausgewählten Lerninhalte

Im ersten Durchlauf der didaktischen Reduktion ist deutlich geworden, dass sich theoretisch einige Inhalte gut und andere Inhalte weniger gut in der Schule umsetzen lassen. Um eine Diskussionsgrundlage zu schaffen, werden die in Kapitel 3.2 gewonnenen Erkenntnisse in Tabelle 1 zusammengefasst. Die Tabelle zeigt, welche Themen innerhalb eines Themenblocks auf der Basis der bisherigen Untersuchung prinzipiell geeignet erscheinen, in der Schule umgesetzt zu werden.

Dabei sind die Themenblöcke noch nach denen in der Buchvorlage ausgerichtet. Es wird sofort erkenntlich, dass ein ganzheitlicher Ansatz diese Blockung nicht verfolgen kann, da der Themenbereich Evaluation und Validation zur Gänze herausfällt und bei der Implementierung eingebetteter Systeme lediglich ein Thema übrig geblieben ist. Dies muss aber auch auch daraufhin überprüft werden, ob es für einen ganzheitlichen Ansatz einen Sinn ergibt, das Problem der ganzzahligen Programmierung als Standard-Optimierungstechnik überhaupt zu behandeln.

Bevor also der Versuch einer sinnvollen Anreihung der Unterrichtsinhalte vorgenommen wird, sollen die Themen in einem zweiten Durchlauf didaktisch dahingehend bewertet werden, welchen Wert sie innerhalb eines Moduls über eingebettete Systeme besitzen. Mit dem Wert ist auch gemeint, welche Bereiche der Informatik sich mit den entsprechenden Themen verdeutlichen lassen. Der Unterschied zum ersten Durchlauf bestehet darin, dass nun nicht die fachlichen Argumente mit den didaktischen kombiniert werden. Im ersten Durchlauf ging es um eine grobe Vorauswahl. Nun ist diese zu verfeinern und unter dem Aspekt eines in sich geschlossenen Moduls zu bewerten.

Wie schon im Verlauf der Arbeit deutlich geworden ist, gibt es bei einigen Themen und Übergängen Probleme, die gelöst werden müssen. Daher werden nun im Folgenden zunächst die Themen genauer analysiert, die im ersten Durchlauf keine theoretischen Probleme in der Umsetzung aufgewiesen haben. Im Anschluss daran werden verschiedene Problemfelder aufgezeigt und analysiert, bevor eine abschließende Bewertung hinsichtlich eines ganzheitlichem Ansatzes stattfindet.

3.3.1 Positive Aspekte ausgewählter Themen

In diesem Abschnitt geht es darum, die in Tabelle 1 aufgeführten Inhalte zu bewerten, einzuordnen und ggf. Verknüpfungsmöglichkeiten innerhalb des Moduls aufzuzeigen. Näher beleuchtet werden die Eigenschaften von eingebetteten Systemen, die Spezifikationssprachen und der Bereich des Schedulings.

Themenblock	**Themen**
Eigenschaften von eingebetteten Systemen	- Eingebettete Systeme im täglichen Leben - Definition und Eigenschaften
Spezifikationssprachen	- Berechnungsmodelle und Kommunikationsarten - *StateCharts* - Kahn Prozessnetzwerke
Hardware eingebetteter Systeme	- Das EVA-Prinzip - A/D-Wandler - ASICs, rekonfigurierbare Logik und Prozessoren
RTOS und Scheduling	- Scheduling als wichtige Aufgabe eines RTOS - Klassifikation von Scheduling-Algorithmen - Konkrete Algorithmen als Vertreter ausgewählter Klassen (*Rate Monotonic*, *Earliest Desdline First*, *Least Laxity*, *Latest Deadline First*, *Earliest Due Date*)
Implementierung eingebetteter Systeme	- *Integer Programming* als Standard-Optimierungstechnik
Evaluation und Validation	

Tabelle 1: Unterrichtsinhalte nach erster didaktischer Reduktion

Quelle: Eigene Darstellung

3.3.1.1 Eigenschaften von eingebetteten Systemen Sich mit Eigenschaften eingebetteter Systeme zu beschäftigen, resultierte aus den vorangegangenen Überlegungen, den Schülerinnen und Schülern so die Informatik in ihrer Umwelt bewusster werden zu lassen. Für die eingebetteten Systeme hat diese Motivation keinen fachlichen Nutzen. Das Bewusstmachen befähigt nicht unmittelbar, ein solches System zu entwerfen. Der Gehalt dieses Punktes liegt daher in dem allgemeinbildenden Charakter. Hubwieser hatte die Definition des Allgemeinbildungsbegriffs von Bussmann und Heymann zitiert. Ein weiterer Punkt, der im Rahmen der Bewertung in Kapitel 2.5.1 nicht genannt wurde, ist der der „Anleitung zum kritischen Vernunftgebrauch“ [Hubwieser, 2007, S. 57], in dessen Zuge Grundkenntnisse über „Funktionsweise von Rechenanlagen und Netzen“ [Hubwieser, 2007, S. 64] erworben werden müssten. Somit fällt diesem Bereich aufgrund des allgemeinbildenden Charakters ein hoher Wert innerhalb einer Unterrichtssequenz zu und sollte zur Motivation auch zu Beginn behandelt werden.

