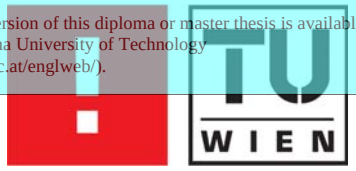


FAKULTÄT FÜR !NFORMATIK

Lehre von Software-Verifikation in der Berufsbildung unter dem Aspekt von Bildungsstandards

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Informatikmanagement

eingereicht von

Thomas Reinbacher, MSc
Matrikelnummer 0828472

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Futschek
Mitwirkung: Dir. Dipl.-Ing. Mag. Dr.phil. Martin Weissenböck

Wien, 29.4.2010

(Unterschrift Verfasser) (Unterschrift Betreuer)

Eidesstattliche Erklärung

Thomas Reinbacher, MSc, Weidenweg 44, A-2020 Kleinstelzendorf

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Ort, Datum	Unterschrift

Kurzfassung

Kontext: Bildungsstandards, Kompetenzmodelle, Qualifikationsrahmen sind die Schlagwörter der jüngsten Bildungsdiskussionen. Eine Vergleichbarkeit zwischen den EU Mitgliedstaaten und eine eindeutige Kategorisierung der erworbenen Kompetenzen ist das erklärte Ziel. Mit dem Projekt Bildungsstandards in der Berufsbildung des Bundesministeriums für Unterricht, Kunst und Kultur (bm:ukk) sollen in einem ersten Schritt transparente Darstellungsformen von Lernergebnissen sowie Systemvergleichbarkeit erarbeitet werden. Einzelne Arbeitsgruppen arbeiten konkrete Kompetenzmodelle aus und definieren somit Fähigkeiten in verschiedenen Fachgebieten, die ein künftiger (z.B. HTL) Absolvent beherrschen soll.

Wissenschaftliche Fragestellung: Wie kann die Informatikausbildung in der Höheren Technischen Lehranstalt (HTL) um Lehrinhalte aus dem Themenkreis (formale) Verifikation von Software und Systemen unter dem Aspekt von Bildungsstandards bereichert werden?

Wissenschaftliche Methode: Mithilfe von qualitativen Interviews wird der IST Zustand der Informatikausbildung in der HTL Hollabrunn erhoben. Weitere Interviews bringen die Anforderungen der Industrie an künftige HTL Absolventen in Erfahrung. Die in der Arbeit ausgearbeiteten Unterrichtsblöcke wurden in einer Experten Feedback Loop unter Mitwirkung von Dir. Dipl.-Ing. Mag. Dr.phil. Martin Weissenböck (Direktor HTL Rennweg 3) und Dipl.-Ing. Dr. Walter Führer (Informatik Lehrender HTL Hollabrunn) entwickelt. Eine umfassende Literaturrecherche stellt die theoretische Basis der Arbeit dar.

Zentrale Ergebnisse: In der industriellen Praxis ist weitläufig bekannt, dass Verifikation einen Großteil der Ressourcen in modernen Softwareprojekten in Anspruch nimmt. Im gegenwärtigen Informatikunterricht finden sich aber nur äußerst selten Themen der (formalen) Softwareverifikation wieder. Diese Vermutungen werden durch die durchgeführten Interviews mit Lehrenden als auch mit Personalverantwortlichen der Industrie bestätigt. Es stellt sich heraus, dass es ein signifikantes Ungleichgewicht zwischen Lehre und den Anforderungen der Industrie gibt. Die vorliegende Arbeit zeigt erstmals eine konkrete Umsetzung des Kompetenzmodells für eine Elektrotechnik HTL (Ausbildungsschwerpunkt Informationstechnik) für einen auf Verifikation von Software und Systemen ausgerichteten Unterricht. Die ausgearbeiteten Unterrichtsblöcke sind mit einer Einordnung in den aktuell vorliegenden Entwurf des Kompetenzmodells versehen, aber bewusst so konzipiert, dass diese auch außerhalb dieses Modells im Unterrichtsalltag eingesetzt werden können.

Schlagwörter: Informatikunterricht, Kompetenzmodell, Bildungsstandards, formale Software Verifikation und Test, Berufsbildung, Höhere Technische Lehranstalt

Abstract

Context: Educational standards, curriculum frameworks, competence models, and qualification frameworks are the major buzzwords in recent educational debates. For most of these, a comparability of qualifications among the EU member states and a clear classification of competences is the goal set. The project entitled "educational standards in vocational education" is initiated by the Austrian Federal Ministry for Education, Arts and Culture and aims at deriving transparent representation forms of learning results as well as system comparability. Individual work groups are preparing actual competence models, thereby, defining skills in various knowledge branches that a future graduate (e.g. from an Higher Technical and Vocational College) should own.

Scientifical objectives: How can the present computer science education for the Higher Technical and Vocational College (HTL) be enriched with the topics (formal) verification of software and systems under the considerations of educational standards?

Scientifical methods: Qualitative interviews with a teacher of the HTL Hollabrunn evaluate the current situation of computer science education. Another set of interviews with representatives of related industry branches shows the actual needs of the industry regarding future employees. The presented teaching units are developed within an expert feedback loop together with Dir. Dipl.-Ing. Mag. Dr.phil. Martin Weissenböck (principal of HTL Rennweg 3) and Dipl.-Ing. Dr. Walter Führer (teacher at HTL Hollabrunn). A comprehensive overview of related literature is the theoretical basis of this work.

Key results: It is known that, in industrial practice, verification of software and systems is a key work package in modern software projects. As for the current situation at schools, (formal) software and systems verification is hardly ever a topic in computer science education. This assumptions are backed up with interviews of both representatives of school and industry. It turns out, that there is a significant mismatch of what is taught and what is actually desired by the industry. The present thesis shows an actual implementation of the competence model for an electrical engineering HTL (with focus on information technology) in order to make (formal) verification of software and systems a topic in computer science education. Thus, teaching units are prepared and classified over the existing competence model draft. From the conceptional point of view, the teaching units are not meant to be solely used in conjunction with the competence model, moreover, they are ready be integrated into the daily teaching practice.

Keywords: computer science education, competence modells, educatioinal standards, formal software verification and testing, vocational education, HTL

Danksagung

Vielen Dank an Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Futschek für die Betreuung der Arbeit und seine wertvollen, konstruktiven Kommentare.

Weiters bedanke ich mich herzlich bei Dir. Dipl.-Ing. Mag. Dr.phil. Martin Weissenböck für die Mitwirkung an der Arbeit. Sein kompetentes Feedback und jahrelange Erfahrung im österreichischen Bildungswesen haben mir während der Durchführung sehr geholfen, meinen pädagogischen Horizont erweitert und einen interessanten Einblick ermöglicht.

Ein besonderes Danke geht an Dipl.-Ing. Dr. Walter Führer der mich vom ersten Konzept hinweg bis zur fertigen Arbeit mit Rat und Tat unterstützt hat.

Danke auch an Dipl.-Ing. Dr. Stephan Grünfelder und 谢谢 (Xiè Xiè) an Dipl.-Ing. Christian Galbavy für Ihre Mitarbeit als Interviewpartner.

Mein größter Dank geht an meine Freunde und Familie – die mich immer unterstützt und mir geholfen haben meinen Weg zu finden.

Thomas Reinbacher
Wien, April 2010

Inhaltsverzeichnis

1 Motivation

> Remember that not getting what you want is sometimes a wonderful stroke of luck.
>
> *(The 14^{th} Dalai Lama)*

Spätsommer 2008, ein lauer Sonntagabend im westlichen Weinviertel geht zu Ende. Ich bringe die von mir betreute Austauschstudentin in letzter Minute zum Bahnhof um sie wieder in die schöne Donauhauptstadt zurückzubringen. Während der Autofahrt spreche ich beruhigend von dem neuen, touch-screen basierten Automaten, der es ermöglicht in Sekundenschnelle ein Zugticket zu lösen. Meine koreanische Austauschstudentin zeigt sich davon nur wenig bis gar nicht beeindruckt – was auch wenig verwunderlich ist, vergleicht man Seoul's super modernes öffentliches Verkehrsnetz mit der Infrastruktur der niederösterreichischen Vorstadtidylle – und bereitet die von mir zuvor angesprochenen 6.8 € für den Einzelfahrschein Hollabrunn – Wien Handelskai vor ...