3.3.1.2 Spezifikationssprachen Der Bereich Spezifikationssprachen ist der für die Schule sicher attraktivste Punkt, da hier eine methodische Komponente geschult wird, die immer stärker in den Fokus des Informatikunterrichts gerückt wird:

> „[Es] gibt [...] einen Themenbereich der Informatik, der aufgrund seiner immensen Bedeutung für die Allgemeinbildung [...], in keinem Informatikunterricht übergangen werden kann. Es handelt sich um den Bereich der Modellierung [...]." [Hubwieser, 2007, S. 85]

Betrachtet man zudem die Ausführungen von Schubert und Schwill, was das Thema Spezifikation anbelangt, wird deutlich, dass den Spezifikationssprachen hierdurch eine weitere Legitimation, als Unterrichtsinhalt zu dienen, geliefert wird und diesem Bereich somit ein besonderes Gewicht innerhalb des Moduls zukommen muss:

> „[Die] Spezifikation bildet [...] die Grundlage [...], auf der man anschließend ein Modell bildet, also die Gegenstände der realen Welt analysiert, Beziehungen zwischen den Gegenständen aufdeckt, Operationen sucht und die Erkenntnisse schließlich in einer formalen Notation darstellt." [Schubert und Schwill, 2004, S. 158]

Es soll nun nicht weiter ins Detail gegangen werden, was die Modellierung anbelangt, da dieses Thema unerschöpflich scheint und in vielen anderen Arbeiten den Schwerpunkt bildet. Aus Sicht der Fachwissenschaft ergibt es einen Sinn, wenn die Spezifikation auf der Grundlage eines Wissens über Berechnungsmodelle und Kommunikationsarten stattfindet. Aus didaktischer Sicht muss aber darüber nachgedacht werden, ob diese hohe Abstraktionsstufe für alle Schülerinnen und Schüler geeignet ist. Auch unter dem zeitlichen Aspekt, der in 3.3.2.1 näher erläutert wird, empfiehlt es sich daher, den Stoff für Grund- und Leistungskurse zu unterscheiden.[20] *StateCharts* bieten sich auch aufgrund ihrer inhaltlichen Nähe zu UML an, für beide Kursarten zur Obligatorik gezählt zu werden. Kahn Prozessnetzwerke hingegen stellen eine Erweiterung der Sichtweise dar und verlangen somit den theoretischen Vorbau der Berechnungsmodelle und Kommunikationsarten. Sollte die Entscheidung nun dahingehend gefällt werden, dass der beschriebene theoretische Vorbau sowie die KPN dem Leistungskurs vorbehalten sein sollen, müssen die Auswirkungen auf das Gesamtbild im Grundkurs überprüft werden. Erst dann kann diese Entscheidung gerechtfertigt werden. Der Lehrplan des Landes Nordrhein-Westfalen charakterisiert die Grund- und Leistungskurse wie folgt:

> „Die Grundkurse repräsentieren das Lernniveau der gymnasialen Oberstufe unter dem Aspekt einer grundlegenden wissenschaftspropädeutischen Ausbildung.

[20] Diese Unterscheidung müsste in Zukunft auch den neu einsetzenden Strukturen, bestehend aus Kern-, Profil- und Neigungsfächern, angepasst werden. Siehe dazu auch den elften Punkt in `http://www.schulministerium.nrw.de/BP/Schulrecht/Gesetze/SchulG_Info/30_Argumente.html` [Stand: 2008-09-15].

> [...]
> Nicht die Stoffhäufung ist das Ziel der Leistungskurse, vielmehr muss auf der Grundlage gesicherter Kenntnisse das methodische Lernen im Vordergrund stehen.“ [MSW, 1999a, S. XV]

Es ist daher zu hinterfragen, welcher konkrete Unterschied zwischen Grund- und Leistungskursen generell im Fach Informatik angedacht ist und ob sich der Unterschied, der sich durch die oben vorgeschlagene Reduktion zwischen Grund- und Leistungskurs ergibt, in dieser Form durch den Lehrplan rechtfertigen lässt. Dass diese Reduktion gerechtfertigt ist, zeigt die folgende Aussage über die Ausprägungen eines Grund- und Leistungskurses:

> „[Grundkurse] sollen in grundlegende Fragestellungen, Sachverhalte, Problemkomplexe, Strukturen und Darstellungsformen eines Faches einführen [...].
> [...]
> [Leistungskurse] sind gerichtet auf eine systematische Beschäftigung mit wesentlichen, die Komplexität und Aspektreichtum des Faches verdeutlichenden Inhalten, Theorien und Modellen [...].“ [MSW, 1999a, S. 43]

Gerade der Aspektreichtum wird durch eine Behandlung von verschiedenen Berechnungsmodellen und Kommunikationsarten gefördert. Im Grundkurs hingegen reicht die Einführung in Problemkomplexe wie der Spezifikation eines eingebetteten Systems durch *StateCharts*. Daher erscheint diese Reduktion an dieser Stelle sehr sinnvoll.

3.3.1.3 Scheduling Um das Ziel zu erreichen, den Schülerinnen und Schülern die Funktionsweise dieser speziellen Informatiksysteme nahe zu bringen, gehört nach Schubert und Schwill [Schubert und Schwill, 2004, S. 269] auch, dass sich ein Grundverständnis von Systemen angeeignet wird. Als Schwerpunkte empfehlen sie Rechnerarchitektur, Betriebssysteme (hierbei unter anderem das Thema Prozesse) und Rechnernetze. Die Frage der Architektur wird im Rahmen des Moduls über eingebettete Systeme im Abschnitt der Spezifikation (3.3.1.2) nach obiger Argumentation zumindest im Leistungskurs behandelt. Für den Grundkurs bliebe immer noch das in vorangegangenen Unterrichtseinheiten besprochene Von-Neumann-Modell als Grundlage für diesen Bereich.