Der koreanische Gast und ich schreiten schnellen Schrittes zum Fahrkartenautomaten zu, sie navigiert diesen flink zur Sprachauswahl und wählt Englisch, ist kurz verwundert über die langsame Reaktionszeit der Eingabe, selektiert danach Abfahrtsbahnhof und Ankunftsbahnhof und drückt schließlich „OK buy ticket" um das begehrte Ticket zu erhalten. Nach Einwurf der vorbereiteten Euro Münzprägungen wird indirekt proportional zur noch verbleibenden Wartezeit auf den Zug ihr Gesicht immer länger und sie schaut mich schließlich voller Verwunderung an. Ein Blick auf den Automaten erklärt die schlagartige Verwirrung, dieser weist einen noch fehlenden Betrag von stolzen 3720.8 € aus. Für die 49 km lange Bahnstrecke macht das circa 75 € pro Kilometer oder ein über-durchschnittliches ganzes Jahreseinkommen von Angestellten im asiatischen Raum aus. Nach dem Ertönen des obligatorischen „Zug fährt ab" durch die Beschallungsanlage ist nun auch Zeit für eine kurze rationale Kosten-Nutzen Analyse und wir kommen zum Schluss, dass die 3720.8 € für einen – wenn auch klimatisierten – Regionalzug doch etwas überteuert sind und weit über unseren finanziellen Möglichkeiten liegen. Ein klassischer Softwarefehler, ein Bug also. Wieder einmal. Was wäre denn passiert, wenn wir die bargeldlose Option „zahlen mit Kreditkarte" gewählt hätten?

Im schlimmsten Fall hätte sich das mühsam angesparte Geld für die noch bevorstehende Europareise in ein Regionalzugticket manifestiert.

Dieses aus Eigenerfahrung wiedergegebene Beispiel ist nur eines der vielen Softwarefehler, die wir im alltäglichen Leben antreffen. Der Fahrkartenkauf mag harmlos erscheinen, die kürzliche Geschichte hat aber auch ganz andere Zwischenfälle mit teilweise verheerenden Ausgang aufgezeigt. Als teuerster Software Fehler in der Geschichte der Raumfahrt markierte die **Explosion der Ariane 5** [1] Trägerrakete im Juni 1996, ausgelöst durch eine fehlerhafte Datenumwandlung, einen schwarzen Tag für die Softwareentwicklung. Der **NASA Mars Climate Orbiter** [2] ging 1999 aufgrund eines Einheitenfehlers in der Navigationssoftware verloren. Es stellte sich heraus, dass das angloamerikanische Maßsystem anstatt dem metrischen Einheitensystem verwendet wurde. Auch abseits der Luft- und Raumfahrt gibt es immer wieder gravierende Softwarefehler. Am 14ten August 2003, ließ der **US-Northeast Blackout** [3] rund 50 Millionen Menschen in den USA und Kanada im Dunkeln. Eine Wettlaufsituation (race condition) im Energiemanagement-System war eine der Ursachen die zum Ausfall geführt hat. Ein stellvertretendes Beispiel aus der Automobilbranche ist der als **Toyota Prius Stalling Failure** [4] bekannt gewordene Softwarefehler, der die Automobilelektronik zum Absturz bringt, wenn das Fahrzeug mit Autobahngeschwindigkeit unterwegs ist. Stellvertretend für die Applikationsentwicklungsbranche steht der **Microsoft Excel 2007 Multiplication Bug** [5], in dem die Tabellenkalkulationssoftware alle Multiplikationen die sich zu 65.535 ergeben fälschlicherweise zu 100.000 errechnet. Im Oktober 2008, kam es zu einem **Ausfall der Mobilfunkservices von A1** [6], der eine halbe Million Kunden in Wien und Niederösterreich von der mobilen Außenwelt abschnitt.

Dieses Mini-Logbuch einiger bekannter Softwarefehler soll darauf hinweisen, dass wir tag-täglich mit immer mehr Kommunikations- und Informationssystemen arbeiten und wir zunehmend von deren korrekten Funktion abhängig sind. Umso fataler sind Softwarefehler, die uns im schlimmsten Fall nur unsere wertvolle Zeit kosten. Die zunehmende Funktionalität zeitgenössischer Softwareapplikation steigert auch die Softwarekomplexität und die Anforderungen an den Entwicklungsprozess. Softwarequalitätskontrolle für moderne Produkte wird zunehmend schwieriger [7]. In der Industrie sind daher Kompetenzen im Themenbereich Softwaretesten [8] und Softwareverifikation [9] immer wichtiger, reines Entwicklungs Know-How reicht längst nicht mehr aus um als Ingenieur langfristig erfolgreich zu sein.

Seit dem Beginn des Informatikunterrichts (vor ca. 35 Jahren [10] in der BHS) im österreichischen Bildungssystem gab es schier endlose Diskussionen wie man den Lernenden das „Programmieren im Kleinen“ am besten beibringen soll. Eine ganze Reihe von Publikationen [11, 12, 13, 14] argumentieren für oder gegen den Einsatz einer spezifischen Programmiersprache. Mal objektorientiert, mal prozedural, mal funktional, mal imperativ. Rund 35 Jahre Schulinformatik in Österreich haben aber auch die Fachdidaktik der Informatik bedeutend weiterentwickelt und moderner Informatikunterricht ist heute weit mehr als nur programmieren, Struktogramme entwerfen oder alle Schlüsselwörter einer bestimmten Programmiersprache zu erklären.

Die berufsbildenden höheren Schulen im Allgemeinen und die höheren technischen Lehranstalten (HTL) im Speziellen genießen national als auch international einen durchwegs guten Ruf. Hier werden die Techniker und Ingenieure von morgen ausgebildet. Ungeachtet des Ausbildungsschwerpunkts, ob Maschinenbau, Elektrotechnik, Elektronik, Wirtschaftsingenieurwesen oder Technische Informatik, bekommt jeder der Absolventen mindestens die Grundzüge der Softwareerstellung mit auf den Weg gegeben. Für die Informatik nahen Ausbildungsschwerpunkte, wie etwa Elektronik, Elektrotechnik, Technische Informatik ist im HTL Curriculum meist auch eine vertiefende Spezialisierung für die Programmerstellung für Eingebettete Systeme (Embedded Systems)[1] vorgesehen.

Die Lernenden arbeiten hier hardwarenahe mit Mikrocontroller. In der Industrie werden diese unter anderem auch für sicherheitskritische Applikationen eingesetzt. Für die Entwicklung von sicherheitskritischer Software beschäftigt die Industrie hauptsächlich die Frage *"Wie kann garantiert werden, dass die erstellte Software unter allen möglichen Umständen wie erwartet funktioniert und somit die Spezifikation erfüllt?"*

Somit stellt die alleinige Vermittlung von Programmierkompetenz einen klaren Widerspruch mit den Anforderungen der Industrie dar. Erfahrungsberichte in der Praxis haben gezeigt, dass für ein durchschnittliches Softwareprodukt im sicherheitskritischen Bereich (z.B.: Automobilbranche [16], Öffentlicher Verkehr, Raumfahrt [17], ...) ein Großteil der Zeit für die „Richtigstellung" des Programms verwendet wird, lediglich ein Bruchteil der gesamten Projektzeit fällt dabei auf die eigentliche Erstellung des Programmtextes.

In dem Curriculum der HTL für Informationstechnik sind für das Pflichtfach „Industrielle Informationstechnik" (INIT) folgende Lehrinhalte niedergeschrieben:

- Programmiersprachen der Informationstechnik
- Prozessrechnergrundlagen
- Mikrocontroller

Dies lässt durchaus eine Orientierung an der modernen Elektronikindustrie erkennen. Jedoch liegt die Vermutung nahe, dass in der Unterrichtspraxis der Fokus dieser Lehrveranstaltungen weitgehend beim Erlernen vom klassischen „Programmieren" liegt.