Zu dem Punkt Betriebssysteme kann ein Modul über eingebettete Systeme im Bereich der Prozesse viel zum Grundverständnis von Systemen beitragen. Schubert und Schwill [Schubert und Schwill, 2004, S. 270-272] empfehlen, im Zuge einer schrittweisen Entwicklung von Zusammenhängen auch Themen wie Parallelität und Konflikte zwischen Prozessen sowie geeignete Lösungsstrategien zu behandeln.

Hierbei kann das Scheduling in eingebetteten Systemen, die oftmals Echtzeitsysteme sind, dazu beitragen, den Blickwinkel der Schülerinnen und Schüler zu erweitern.

Somit besitzt dieses Thema aus didaktischer Sicht auch eine Legitimation, innerhalb dieses Moduls behandelt zu werden. Es bleibt aber die Frage offen, welchen Stellenwert und welche Möglichkeiten der Einbindung dieses Thema besitzt. Dies wird in Abschnitt 3.3.2 näher erläutert.

3.3.2 Problemfelder ausgewählter Themen

Es ist im Verlauf der Untersuchung deutlich geworden, dass nicht alle Themen des ursprünglichen universitären Ansatzes auch für die Schule geeignet sind. Die didaktische Reduktion hat dafür gesorgt, dass aus dem breiten Ansatz der TU Dortmund eine kleinere Menge von zunächt zusammenhangslosen Einzelthemen entstanden ist. Einige Vorteile der Themen sowie Versuche, die Gewichtung und Verknüpfungsmöglichkeiten zu bestimmen, wurden im letzten Unterkapitel erläutert. Doch sind durch die Reduktion viele Probleme entstanden, die es zu diskutieren gilt. Dabei geht es einerseits um bekannte Probleme wie den Stellenwert innerhalb des Moduls oder die Anknüpfungsmöglichkeiten, andererseits um noch nicht betrachtete Probleme wie den Zeitfaktor.

3.3.2.1 Zeitmangel Man muss sich auch noch eines Aspekts bewusst werden, der bisher in der Untersuchung kaum eine Rolle gespielt hat. Ein Modul über eingebettete Systeme kann nicht ein ganzes Jahr in Anspruch nehmen und muss daher so ausgerichtet sein, dass der Stoff innerhalb eines Zeitraums von etwa acht bis zehn Wochen zu absolvieren ist. Rechnet man für jedes Thema mit einer Woche - und es sei schon an dieser Stelle angemerkt, dass das völlig abwegig ist - müsste man für das gesamte Modul schon zwölf Wochen einplanen. Nun sind Themen wie Scheduling oder *StateCharts* nicht in einer Woche zu erarbeiten. Vor allem die Semantik der *StateCharts* oder auch die Gewöhnung an die Simulationsprogramme sorgen dafür, dass die Masse an Themen jeglichen verantwortbaren zeitlichen Rahmen sprengte. Das Problem ist natürlich, dass dies im Rahmen dieser Arbeit nicht nachgewiesen werden kann, da bisher noch kein Praxisversuch auf der Grundlage dieser Ergebnisse stattgefunden hat. Die Argumentation stützt sich daher auf Vergleiche zu anderen Themenfeldern, wie der Softwareentwicklung, bei der es erfahrungsgemäß auch einige Zeit in Anspruch nimmt, bis die Schülerinnen und Schüler die Entwicklungsumgebungen produktiv einsetzen können.

3.3.2.2 Implementierung Die konkrete Implementierung eines eingebetteten Systems stellt sich als das größte Problem dar, wenn man versucht, den Entwurf innerhalb eines Unterrichtsmoduls in der Schule unterzubringen. Es gibt sicher Gründe, die dafür sprechen, dass die Schülerinnen und Schüler nicht in der Lage sein müssen, das System, das sie entwerfen, tatsächlich zu bauen. So kommt es im Rahmen eines ganzheitlichen Ansatzes auch nicht darauf an, Fertigkeiten zu schulen. Vielmehr sind die Methoden und Prinzipien gefordert, die hinter dem Entwurf stehen.

Nun könnte man argumentieren, dass dies in der Gesamtheit dann einen zu theoretischen Kurs gebe, der keinerlei praktische Anteile innehabe. Unabhängig davon, ob dies der Wirklichkeit entspricht, ist dieses Ergebnis sogar durchaus erwünscht:

> „Ein richtig verstandener Informatikunterricht in der gymnasialen Oberstufe wird die Prinzipien und theoretischen Grundlagen herausarbeiten und diese stärker betonen als vordergründige Anwendungen, die heute wichtig erscheinen, morgen jedoch vielleicht schon überholt sind.“ [MSW, 1999a, S. 17]

Es soll demnach kein Wert darauf gelegt werden, dass den Schülerinnen und Schülern beigebracht wird, wie man beispielsweise rekonfigurierbare Bausteine so programmiert, dass sie das gewünschte Verhalten aus der Spezifikation zeigen.