Dass aber in einer HTL durchaus Projekte mit hohen Ansprüchen an einen ganzheitlichen Softwareentwurf durchgeführt werden, lässt sich auf einen Blick der abgewickelten Diplomarbeiten an der HTL Hollabrunn erkennen. Hier finden sich beispielsweise Projekte wie „*μC basierte Messdatenerfassung mit Internetzugang, Minifileserver, Fernbedienung*

[1] Embedded Systems [15], sind applikationsspezifische Mikrocomputersysteme mit applikationsspezifischer Software innerhalb einer zu beeinflussenden Umgebung. Sie realisieren die benötigte „Intelligenz" in Systemen im Automobilbereich zur Leistungsoptimierung und zur Erhöhung der Sicherheit und des Fahrkomfort bis hin zu Hochleistunganlagen im Bereich der Telekommunikation und Automatisierungstechnik. Sowohl die Geräte der modernen Messtechnik, als der Medizintechnik wären ohne den Einsatz von leistungsfähigen Embedded Systems nicht denkbar.

einer Embedded Realtime-Steuerung, Prüfstand zur Kennlinienaufnahme von elektrischen Maschinen, Rechnergestützte Streckenidentifikation, ..." [18].

Besonders für die industrielle Informationstechnik ist daher neben der korrekten Funktion von Software auch das Einhalten einer vorher vereinbarten Spezifikation von höchster Bedeutung. Für die Industrie sind daher vor allem jene Absolventen interessant die neben der Fähigkeit des Programmierens auch über Fachwissen zum Testen und Verifizieren von selbst erstellter Software verfügen.

Eine technische Ausbildung wie es eine HTL darstellt versucht in vielen Bereichen stets dem Innovations Puls der Industrie, dem State-Of-The-Art, zu folgen und die Absolventen bestmöglich auf ihr bevorstehendes Berufsleben vorzubereiten. *Warum sollte dies nicht auch in einem so signifikanten Ausbildungsbereich wie der Softwareentwicklung angewandt werden?*

Hier knüpft diese Diplomarbeit an, um didaktische Konzepte und Unterrichtseinheiten für das Fach „Industrielle Informationstechnik" einer Elektrotechnik HTL zu erarbeiten, die für die Lehre der Fachgebiete Software Testen und (formale) Verifikation verwendet werden können. In dieser Arbeit wird erstmals eine konkrete Umsetzung des vom bm:ukk ausgearbeiteten Kompetenzmodell hinsichtlich der Anwendbarkeit für eine HTL präsentiert.

1.1 Warum (formale) Software-Verifikation unterrichten?

Formale Verifikationsmethoden können als die Angewandte Mathematik für die Modellierung und Analyse von Informations- und Kommunikationstechnologie (IKT) Systemen betrachtet werden. Sie sind eine der „highly recommended" Verifikationstechniken für Softwareentwicklung von sicherheitskritischen Anwendungen, lt. dem „Best Practices" Standard [19] der IEC (International Electrotechnical Commission) und den Standards der ESA (European Space Agency) [20].

Der finale Report einer Untersuchung über formale Verifikation der FAA (Federal Aviation Authority) und der NASA (National Aeronautics and Space Administration) kommt zu folgender Empfehlung (aus [21]):

> Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers.

Warum sollten daher (formale) Verifikation und Software und Systemen nicht Teil der Informatikausbildung zukünftiger Ingenieure sein?

1.2 Hypothesen und wissenschaftliche Herangehensweise

In der geplanten Arbeit sollen folgende Thesen und Überlegungen wissenschaftlich untersucht, bewertet, verifiziert oder falsifiziert werden. Für den Schultyp HTL gilt:

(H.a) Der Informatikunterricht bedarf einer Verbesserung *und ist aktuell nicht im perfekten Einklang mit den Anforderungen der Industrie an die Absolventen, es bedarf einer Änderung der Unterrichtsschwerpunkte.*

(H.b) Eine Anpassung des Informatikunterrichts *mit thematischer Ausrichtung auf einen industrienahen Softwareentwicklungsprozess (Entwickeln, Testen, Verifizieren)* **ist machbar** *und stellt einen signifikanten Qualitätszuwachs der Ausbildung dar.*

(H.c) Das Kompetenzmodell des bm:ukk ist eine Bereicherung *für den standardisierten Unterricht. Es ist möglich den bestehenden Kompetenzmodell Entwurf mit Beispielen rund um die formale Softwareverifikation zu erweitern.*

1.3 Struktur und Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt strukturiert: Nach einer Einführung in die Thematik und der didaktischen Lehrmöglichkeiten in Kapitel 2, folgt in Kapitel 3 eine Zustandserhebung der Anforderungen der Industrie an zukünftige HTL Absolventen. Diese wird auf Basis zweier Interviews durchgeführt. Im Kapitel 4 wird stellvertretend ein Interview mit einem Lehrenden in einer Informationstechnik HTL durchgeführt um die aktuellen Lehrinhalte und Schwerpunkte des Unterrichtsfaches INIT stichprobenartig zu erfassen und zu evaluieren. Darauf folgend werden in Kapitel 5 Bildungsstandards der EU und das Kompetenzmodell des bm:ukk vorgestellt und erläutert. Kapitel 6 führt konkrete Unterrichtsbeispiele für den Themenkreis (formaler) Software Test und Verifikation und deren Einordnung in das bestehende Kompetenzmodell des bm:ukk für die HTL Informationstechnikausbildung an. Zum Abschluss werden die gewonnenen Erkenntnisse und Erfahrungen in Kapitel 7 zusammengefasst und ein persönlicher Ausblick auf die zukünftige Informatikausbildung gegeben.

1.4 Verwandte Arbeiten

Eine verwandte Arbeit ist die Diplomarbeit von Prein [22], die Entwürfe zum Pflichtgegenstand Angewandte Programmierung für eine HTL bearbeitet, aber im Gegensatz zur vorliegenden Arbeit keine Ausrichtung auf industrielle Anforderungen wie Test und

Verifikation [21, 23, 24] behandelt und sich auf eine Einführung in das Programmieren beschränkt.

Eine ganze Reihe von Publikationen [12, 14, 11, 13, 25] argumentieren für oder gegen den Einsatz einer spezifischen Programmiersprache im Schulunterricht, jedoch finden sich kaum Publikationen die sich mit der Vermittlung der Fachthematiken Test und Verifikation von Software beschäftigen.

Der aktuelle Status des bm:ukk Projekts „Bildungsstandards in der Berufsbildung“ ist in [26] dargestellt. Ein erstes Konzept für HTLs findet sich in [27, 28].

Anmerkung

Soweit in der nachfolgenden Arbeit aus sprachlichen Gründen nur die männliche Form gewählt wurde, gilt jeweils auch die weibliche Form.

2 Grundlagen

> Education is what remains after one has forgotten everything he learned in school.
>
> *(Albert Einstein)*

2.1 Entstehung der Schulinformatik in Österreich

Im Schuljahr 1985 wurde erstmals im österreichischen Bildungssystem das Pflichtfach Informatik in der 5ten Klasse der allgemein bildenden höheren Schule (AHS) lehrplanmäßig verankert. Neben der AHS wurde dies auch in den Lehrplan der Polytechnischen Lehrgängen mitaufgenommen. Das Fort- und Weiterbildungsprojekt „Computer Bildung Gesellschaft“ (CBG) hatte es unter anderem auch zum Ziel mit Hilfe von Industriepartnern ein zweiwöchiges Schulungsseminar in Wien zu veranstalten, an dem rund 500 Lehrende teilnahmen und ihr Wissen dann an Kollegen ihrer Schule weitergegeben haben. Der Fokus dieser Seminare lag auf (i) den technischen Belangen von Hard- und Software einerseits und (ii) die Nutzung von Anwendungen und Problemlösungsstrategien andererseits [29].

Diese ersten Informatiklehrenden konnten sich natürlich noch nicht auf die heute breit vorhandene Theorie und Erfahrungswerte der Didaktik der Informatik abstützen. Leider hatte dies auch oft zur Folge, dass der Informatikunterricht stark an die Fachwissenschaft gekoppelt war und einem „Programmierkurs“ glich. Man versuchte das „Programmieren im Kleinen“ zu vermitteln und stürzte sich von einem Schlüsselwort der verwendeten Programmiersprache zum Nächsten. Diese Meinung wird etwa auch von [30] vertreten.

Zehn Jahre später, 1995, war es die EU Devise „Schulen ans Netz“ die den österreichischen Schulen Zugang zum `www` einherbrachte. Ein typisches state-of-the-Art System bestand aus einer CPU der Intel 80486 oder Intel 80586 Architektur, einigen hundert Megabyte an Festplattenkapazität, Arbeitsspeicher im einstelligen Megabyte Bereich und einem Windows 3.11 for Workgroups Betriebssystem. In der heutigen Zeit werden diese Hardwarekonfigurationen meist von Embedded Systems die jedermann mit sich herumtragt, wie etwa MP3 Player oder Smartphones, bei weitem überboten. Nichts desto trotz – die österreichischen Schulen sind auf den Datenhighway aufgesprungen.

1997 wurde die Standardisierung des Bildungsniveaus mit dem Umgang von Computern

mit der Einführung des Europäischen Computer Führerschein (ECDL) [31] eingeläutet. Das Basismodul umfasst insgesamt sieben Module, angefangen von den Grundlagen der Informations- und Kommunikationstechnologie über Text und Tabellenkalkulation bis hin zu Web und Kommunikation. Neben dem Basismodul gibt es ein Fortgeschrittenes Zertifikat, eines auf Computer Aided Design (CAD) abgestimmtes, ein WebStarter und ein ImageMaker Zertifikat. Über den Schulsektor hinaus, bietet der Computerführerschein einen europaweit abgestimmten Standard und durch eine standardisierte Prüfung ein entsprechendes Zertifikat, das den erfolgreichen Einstieg in die Informationsgesellschaft bestätigt [32]. Das Konzept des ECDL ging auf, bis Februar 2010 wurden allein in Österreich insgesamt rund 250.000 Zertifikate ausgestellt.

Der Trend zur Zertifizierung setzt sich auch in den berufsbildenden höheren Schulen durch, facheinschlägige HTLs fördern und unterstützen die Schüler am Weg zum CISCO - Certified Network Associate (CCNA). Vielerorts wurden eigene Networking Academies gegründet um Schüler handwerkliches Wissen als auch die Implementierung, Installation und Wartung komplexer Hard- und Software zu vermitteln. Eine dieser Niederlassungen findet sich zum Beispiel an der HTL Hollabrunn und der HTL Donaustadt. Detaillierte Informationen zur Umsetzung an berufsbildenden mittleren und höheren Schulen finden sich in [33].

Im Studienjahr 2000/2001 – ganze 15 Jahre nach der Einführung des Informatikunterrichts an AHS – wurde das Lehramtsstudium Informatik und Informatikmanagement geschaffen [34]. Bis zur Einführung dieses Studiums unterrichteten im Fach Informatik Lehrer welche andere Unterrichtsfächer studiert haben und sich durch Zusatzqualifikationen ihr Fachwissen angeeignet haben. Besonders hinsichtlich der Entwicklung einer eigenständigen Fachdidaktik der Informatik und ein neues Selbstverständnis des Faches Informatik ist dieses neue geschaffene Lehramtsstudium ein positiver Impuls.

2.2 Berufsbildende Höhere und Mittlere Schulen in Österreich

Die österreichischen berufsbildenden mittleren und höheren Schulen sind von einem manigfaltigen Angebot an Ausbildungsschwerpunkten geprägt. Eine höhere Schule schließt mit der Reife- und Diplomprüfung (Matura), eine mittlere Schule mit einer Abschlussprüfung (Diplom) ab.

Lt. dem Bundesministerium für Unterricht, Kunst und Kultur (bm:ukk) [35] werden in einer berufsbildenden mittleren Schule, vulgo Fachschule, berufliche Qualifikationen und Allgemeinbildung vermittelt. Sie dauert drei oder vier Jahre und ermöglicht eine Weiterqualifizierung z.B. durch eine Berufsreifeprüfung.

Ziel einer berufsbildenden höheren Schule ist es eine höhere berufliche Ausbildung einerseits und eine fundierte Allgemeinbildung andererseits zu vermitteln. Sie dauert im Regelfall fünf Jahre und schließt mit einer Doppelqualifikation ab: Die abschließende Reifeprüfung

berechtigt zum allgemeinen Hochschulzugang, die Diplomprüfung deckt die beruflichen Qualifikationen ab. Absolventen erhalten nach nachweislicher facheinschlägiger Berufspraxis die Standesbezeichnung „Ingenieur“ (Ing.) verliehen. Diese wird vom Bundesministerium für Wirtschaft, Familie und Jugend (bm:wfj) verliehen und basiert auf §3 des Ingenieurgesetz von 2006 (IngG 2006).

Geläufige Vertreter der berufsbildenden höheren Schule sind die höheren technischen Lehranstalten (HTL), die Handelsakademien, die Land- und forstwirtschaftlichen Lehranstalten und die höheren Lehranstalten für wirtschaftliche Berufe. Aufgrund der großen Typenvielfalt der berufsbildenden mittleren und höheren Schulen zeichnet sich auch der Informatikunterricht durch ein großes Angebotsspektrum aus [36]. In der Kategorie Sekundärstufe II, bilden die berufsbildenden höheren Schulen mit deutlichem Abstand - noch vor den Berufsschulen - die meisten Schüler aus, wie Tabelle 2.1 belegt (aus [37]).

Tabelle 2.1: Entwicklung der Schülerzahlen zwischen 2000/01 und 2004/05.

Schultyp	**2000/01**	**2002/03**	**2004/05**	**Durchschnitt**	
Polytechnische Schulen	19.594	20.626	21.769	20.663	4,97 %
AHS Oberstufe	77.788	77.121	81.135	78.681	18,92 %
Berufschulen	132.613	127.806	124.983	128.467	30,88 %
BMS	48.849	51.050	53.735	51.211	12,31 %
BHS	132.747	137.138	140.949	136.945	**32,92 %**
Gesamt	411.591	413.741	422.571	415.968	–

Die technischen Ausbildungsaspekte einer HTL fokussieren dabei schwerpunktmäßig auf z.B.: Automatisierungstechnik, Bautechnik, Elektronik, Elektrotechnik, Informatik, Maschinenbau und Wirtschaftsingenieurwesen. Sie vermittelt eine hochwertige, fachtheoretische und fachpraktische Bildung als Grundlage für einen reibungsfreien Einstieg in das Berufsleben und für eine erfolgreiche Tätigkeit in verschiedensten Einsatzbereichen. Neben der fachlichen Bildung findet auch ein Ausbau der allgemeinen und sozialen Qualifikationen statt. Dem hohen Ausbildungsniveau der HTL wird mit der EU Richtlinie 2005/36/EG [38] nun auch EU weit Rechnung getragen. Somit ist der erworbene Abschluss in das in der Richtlinie definierte „Diplomniveau“ einzuordnen. Die Richtlinie legt den Berufszugang fest, trifft aber keine Entscheidung über die Gleichstellung mit akademischen Graden.

Im europäischen Umfeld konkurrieren Absolventen einer HTL durchaus mit Absolventen einer Fachhochschule, oder Absolventen mit dem akademischen Grad Bachelor (Bachelor of Engineering, Bachelor of Science). Beispielsweise in der Schweiz wurden die Ingenieurschulen, vergleichbar mit österreichischen HTLs, um die Jahrtausendwende in Fachhochschulen umgewandelt. Österreich hat sich dazu entschlossen an dem Konzept der HTLs festzuhalten und die Fachhochschulstudiengänge einzuführen.

Eine im Regelfall bis zu zwei Semester anrechenbare Vorbildung erlaubt es, Absolventen in eine Fachhochschule quer einzusteigen, und somit das Bakkalaureat innerhalb der

überschaubaren Zeitspanne von vier Semestern zu absolvieren. Im Einzelfall ist eine Anrechnung einzelner Lehrveranstaltungen auch an österreichischen Universitäten möglich, generell ist diese aber nur in viel geringerem Ausmaß möglich.