Hubwieser steht dieser Argumentation aber kritisch gegenüber:

> „[Es ist] im Schulunterricht aus nahe liegenden didaktischen Gründen geboten,
>
> - die Modellierungstechniken zunächst einzeln einzuführen,
> - einzeln auf geeignete Probleme anzuwenden und
> - die erzeugten Modelle möglichst sofort zu implementieren.“ [Hubwieser, 2007, S. 90]

Besonders der dritte von Hubwieser angesprochene Punkt kann von dem hier hergeleiteten Unterrichtsmodul über eingebettete Systeme nicht geleistet werden. Bevor Vorschläge zur Lösung dieses Problems diskutiert werden, soll zunächst auf zwei weitere mit gerade erörterter Problematik zusammenhängende Schwierigkeiten in der Umsetzung hingewiesen werden.

Einerseits hat sich das Problem im Bereich der Hardware eingebetteter Systeme ergeben, dass das Prinzip der Eingabe, Verarbeitung und Ausgabe nicht zur Gänze schulgerecht umgesetzt werden kann, wenn man streng nach den Inhalten der Buchvorlage vorgeht. Der Bereich der Ausgabe ist sogar komplett vernachlässigt. Auch die Argumentationen, die die Eingabe in gewisser Weise noch rechtfertigen konnten, stellen sich im globalen Kontext nicht als zielführend hinsichtlich der Befähigung zu Designentscheidungen dar.

Andererseits besteht ein Problem darin, dass auf der Basis der Untersuchung bei der Implementierung nur ein und im Block Evaluation und Validation sogar kein Thema für die Schule geeignet erscheint. Die ganzzahlige Programmierung als Repräsentant für die Implementierung eingebetteter Systeme zu behandeln, hat eine gewisse Legitimation, wenn man den Aspekt der Optimierung betrachtet. Geht man aber von dem ganzheitlichen Standpunkt aus, so fällt ein Stilbruch dahingehend auf, dass von der globalen Sichtweise durch dieses Thema eine sehr detailierte Ebene betreten wird, die im Gesamtkontext nicht mehr zu vertreten ist. Bei der Evaluation und Validation sieht man deutlich, dass sogar die den Schülerinnen und Schülern so vertraute Praxis der Simulation nicht durchgeführt werden kann, da weder ein ausführbares Modell noch ein konkretes System im Rahmen eines ganzheitlichen Ansatzes konstruiert werden kann, das durch das Mittel der Simulation getestet werden könnte.

Für beide Probleme gibt es bereits Lösungen, die nun diskutiert werden sollen. Zur Entgegnung des Problems mit dem Prinzip *Hardware in the loop* könnten programmierbare Einheiten wie die LEGO® Mindstorms[21] verwendet werden. Die programmierbaren NXT-Einheiten können über verschiedene Sensoren Signale aus der Umwelt des Roboters aufnehmen. Die NXT-Einheit verarbeitet diese dann und reagiert über die Steuerung der Servo-Motoren. Somit ist scheinbar das Problem der komplizierten Theorie der Eingabe und Ausgabe überwunden.

Sollte die Entscheidung auf diese Variante fallen, besteht jedoch die Gefahr, dass die eigentlichen Prinzipien, die zu vermitteln versucht werden, nicht mehr im Vordergrund stehen und die Unterrichtseinheit somit in weiten Teilen zu einem Programmierkurs entartet. Hubwieser [Hubwieser, 2007, S. 87-90] führt hierzu einige Standpunkte an, die die kontroverse Diskussion um den Stellenwert der Programmierung innerhalb des Informatikunterrichts verdeutlichen. Im Sinne einer ganzheitlichen Ausbildung im Entwurf eingebetteter Systeme sollte aber nicht die Programmierung von informationsverarbeitenden Einheiten einer speziellen Firma im Vordergrund stehen, sondern es sollte auch hier darauf geachtet werden, dass man sich auf Prinzipien beschränkt, um der Ganzheitlichkeit gerecht zu werden. Ähnlich argumentiert Hubwieser, indem er zwei Voraussetzungen anführt, unter denen Programmierung im Informatikunterricht wesentlich erscheint:

- „Es muss sich [...] wirklich um die Implementierung eines vorher entwickelten Modells handeln [...].
- Die Syntax der verwendeten Sprache darf nicht in den Vordergrund treten. [...] Ein „Programmierkurs“, der von Sprach- anstatt von Problemstrukturen

[21] Weitergehende Informationen sind auf der offiziellen Website `http://mindstorms.lego.com` [Stand: 2008-09-15] verfügbar.

ausgeht, ist im Pflichtfachbereich fehl am Platze.“ [Hubwieser, 2007, S. 88-89]

Es soll dieser Variante aber nicht abgesprochen werden, dass es unterrichtliche Möglichkeiten durch Schwerpunktsetzungen gibt, die dafür sorgen, dass den Bedenken Hubwiesers entgegengewirkt werden kann. Doch ist es nicht Teil dieser Arbeit, für konkrete Themen Konzepte zu entwickeln. Es sei aber darauf hingewiesen, dass eine Konzeption für den Bereich der Hardware eingebetteter Systeme diesen Grundsatz Hubwiesers beachten sollte.

Für das Problem der Implementierung eines spezifizierten Systems beschreibt Vahid [Vahid und Givargis, 2002] zwei Möglichkeiten, wie man eine Spezifikation mit Zustandsautomaten in einem sequentiellen Programm realisiert. Die konkrete Umsetzung auf diese Art und Weise ist natürlich wieder nur eine Möglichkeit und hat keinen Anspruch darauf, als allgemeingültiges Mittel für alle Plattformen zu gelten. Doch bestünde ein Vorteil darin, der konkreten Realisierung einen Schritt näher gekommen zu sein und somit eine neue Ebene der Spezifikation zu erreichen. Die verschiedenen Ebenen werden im Buch von Marwedel [Marwedel, 2007, S. 92] in Abb. 2.65 illustriert.