Auch die Industrie weiß die Qualitäten der HTL Ausbildung zu schätzen. Das Berufsbild wird zum Beispiel lt. Eigendefinition der HTL Hollabrunn [39] für den Ausbildungszweig Elektronik wie folgt angegeben:

> Unsere Absolventen sind in Betrieben jeder Größe auf allen Organisationsebenen gefragt. Sie arbeiten oft in der Automatisierungstechnik, Informatik, Telekommunikation und Kfz- Elektronik, und werden dabei überwiegend in der Rechner- und Netzwerkbetreuung, der Hard- und Softwareentwicklung und im technisch-kaufmännischen Bereich eingesetzt. **Branchen**: Automatisierungstechnik, Informatik, Telekommunikation, Kfz-Elektronik, ... **Tätigkeiten**: Rechner- und Netzwerkbetreuung, Hard- und Softwareentwicklung, technisch-kaufmännischer Bereich, ...

Vergleicht man dies mit dem Berufsbild wie es zum Beispiel die Fachhochschule Technikum Wien für den Bachelor Studiengang Elektronik definiert [40], stellt man große Ähnlichkeiten fest:

> Zahlreiche österreichische Elektronikunternehmen zählen weltweit zur absoluten Technologiespitze. Diese Firmen entwickeln die Anwendungen der Zukunft – im Automobil-, Flugzeug- oder Telekommunikationssektor ebenso wie im Chipdesign oder in der Medizintechnik. So vielfältig wie die elektronischen Anwendungen sind auch die beruflichen Möglichkeiten von ElektronikerInnen. **Branchen**: Elektro- und Elektronikindustrie, Luftfahrt- und Automobilindustrie, Telekommunikation, Industrieautomation, ... **Tätigkeiten**: Hard- und SoftwaredesignerIn, SystementwicklerIn, ProjektleiterIn / ProjektmanagerIn, ProduktmanagerIn ...

Besonders die Branchen und die Tätigkeitsbereiche unterscheiden sich nur marginal, was auch der Realität im Arbeitsleben entspricht. HTL Absolventen konkurrieren aufgrund ihrer hohen fachlichen Kompetenz mit Bachelor Absolventen von fachähnlichen Fachhochschulen und Universitäten.

2.3 Grundlagen der Didaktik

Die Didaktik beschäftigt sich im speziellen mit der Theorie des Unterrichts, im Allgemeinen mit der Theorie und Praxis des Lehrens und Lernens. Sie ist eines der Kernthemen der Pädagogik und arbeitet mit mehreren (theoretischen) Bildungstheorien. Sie ist die Wissenschaft des Lehrens.

2.3.1 Theoretische Ansätze

Während sowohl Lehrende als auch Lernende aus den verschiedensten Theorien der Allgemeinen Didaktik nur wenig unmittelbaren Nutzen und Zusammenhang mit ihrem Unterricht sehen, ist es doch interessant welchen Einfluss diese Theorien auf das tägliche Lehrgeschehen nehmen – wenn oft auch unbewusst.

Da diese Arbeit auch eine Unterrichtsplanung zum Gegenstand hat, sollen hier – gemäß der Kategorisierung nach Hubwieser [41] – die vier bekanntesten Theorien im Überblick behandelt werden.

Bildungstheoretischer Ansatz

Der bildungstheoretische Ansatz wurde von Klafki [42] geprägt und basiert auf folgender Kernaussage [41]:

> Ziel muss die Bildung des Menschen im Ganzen sein. Die Aufnahme und Aneignung von Inhalten ist stets verbunden mit der Formung, Entwicklung und Reifung von körperlichen, seelischen und geistigen Kräften.

Im Vordergrund steht das WAS (wird unterrichtet) und nicht das WIE (wird etwas vermittelt). Womit müssen sich Jugendliche auseinandersetzen damit sie mündig und gebildet werden? Welche gegenwärtige und zukünftige Bedeutung hat der vermittelte Stoff? Die von Klafki geforderte didaktische Analyse bringt fünf Überlegungen zur Auswahl der Unterrichtsgegenstände hervor:

1. Welche exemplarische Bedeutung hat der Unterrichtsgegenstand?
2. Wie bedeutend ist er für die Gegenwart?
3. Welche Bedeutung für die Zukunft lässt sich vermuten?
4. Wie ist die Struktur des Inhalts?
5. Wie steht es mit der unterschiedlichen Zugänglichkeit?

Im Hinblick auf einen modernen Unterricht ist dieser Ansatz immer noch aktuell und unterstützend bei der Auswahl von Unterrichtsgegenständen und dem Erstellen von Curricula.

Lerntheoretischer Ansatz

Der lerntheoretische Ansatz entstand unter Federführung von Heimann [43, 44]. Hierbei wird die Didaktik als Theorie des Lehrens und Lernens verstanden. Im Gegensatz zum bildungstheoretischen Ansatz steht hier neben dem WAS (wird unterrichtet) auch das

WIE (wird etwas vermittelt). Der Lehrinhalt ist immer stark formal strukturiert, jedoch situationsbedingt änderbar und somit inhaltlich variabel.

Zur Unterrichtsplanung werden die in Tabelle 2.2 dargestellten Entscheidungsfelder gegeneinander abgewogen und in Einklang gebracht.

Tabelle 2.2: Entscheidungsfelder (angelehnt an [45]).

Intentionen	Welche Zielsetzung hat der Unterricht?
Lerninhalte	Was wird gelehrt?
Methoden	Wie wird der Stoff vermittelt?
Medien	Womit wird der Lernstoff transportiert?
Folgen	Welche soziokulturellen und anthropologisch-psychologischen Auswirkungen wird der Lernvorgang haben?

Diese Überlegungen stehen wiederum in Abhängigkeit mit Bedingungsfeldern, die die gesamtheitlichen Folgen der Überlegungen aufzeigen. Diese Bedingungsfelder finden sich in Tabelle 2.3.

Tabelle 2.3: Bedingungsfelder (angelehnt an [45]).

sozio-ökonomische	finanzielle und wirtschaftliche Rahmenbedingungen	Klassenstärken, Ausstattung mit Lehrmitteln, etc.
sozio-ökologische	Einbettung des Unterrichts in ein räumliches Umgebungs-konzept	Schulstandort, Verkehrsanbindung, Lärmbelästigung, etc.
sozio-kulturelle	aus der geschichtlich-geistigen Situation erwachsende Strömungen, Einstellungen, etc.	No-Gos, Kommunikationsweisen, Sprachformen, Umgangsformen Symbole, etc.
ideologisch-normbildende	aus Interessenlagen einzelner gesellschaftlicher Gruppen und Mächte stammender Einflüsse	politische Richtziele, Umweltschutzbewegungen, etc.

Die Folgen des Unterrichts, die durch die Entscheidungsfelder evaluiert werden, sind in der nächsten Unterrichtsplanung Voraussetzungen, somit entsteht ein in sich geschlossener Kreislauf. Zur Abgrenzung zum geisteswissenschaftlich beeinflussten bildungstheoretischen Ansatzes stellt diese Theorie ein praktikables Entscheidungsmodell zur wertfreien Analyse des Unterrichts und dessen Planung dar. Das Modell wurde in weiterer Folge von Schulz in den 1980er [46] zu einem Handlungsmodell für einen emanzipatorisch-relevanten, professionell-pädagogischen Unterricht weiterentwickelt.

Informationstheoretisch-kybernetischer Ansatz

Shannon's Informationstheorie [47] von 1948 wurde von Norbert Wiener unter dem Namen Kybernetik auch für Anwendungen fernab der Technik vorgeschlagen. Als grundlegendes Modell fungiert immer ein Regelkreis, der auch auf die Didaktik angewendet werden

kann [48, 49]. Die Didaktik wird dabei auf eine reine Methodik reduziert, die Auswahl des Lehrstoffs ist nicht die erste Priorität. Es wird unterschieden in einen pädagogischen Raum (Lehrziel, Medium, Psychostruktur) und die eigentliche Struktur des Lernprozesses, der als klassischer Regelkreis modelliert werden kann (vgl. Abbildung 2.1).