Eine Möglichkeit bestünde darin, ein zusätzliches Programm zu installieren, das die Konvertierung übernimmt und Code erzeugt, der die gleiche Funktionalität aufweist, wie das durch *StateCharts* spezifizierte System. Vahid sieht bei diesem Lösungsversuch einen Nachteil:

> *„The drawback of this approach is that we must support yet another tool, which includes additional licensing costs, version upgrades, training, integration problems with our development environment, and so on.“* [Vahid und Givargis, 2002, S. 216]

Da es in der Schule nach Hubwieser nicht um Anwendungsschulung gehen soll, fällt diesem Ansatz - auch aufgrund des Kostenarguments - keine Bedeutung zu.

Vahid beschreibt aber noch einen zweiten Ansatz, den es zu diskutieren gilt:

> *„[. . .] we can use a language subset approach. In this approach, we directly capture our state machine model in a sequential program language, by following a strict set of rules for capturing each state machine construct in an equivalent set of sequential program constructs.“* [Vahid und Givargis, 2002, S. 216]

Hierzu bestehen aber zwei Bedenken. Es gilt hier das Gleiche wie für die LEGO® Mindstorms. Es muss darauf geachtet werden, nicht den Anschein eines Programmierkurses mit geändertem Kontext aufkommen zu lassen. Falls dieser Weg gegangen werden sollte, muss eine Diskussion über die didaktischen Konsequenzen stattfinden, die der von Vahid vorgeschlagene Weg mit sich bringt:

> *„We [...] create an infinite loop, containing a single switch statement that branches to the case corresponding to the value of the state variable.“* [Vahid und Givargis, 2002, S. 217]

Bei dieser Vorgehensweise ist es zumindest bedenklich, den Schülerinnen und Schülern zu zeigen, dass die Programmierung mit Endlosschleifen ein gängiges Vorgehen zu sein scheint. Es besteht die Gefahr der Bildung einer Fehlvorstellung und der negativen Beeinflussung des Entwicklungsstils auf der Ebene der Programmierung.

3.3.3 Abschließende Bewertung

Um eine abschließende Bewertung abgeben zu können, muss man sich die Vorzeichen ins Gedächtnis rufen, unter denen die Vorlesung, die der Untersuchung zugrunde lag, analysiert wurde. Es sollte ein ganzheitlicher Ansatz entwickelt werden, also ein Modul, das in sich geschlossen als Alternative zu den bisher bestehenden Modulen im Zentralabitur in Nordrhein-Westfalen absolviert werden kann.

Im Verlauf der didaktischen Reduktion sind viele positive Aspekte herausgestellt worden, die den Schülerinnen und Schülern einen verantwortungsvolleren Umgang mit und ein besseres Verständnis von Informatiksystemen ermöglichen könnten. Zudem bestünde durch dieses Modul die Gelegenheit, weitere Modellierungstechniken kennenzulernen und im Kontext realer Systeme anzuwenden. Dabei ist vorteilhaft, dass der Blickpunkt stets auf dem Verhalten des Systems liegt und beispielsweise Datenhaltungsaspekte eine untergeordnete Rolle spielen.

Abschließend muss man aber zur Kenntnis nehmen, dass im Sinne der Ganzheitlichkeit das Modul nicht umgesetzt werden kann. Den beschriebenen Problemfeldern kann zwar mit Bedenken in gewisser Weise entgegengewirkt werden, doch fehlt dabei die Leitlinie, an der sich die Themen positionieren. Marwedel benutzt dazu ein Strukturelement (Abb. 1.5 [Marwedel, 2007, S. 11]), das die Position und den Wert eines Themenbereichs innerhalb der Vorlesung transparent aufzeigt. Für einen ganzheitlichen Ansatz müsste ein ähnliches Vorgehen von der Spezifikation über das Wissen um Hard- und Software bis hin zur Implementierung und Validation angestrebt werden. Die Elemente, die im Rahmen der Schule möglich sind, weisen aber keine größeren Zusammenhänge auf. Was in der Spezifikation noch gelingt - die Schaffung eines breiten Überblicks - ist bei der Thematisierung von Hard- und Software kaum noch möglich. Hier verstrickt man sich in Spezialfällen, deren praktischer Nutzen kaum ersichtlich wird, da die Implementierung einer Spezifikation ein großes Problem darstellt (siehe dazu Abschnitt 3.3.2.2). Durch diese begrenzten Möglichkeiten, ein eingebettetes System in Hard- und Software zu implementieren, stellt sich Schülerinnen und Schülern die Frage, welchen Nutzen das zuvor

Gelernte innehat. Beispielsweise haben die Unterrichtsstunden, die man für Scheduling aufgewendet hat, in der von Vahid beschriebenen Methode zur Umsetzung von *StateCharts* in sequenzielle Programme keine Bedeutung. Ohne einen sinnvollen Zusammenhang zwischen den Themen, der auf der Grundlage dieser Arbeit nicht gegeben zu sein scheint, ist ein eigenständiges Modul über eingebettete Systeme nicht zu konzipieren.

Es sei aber angemerkt, dass die positiven Aspekte Anlass dazu geben, sich Gedanken darüber zu machen, wie man sie in bereits bestehenden Modulen integrieren kann. Diese Arbeit stellte den Versuch dar, ein eigenständiges Modul zu begründen. Streicht man diesen Anspruch, so ergibt sich durch den Entwurf eingebetteter Systeme ein Potential, das dem bestehenden Modulkatalog zuträglich sein könnte. Eine Untersuchung dessen sprengte aber den Rahmen dieser Arbeit, weshalb weitere Ansätze, die das Ziel haben, den Entwurf eingebetteter Systeme in der Schule unterrichtlich zu thematisieren, notwendig erscheinen.

3.4 Zusammenfassung

In diesem Kapitel wurde auf der Basis des Ansatzes der TU Dortmund, was die Lehre in eingebetteten Systemen anbelangt, eine Menge von Themen bestimmt, die potentiell als Inhalte in einem Modul über eingebettete Systeme im Rahmen des Abiturs gelehrt werden könnten. Dabei lag die Priorität darauf, dass dieses Modul eigenständig und in sich abgeschlossen sein sollte.

Als Methoden, die geeigneten Themenbereiche aus dem Ansatz aus Dortmund auszuwählen, wurden die didaktische Reduktion nach Hering sowie die Auswahlkriterien Hubwiesers vorgestellt. Die didaktische Reduktion gibt die prinzipiellen Möglichkeiten vor, wie eine große Menge von Aussagen sinnvoll verkleinert werden kann, während Hubwieser konkrete Kriterien vorgibt, nach denen man die Themen auswählt, auf die reduziert wird.

Im Anschluss daran erfolgte eine Diskussion über die einzelnen Inhalte, die in der Vorlesung gelehrt werden. Hierbei kamen die vorher vorgestellten Methoden zum Einsatz. Durch verschiedene Argumentationsketten wurde bei den diversen Themen hergeleitet, ob und in welcher Form sie für die Schule geeignet zu sein scheinen. Dabei wurden sowohl fachliche als auch didaktische Argumente verwendet.

Nach dem ersten Durchlauf erfolgte die Strukturierung und Bewertung der ausgewählten Lerninhalte. Die Strukturierung hatte als Ergebnis, dass es sowohl positive als auch negative Aspekte in der Menge der Inhalte gab. Die anschließende Bewertung stellte diese Aspekte unter neuen didaktischen Aspekten dar.

Positiv ist die Möglichkeit, den Schülerinnen und Schülern zu einem bewuss-

teren Umgehen und vertieftem Verständnis von Informatiksystemen in ihrer Umwelt zu befähigen. Ein wesentliches Problem stellt sich aber in der Anordnung und Verknüpfung der einzelnen Themen dar, was zu dem Schluss führte, dass ein ganzheitliches Modul nicht im Rahmen der gymnasialen Oberstufe unterrichtet werden kann.

4 Schluss

Zum Abschluss dieser Arbeit soll zunächst eine Überprüfung dessen durchgeführt werden, was als Zielsetzung in der Einleitung festgelegt wurde. Dabei spielen inhaltliche Fragen eine ebenso wichtige Rolle wie die der Methodik. Zudem soll noch die Entscheidung reflektiert werden, das Thema einzuschränken. Die Gründe hierfür wurden in der Einleitung erwähnt und sollen an dieser Stelle noch einmal bewertet werden.

Da diese Arbeit auch durch die Einschränkungen einen recht speziellen Bereich untersucht hat, werden im Anschluss Möglichkeiten aufgezeigt, wie im Rahmen von weiteren Hausarbeiten dieses Thema aus einem anderen Blickwinkel betrachtet oder sogar fortgeführt werden kann. Dabei ist immer auch im Hinterkopf zu behalten, dass es unvermeidlich und eigentlich sogar gewünscht ist, dass verschiedene Sichtweisen existieren, die verschiedene Arten der Behandlung der Thematik ermöglichen.

4.1 Reflexion hinsichtlich der Zielsetzung

Zu Beginn dieser Arbeit wurden einge Ziele formuliert, die einerseits diese Arbeit abgrenzen, andererseits ermöglichen sollten, dass im Rahmen dieser Arbeit punktuell in die Tiefe gegangen werden konnte. Am Ende sollte entweder eine begründet hergeleitete Menge von Unterrichtsinhalten, die untereinander verknüpft und an einem erkennbaren roten Faden aufgereiht sind oder die Erkenntnis stehen, dass der Entwurf eingebetteter Systeme in der in dieser Arbeit intendierten Form nicht umzusetzen ist. Das Ergebnis war, dass es nicht möglich erscheint, ein eigenständiges Modul über eingebettete Systeme in der gymnasialen Oberstufe anzubieten. Das bedeutet aber weder, dass eingebettete Systeme in keiner Form in der Schule unterrichtet werden können, noch, dass diese Arbeit ihr Ziel nicht erreicht hat.

Inhaltlich lag der Fokus darauf, einen ganzheitlichen Ansatz zu verfolgen, also zu überprüfen, ob die Voraussetzungen dafür gegeben sind, ein eigenständiges Modul in der Oberstufe unterzubringen, das zu den bereits bestehenden Modulen gleichwertig ist. Nur unter dieser Voraussetzung ist das Resultat zu verstehen. In dieser Arbeit wurde nicht explizit untersucht, welche Möglichkeiten sich für eingebettete Systeme bieten, innerhalb von anderen Modulen des Informatikunterrichts thematisiert zu werden. Daher wurden die inhaltlichen Ziele im Sinne der Themenstellung erreicht.