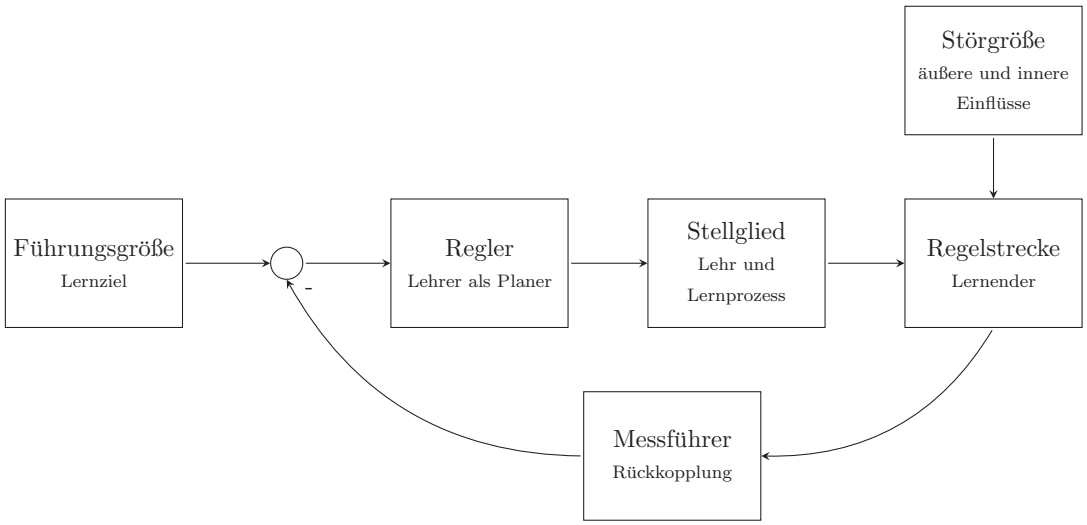


Abbildung 2.1: Lernprozess als Regelkreis (frei nach [41]).

Hubwieser argumentiert in [41], es sei geradezu naiv, einen derart komplexen Lernprozess mit so einfachen Modellen behandeln zu wollen. Dennoch hatte dieser Ansatz bemerkenswerte Auswirkungen auf Modelle zur Lehrveranstaltungsevaluierung und die Entwicklung von computerunterstützten Lehr- und Lernsystemen.

Kommunikative Didaktik

Der Grundstein für die kommunikative Didaktik wurde von Schäfer und Schaller [50] gelegt. Ihre Theorie beschreibt den Unterricht als kommunikativen und edukativen Prozess, wobei zwischen zwei Dimensionen des Kommunikationsgeschehens unterschieden wird [41]:

- **Inhaltsdimension**, die durch das Curriculum festgelegt ist.
- **Beziehungsdimension**, die das Verhältnis zwischen Lehrenden und Lernenden beschreibt.

Ein wesentlicher Augenmerk liegt auf der Weiterentwicklung der Lernenden gegenüber den Lehrenden als auch gegenüber ihrer Umgebung. Neben der direkten Kommunikation im Unterrichtsprozess wird hier auch die Bedeutung der Metakommunikation behandelt. Für den modernen Unterricht liefert diese Theorie eine Klarstellung der Bedeutung der sozialen Wechselwirkung zwischen Lehrer und Schüler.

2.3.2 Die Didaktik der Informatik und ihr Umfeld

Schubert und Schwill geben in [51] eine Einbettung der Didaktik der Informatik in andere Fachbereiche. Diese ist in Abbildung 2.2 dargestellt. Die Didaktik der Informatik ist nach dieser Definition die Schnittmenge aus Pädagogik, Psychologie, Informatik und Schule.

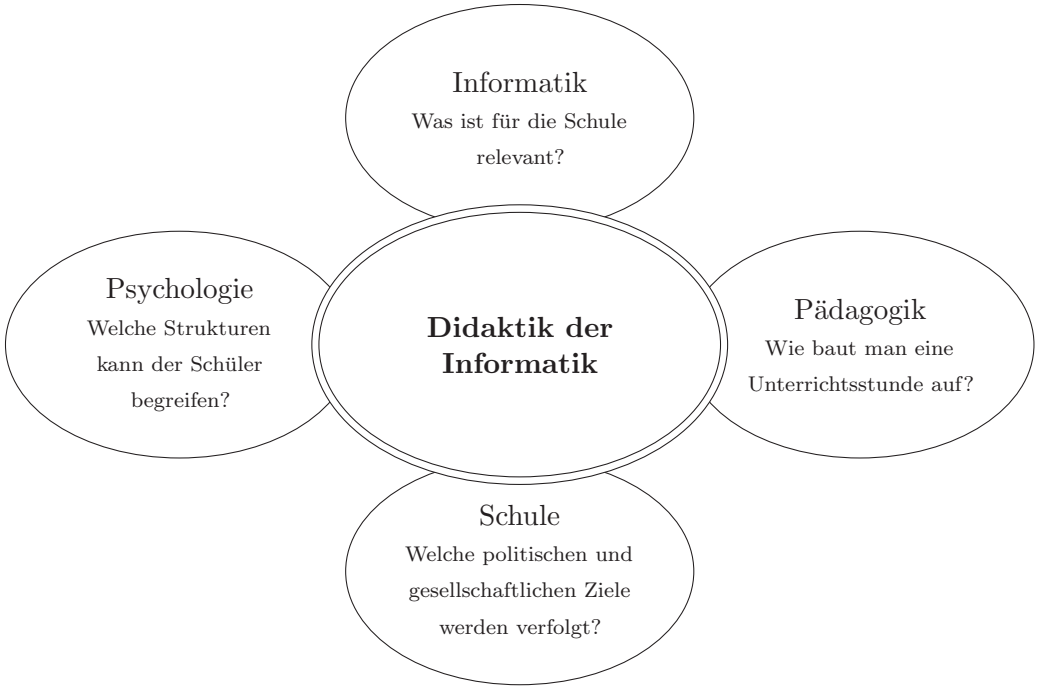


Abbildung 2.2: Einbettung der Didaktik der Informatik [51].

Didaktik der Informatik bedient sich daher der Pädagogik um der Frage nach einer idealen Unterrichtsstunde nachzugehen, der Psychologie um einfach begreifbare Strukturen und Stoffaufbereitungen zu finden, der Informatik um relevante Fachthemen zu selektieren. Als zusätzlicher Faktor ist das Schulumfeld zu sehen, das politische und gesellschaftliche Grenzen und Ziele diktiert.

2.3.3 Konzeptwissen und Produktwissen

Wissen kann auf viele verschiedene Arten kategorisiert werden. Eine einfache, auf den Informatikunterricht abgestimmte Einteilung, findet sich sowohl in [45] als auch in [52], und soll im nachfolgenden kurz erläutert werden.

Eine Klassifizierung in Konzeptwissen und Produktwissen eignet sich auch um Lehr- und Lernziele für den Informatikunterricht abzuleiten. Diese Klassifizierung ist in Tabelle 2.4 aufgezeigt.

Konzeptwissen beinhaltet die langfristigen, grundlegenden Zusammenhänge und Querverbindungen eines einzelnen Themengebietes. Produktwissen hingehen, umfasst die Kenntnisse

Tabelle 2.4: Produktwissen vs. Konzeptwissen (Angelehnt an [45]).

Produktwissen	**Konzeptwissen**
produktbezogen	produktunabhängig
kurzlebig	langlebig
auswendig lernen	verstehen
Inhalte wiedergeben können	kategorisieren
Wissenstransfer schwer möglich	Wissenstransfer möglich
konkret, auf Fakten basierend	abstrakt, Know-How basierend

und Fähigkeiten die für den Umgang mit einem konkreten Produkt notwendig sind. Dieses Produkt kann nun eine Softwareapplikation oder eine maßgeschneiderte Hardwareplattform sein.

Ein Beispiel dazu. Die folgenden Aspekte des Themenbereichs Debuggers gehören zum Konzeptwissen:

Grundlegendes: Ein Debugger ist ein Werkzeug zum Auffinden, Diagnostizieren und in weiterer Folge zur Behebung von Fehlern in Computersystemen. Der Begriff *debugging* wurde in Anlehnung an ein Insekt gewählt das in einem der ersten Relais Computer (Harvard Universität 1947, Mark II) zu einem Ausfall geführt hat [53].

Voraussetzung: Debugging funktioniert nur dann wenn der zu analysierende Quellcode auch zugänglich ist. Debugging einer compilierten und gelinkten *.dll ist ohne Spezialwissen nur schwer möglich. Um effizient zu debuggen werden vom Anwender auch Grundwissen über die verwendete Computerarchitektur verlangt.