Methodisch betrachtet sollte durch nachvollziehbare Argumentationsketten ein begründetes Urteil darüber gebildet werden, welche Elemente des Entwurfs eingebetteter Systeme für den Schulunterricht im Sinne der Zielsetzung dieser Arbeit geeignet zu sein scheinen. Es wurde versucht, dabei stets verschiedene Sichtweisen

zu berücksichtigen, die verschiedene Argumente für oder gegen einen Sachverhalt hervorbrachten, die es gegeneinander abzuwägen galt. Es war nicht möglich, bei allen Themen alle Argumente aufzulisten und gegeneinander abzuwägen, so dass ein lückenloses und vollständiges Bild entstand. Dies ist auch daher nicht möglich, da es immer verschiedene Sichtweisen gibt, die dafür sorgen, dass Argumente verschieden gewichtet werden. Diese Arbeit hat sich aber stets um Objektivität bemüht. Daher erscheint das Ziel der begründeten Argumentation im Rahmen dieser Arbeit als erreicht.

Allgemein ist zu sagen, dass die Einschränkung des Themas auf den Bereich der gymnasialen Oberstufe in Nordrhein-Westfalen sinnvoll war, da sich viele Argumentationen auf den bestehenden Lehrplan im Fach Informatik bezogen. Wäre das Thema nicht eingeschränkt geworden, hätte dies zur Auswirkung gehabt, dass die gegebenen Argumentationen hinsichtlich aller Lehrpläne hätten überprüft werden müssen und so die Themen auf einer zu detaillierten und ausführlichen Ebene hätten besprochen werden müssen. Die Einschränkung darauf, nur einen ganzheitlichen Ansatz zu verfolgen, erscheint auch im Nachhinein weiter sinnvoll, da bei einer Untersuchung sowohl eines ganzheitlichen Ansatzes als auch der Möglichkeiten, Elemente in anderen Modulen zu thematisieren, die Untersuchungs- und Argumentationstiefe gelitten hätte.

4.2 Anknüpfungsmöglichkeiten

Wie schon erwähnt wurde, konnte im Rahmen dieser Arbeit nur ein spezieller Bereich untersucht werden. Daher bieten sich einige Möglichkeiten, die Fragestellung unter einem anderen Aspekt zu untersuchen oder die Ergebnisse dieser Arbeit aufzugreifen, um weiterführende Ideen zu verfolgen. Die eigentliche Untersuchung, die in Kapitel 3 durchgeführt wurde, basierte auf dem Vergleich von drei verschiedenen Ansätzen der Lehre von eingebetteten Systemen an verschiedenen Hochschulen. Die Begründung für eben diese drei Hochschulen bestand darin, dass sie aufzeigten, welche verschiedenen Sichtweisen teilweise vertreten werden. Nun ist es aber durchaus legitim, andere Hochschulen für einen solchen Vergleich heranzuziehen. Es besteht dabei die Möglichkeit, dass unter den gleichen Voraussetzungen, einen ganzheitlichen Ansatz verfolgen zu wollen, eine andere Hochschule präferiert wird, als Grundlage für weitere Untersuchungen zu dienen. Dabei enstünde womöglich eine andere Menge von potentiellen Unterrichtsinhalten, da sich diese Untersuchung sehr stark auf das der ausgewählten Vorlesung zugrundeliegende Begleitmaterial als Quelle bezogen hat. Es wurde ja schon erwähnt, dass die Argumentationsketten trotz aller Bemühungen um Objektivität durchaus in gewissem Maße subjektiv sind, da

das Ziel durch die Brille der Informatik betrachtet wurde. Daher ist es möglich, im Rahmen einer ähnlichen Untersuchung zu anderen Ergebnissen zu kommen.

Möchte man jedoch das Ergebnis dieser Untersuchung fortsetzen, so bietet sich an, den Entwurf eingebetteter Systeme daraufhin zu analysieren, ob Themen, die beispielsweise die Modellierung besonders hervorheben, in anderen bestehenden Modulen integriert werden können. Dann stünde dies zwar unter dem Aspekt, dass eingebettete Systeme nicht Hauptgegenstand des Unterrichts sind, sondern als Beispiele für andere Prinzipien dienen, doch bleibt die Relevanz des gesamten Themenbereiches bestehen, so dass diese Form der Thematisierung durchaus auch eine Legitimation hat.

Literatur

[ARTIST, 2003] ARTIST network of excellence: Guidelines for a Graduate Curriculum on Embedded Software and Systems, 2003. Online im Internet: `http://www.artist-embedded.org/docs/Publications/Education.pdf` [Stand: 2008-08-01].

[Baumann, 1990] Rüdeger Baumann: *Didaktik der Informatik*, Stuttgart, Klett-Schulbuchverlag, 1990.

[BITKOM, 2007] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM): *Positionspapier - Standortnachteil Fachkräftemangel: Fakten und Lösungsansätze*, 2007. Online im Internet: `http://www.bitkom.org/files/documents/BITKOM_Positionspapier_Fachkraeftemangel_IT-Gipfel_2007.pdf` [Stand: 2008-09-18].

[BITKOM, 2008] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BITKOM): *Presseinformation - „Eingebettete Systeme“ – die Hidden Champions der Industrie*, April 2008. Online im Internet: `http://www.bitkom.org/files/documents/BITKOM_Presseinfo_Embedded_Systems_21_04_2008.pdf` [Stand: 2008-09-17].