Speichermodifizierung: Eine Besonderheit des Debuggers ist das Modifizieren von Speicher des Computers, beispielsweise dem Hauptspeicher, Ein/Ausgabe Register, Prozessorregister, Statusregister, ... Damit ist es auch möglich, Register während der Ausführung des Programms zu verändern, und somit beispielsweise Funktionsparameter, Rücksprungadressen, gesicherte Register am Stack, usw. anzupassen.

Einordnung: Ein Debugger ist vergleichbar mit typischen Messgeräten der Elektrotechnik und Elektronik, wie einem Oszilloskop, einem Signalgenerator, oder Logik Tester. Er ist daher als Hilfsmittel anzusehen um einen tiefen Einblick in das System zu gewähren.

Dieser kurze Exkurs in das Konzeptwissen über Debugger hilft auch beim Umgang mit einem konkreten Produkt. Für einen industriellen, professionellen Einsatz eines Debuggers ist aber ein parallel zum Konzeptwissen angelerntes Produktwissen unerlässlich. Verwendet man zum Beispiel den populären GNU Debugger so ist eine Einarbeitung in das Handbuch [54] und das Lesen der zugehörigen man-pages unerlässlich. Dieses anwendungsspezifische

Wissen stellt das Produktwissen dar. Alle Debugger beruhen auf den gleichen theoretischen Überlegungen, die Anwendung ist aber von Produkt zu Produkt sehr unterschiedlich. Beispielsweise hat der GNU Debugger kein graphisches User Interface und kann nur per Konsolenanweisungen bedient werden, andere hingehen sind in die Entwicklungsumgebung Eclipse integriert.

2.4 Grundlagen der Informatik

> Computation is the principle; the computer is the tool.
>
> *(Peter J. Denning)*

> Computer Science is no more about computers than astronomy is about telescopes.
>
> *(Edsger W. Dijkstra)*

Um Informatik unterrichten zu können, muss man sich zu allererst Gedanken machen, was denn eigentlich der Begriff Informatik bedeutet. Hierbei geht es nicht um die Wortherkunft sondern vielmehr um die Frage: Was beinhaltet die moderne Informatik?

Basierend auf den 1989 ACM/IEEE Artikel *Computing as a discipline* [55] von Comer et al. greift diese Kategorisierung Peter J. Denning 2003 nochmals im Artikel *Great principles of computing* [56] auf und unterteilt hierin die Informatik in folgende fünf Leitgebiete:

Computation What can be computed, Limits of computing. Related fields: Algorithm, control structures, data structures, automata, languages, Turing machines, universal computers, Turing complexity, Chaitin complexity, self-reference, predicate logic, approximations, heuristics, non-computability, translations, physical realizations.

Communication Sending messages from one point to another. Related fields: Data transmission, Shannon entropy, encoding to medium, channel capacity, noise suppression, file compression, cryptography, reconfigurable packet networks, end-to-end error checking.

Coordination Multiple entities cooperating toward a single result. Related fields: Human-to-human (action loops, workflows as supported by communicating computers), humancomputer (interface, input, output, response time); computer-computer (synchronizations, races, deadlock, serializability, atomic actions).

Automation Performing cognitive tasks by computer. Related fields: Simulation of cognitive tasks, philosophical distinctions about automation, expertise and expert systems, enhancement of intelligence, Turing tests, machine learning and recognition, bionics.

Recollection Storing and retrieving information. Related fields: Hierarchies of storage, locality of reference, caching, address space and mapping, naming, sharing, thrashing, searching, retrieval by name, retrieval by content.

Diese Einteilung wurde in einer weiteren Publikation [57] um die Punkte Evaluation und Design verfeinert:

Evaluation The principal tools of evaluation are modeling, simulation, experiment, and statistical analysis of data. Computing systems can be represented as sets of equations balancing transition flows among states. Network of servers is a common, efficient representation of computing systems. Network-of-server systems obey fundamental laws on their utilizations, throughputs, queueing, response times, and bottlenecks. Resource sharing, when feasible, is always more efficient than partitioning.

Design Design principles are conventions for planning and building correct, fast, fault tolerant, and fit software systems. Error confinement and recovery are much harder in the virtual worlds of software than in the real world of physical objects. The four base principles of software design are hierarchical aggregation, levels, virtual machines, and objects. Abstraction, information hiding, and decomposition are complementary aspects of modularity. Levels organize the functions of a system into hierarchies that allow downward invocations and upward replies. Virtual machines organize software as simulations of computing machines. Objects organize software into networks of shared entities that activate operations in each other by exchanging signals. In a distributed system, it is more efficient to implement a function in the communicating applications than in the network itself (end-to-end principle).

Diese Einteilung bestätigt eindeutig, dass Informatik $\neq$ Computer ist. Der Computer an sich, stellt für die Informatik nur ein Medium bzw. Werkzeug dar, um theoretische Konzepte auch in die Praxis umzusetzen. Informatik ist bedeutend mehr als „nur“ Computer. In einer weiteren Veröffentlichung [58] argumentiert Denning sogar für eine Gleichstellung der Informatik mit einer Naturwissenschaft.

2.5 Unterricht, Methoden und Lernen

In facheinschlägiger Literatur findet man manigfaltige Definitionen des Begriffs Unterricht. Nach [59] ist Unterricht eine systematische, geplante Lernerfahrung. Systematik und Planung deuten aber nicht darauf hin, dass Unterricht ein einfach beschreibbares Phänomen darstellt. Basierend auf der individuellen Feststellung, dass Unterricht untrennbar mit den Begriffen Wissensvermittlung, Erziehung, Charakterbildung, Interaktion zwischen Lehrenden und Lernenden aber auch kommunikatives Handeln verbunden ist, verwendet diese Arbeit folgende Definition aus [41]:

> Unterricht ist ein hoch komplexer Prozess mit zahlreichen, auch zyklischen Wechselwirkungen, bei dem Lehrende und Lernende unter gewissen gesellschaftlichen, bürokratischen und materiellen Vorgaben und Rahmenbedingungen im Hinblick auf eine bestimmte Zielsetzung interagieren. Langfristig hat dieser Prozess wiederum Auswirkungen auf die gesamte Gesellschaft und die von ihr formulierten Vorgaben.

2.5.1 Führungsstile der Lehrkraft

Der Verhaltenstypus der Lehrkraft hat eine entscheidende Wirkung auf den Unterrichtserfolg. Das Verhalten des Lehrers selbst resultiert aus folgenden Überlegungen [41]:

- aus Lehrtradition (vom eigenen Lehrer) übernommen,
- aus sozialer Lernerfahrung hergeleitet,
- durch philosophische Traditionen begründet,
- durch Bedürfnisse des Lehrers bewirkt (Selbstbestätigung),
- durch Bedingungen der Lehrsituation erzwungen (Lärm),
- durch Erforschung des Lernens fundiert;

Eine signifikante Bedeutung kommt dem Führungsstil der Lehrperson zugute. Dieser beeinflusst nachhaltig die Kreativität, Leistungsvermögen, Aufmerksamkeit und vor allem die Motivation der Lernenden. In [60] stellt Lewin drei grundlegende Kategorien zur Klassifizierung des Führungsstils vor. Dies ist in Abbildung 2.3 dargestellt.

Dem **autoritären** Führungsstil [60, 41] folgend, legt die Lehrkraft sämtliche Richtlinien fest, schreibt Techniken/Tätigkeiten vor, lobt und tadelt nach persönlichen Gesichtspunkten und hält sich abseits der Gruppe. Als Auswirkungen werden größere Leistungsquantität, geringere Qualität, geringere Arbeitsmoral, höheres Konfliktpotenzial, und fehlender Arbeitseinsatz bei Abwesenheit des Lehrers [60] genannt.

Dem **demokratischen** Führungsansatz [60, 41] folgend, werden Richtlinien nach Diskussionen entschieden, hilft die Lehrkraft beim Entscheiden, schlägt stets Alternativen vor, orientiert sich bei der Bewertung streng an objektiven Kriterien und versucht Mitglied der Gruppe zu sein. Die Auswirkungen sind höhere Arbeitsmoral, weniger Aggressionspotential, besserer Arbeitseinsatz bei Abwesenheit des Lehrers, geringere Produktionsmenge und höhere Qualität.