[GI, 2000] Fachausschuss „Informatische Bildung in Schulen“ der Gesellschaft für Informatik e.V. (GI): Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen, 2000. Online im Internet: `http://www.gi-ev.de/fileadmin/redaktion/empfehlungen/gesamtkonzept_26_9_2000.pdf` [Stand: 2008-08-01].

[GI und BITKOM, 2007] Gesellschaft für Informatik e. V. (GI) und der Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (BIT KOM): Nachwuchs für die Informationsgesellschaft! – Plädoyer für eine zukunftsorientierte Schulbildung, *LOG IN* Heft Nr. 146/147, S. 11, 2007. Online im Internet: `http://www.gi-informatiklehrer.de/LogInBitkom.pdf` [Stand: 2008-08-01].

[Grimheden und Törngren, 2005] Martin Grimheden und Martin Törngren: How should embedded systems be taught? Experiences and snapshots from Swedish higher engineering education, *ACM SIGBED Review* 2(4): 34–39, Oktober 2005.

[Hartmann *et al.*, 2006] Werner Hartmann, Michael Näf und Raimond Reichert: *Informatikunterricht planen und durchführen*. Berlin, Springer, 2006.

[Hubwieser, 2007] Peter Hubwieser: *Didaktik der Informatik*, 3. überarbeitete und erweiterte Auflage, Berlin, Springer, 2007.

[Humbert, 2005] Ludger Humbert: *Didaktik der Informatik mit praxiserprobtem Unterrichtsmaterial*, 2. überarbeitete und erweiterte Auflage, Wiesbaden, Teubner, 2006.

[Marwedel, 2005] Peter Marwedel: Towards laying common grounds for embedded system design education, *ACM SIGBED Review* 2(4):25–28, Oktober 2005.

[Marwedel, 2007] Peter Marwedel: *Eingebettete Systeme*, Berlin, Springer, 2007.

[Modrow, 1991] Eckart Modrow: *Zur Didaktik des Informatik-Unterrichts*, Bonn, Dümmlers Verlag, 1991.

[MSW, 1999a] Ministerium für Schule, Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen, Hg.: *Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium/Gesamtschule in Nordrhein-Westfalen: Informatik*. Frechen, Ritterbach-Verlag GmbH, 1999. Online im Internet: `http://www.ritterbach.de/lp_online/4725.pdf` [Stand: 2008-08-28].

[MSW, 1999b] Ministerium für Schule, Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen, Hg.: *Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium/Gesamtschule in Nordrhein-Westfalen: Mathematik*. Frechen, Ritterbach-Verlag GmbH, 1999. Online im Internet: `http://www.ritterbach.de/lp_online/4720.pdf` [Stand: 2008-09-09].

[MSW, 2007] Ministerium für Schule, Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen, Hg.: *Vorgaben zu den unterrichtlichen Voraussetzungen für die schriftlichen Prüfungen im Abitur in der gymnasialen Oberstufe im Jahr 2010, Vorgaben für das Fach Informatik*, 2007. Online im Internet: `http://www.standardsicherung.schulministerium.nrw.de/abitur-gost/getfile.php?file=1067` [Stand: 2008-08-28].

[MSW, 2008] Ministerium für Schule, Weiterbildung, Wissenschaft und Forschung des Landes Nordrhein-Westfalen, Hg.: *Vorgaben zu den unterrichtlichen Voraussetzungen für die schriftlichen Prüfungen im Abitur in der gymnasialen Oberstufe im Jahr 2011, Vorgaben für das Fach Informatik*, 2008. Online im Internet: `http://www.standardsicherung.schulministerium.nrw.de/abitur-gost/getfile.php?file=1067` [Stand: 2008-08-28].

[Rösler und Schmidkunz, 1996] Horst Friedrich Rösler und Heinz Schmidkunz: Die didaktische Reduktion - Eine Bestandsaufnahme. *NiU-Chemie* 7: 4-5, 1996.

[Sangiovanni-Vincentelli und Pinto, 2005] Alberto Luigi Sangiovanni-Vincentelli und Alessandro Pinto: Embedded system education: a new paradigm for engineering schools?, *ACM SIGBED Review* 2(4): 5–14, Oktober 2005.

[Schubert und Schwill, 2004] Sigrid Schubert und Andreas Schwill: *Didaktik der Informatik*. Wiesbaden, Spektrum Akademischer Verlag, 2004.

[Schwill, 1993] Andreas Schwill: Fundamentale Ideen der Informatik. *Zentralblatt für Didaktik der Mathematik* 25(1): 20-31, Februar 1993.

[Sirocic, 2007] Birgit Sirocic: *Lernmodul Kahn Prozessnetzwerke*, Version 1.0.1 vom 14.11.2007. Online im Internet: `http://ls12-www.cs.tu-dortmund.de/edu/ES/leviKPN.zip` [Stand: 2008-09-15].

[Sirocic, 2008] Birgit Sirocic: *Lernmodul Real Time Scheduling*, Version 0.59 vom 07.1.2008. Online im Internet: `http://ls12-www.cs.tu-dortmund.de/edu/ES/leviRTS.zip` [Stand: 2008-09-15].

[Vahid und Givargis, 2002] Frank Vahid und Tony Givargis: *Embedded System Design: A Unified Hardware/Software Introduction*, Hoboken, Wiley & Sons, 2002.