Dem **laissez-fairen** Führungsansatz [60, 41] folgend, überlässt die Lehrkraft alle Entscheidungen gänzlich den Schülern, beschafft das Lernmaterial, gibt Informationen nur auf

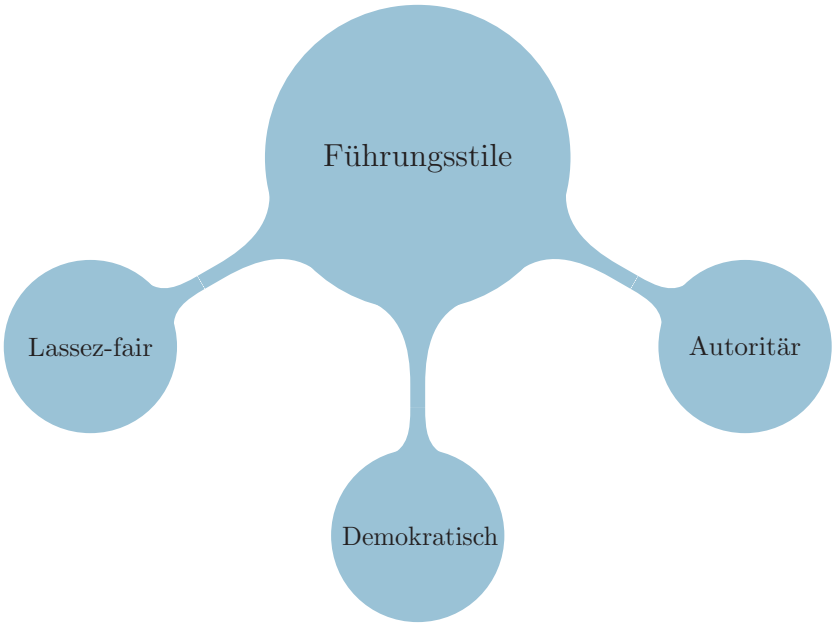


Abbildung 2.3: Führungsstile einer Lehrkraft nach [60].

Befragen, zeigt wenig Teilnahme, nimmt keine Beurteilung vor und verzichtet auf jede Regelung. Die Auswirkungen sind einfach voherzusehen: geringe Arbeits- und Gruppenmoral sowie geringe Produktivität.

Ein tatsächlicher, angemessener Führungsstil lässt sich natürlich nur schwierig durch eine dieser drei Kategorien definieren. Vielmehr ist eine situationsbezogene Führungsstilmischung mit Tendenz zum demokratischen Ansatz zielführend. Für den Lehrenden stellt sich die Auswahl des Führungsstils als eine Wanderung auf dem sprichwörtlichen dünnen Eis dar. Einerseits sollen für ein angenehmes Lernklima beim Lernenden positive Emotionen hervorgerufen werden, andererseits muss eine gewisse Autorität an den Tag gelegt werden, um den notwendigen organisatorischen Rahmen aufrechtzuerhalten.

2.5.2 Kooperatives Lernen

Kooperativ, von lat. cooperatio „Mitwirkung“ beschreibt das Zusammenbringen von Handlungen zweier oder mehrerer Personen. Umgangssprachlich findet sich die Definition etwas gemeinsam schaffen, wie anhand der chinesischen Kindergeschichte „Bá Luó Bo“ in Abbildung 2.4 angedeutet ist. Nur gemeinsam gelingt es die Rübe aus dem Boden zu ziehen.

Im Zusammenhang mit der Didaktik wird das Konzept des kooperativen Lernens oft als ein Überbegriff für Lernmethoden wie Partner- und Gruppenarbeiten verwendet. Pauli und Reusser definieren in [61] kooperatives Lernen folgendermaßen:

> Die Bezeichnung kooperatives Lernen wird sowohl für Gruppen- als auch für Partnerarbeitsformen im schulischen Unterricht verwendet. Damit sind Lern-

Abbildung 2.4: Kooperatives Verhalten - Bá Luó Bo – eine chinesische Kindergeschichte.

> arrangements gemeint, die eine synchrone und koordinierte, ko-konstruktive Aktivität der Teilnehmer verlangen, um eine gemeinsame Lösung eines Problems oder ein gemeinsam geteiltes Verständnis einer Situation zu entwickeln.

Kooperatives Lernen bedeutet daher auch, den Lernenden in der Gruppe zu vermitteln, wie wichtig Zusammenarbeit und unterstützende Interaktion ist. Johnson und Johnson [62, 63] teilen daher den Themenkreis des kooperativen Lernens in fünf grundlegende Elemente ein, die aufgrund ihrer Initialen im englischen Sprachraum auch als **PIGS-Face** [64] bekannt sind:

Positive Interdependence (Positive Abhängigkeit) Gemeinschaftsgefühl und Teamgeist tritt auf, wenn alle Mitglieder einer Gruppe sich miteinander darin verbunden fühlen, ein gemeinsames Ziel erreichen wollen. Damit die Gruppe Erfolg haben kann, muss jeder Einzelne erfolgreich sein. Wichtig dabei ist, dass ein einfaches Zusammensetzen von Schülern an einem gemeinsamen Arbeitsplatz noch lange kein gut arbeitendes Team ausmacht. Speziell für eine Informatikausbildung ist Teamfähigkeit – neben der fachlichen Kompetenz – eine wesentliche Grundvoraussetzung um für den späteren Berufsalltag gerüstet zu sein. Die Personalberatungsagentur Kienbaum weist in [65] aus, dass lt. einer Befragung von Unternehmen die Teamfähigkeit nach der Eigenmotivation die wichtigste Eigenschaft des künftigen Führungspersonals sei.

Individual Accountabiliy (Individuelle Verantwortungsübernahme) tritt auf, wenn jedes Mitglied einer Gruppe dafür verantwortlich ist, die Lernleistungen der Gruppe unter Beweis zu stellen. Hier soll vermieden werden, dass der/die „Beste“ der Gruppe die

Aufgabe übernimmt und die anderen dadurch im Endeffekt nur sehr wenig lernen und zu „Mitläufern“ werden. Dies kann vom Lehrenden gesteuert werden indem z.B.: individuelle Rollen zugewiesen werden (Protokollführer, Fachexperte, Gruppenvertreter, ...). Ist ein Arbeitsbericht anzufertigen können hier die Aufgabengebiete kapitelweise zugeteilt werden.

Group-Processing (Evaluation und Reflexion durch die Gruppe) tritt auf, wenn Gruppenmitglieder ihre gemeinsamen Anstrengungen beurteilen und Verbesserungen anstreben. Frei nach dem Leitsatz „Selbsterkenntnis ist der erste Schritt zur Besserung“ ist die Selbstevaluation der Gruppe ein hilfreiches Instrument zur Verbesserung der gesamten Gruppenleistung. Das Ziel der Gruppenreflexion besteht darin, dass die Effektivität der einzelnen Gruppenmitglieder im Hinblick auf die Erreichung des Gruppenziels eindeutig geklärt werden kann. Eine Möglichkeit ist, eine Reflexion auf zwei Ebenen einzuführen, auf Gruppen und Klassenebene.

Social Skills (Soziale Fähigkeiten) im Bereich des menschlichen Umgangs miteinander ermöglichen es Gruppen, effektiv zu funktionieren. Dazu gehört unter anderem: Aktiv zuhören, helfen und vermitteln, der Reihe nach sprechen, Ablenkungen widerstehen, Material teilen, Antworten ergänzen, andere loben und ermutigen, Anweisungen befolgen, ... Solche Fähigkeiten fördern Kommunikation, Vertrauen, Führungsqualitäten, Entscheidungsfreudigkeit und Konfliktmanagement. Eine eingehende Auseinandersetzung mit dem Begriff der sozialen Kompetenz findet sich in [66]. Konkrete Umsetzungen zum Thema soziales Lernen an Schulen werden in [67] erläutert.

Face-To-Face-Interaction (Direkte Interaktion) „von Angesicht zu Angesicht“ tritt auf, wenn sich Gruppenmitglieder in solch unmittelbarer Nähe zueinander befinden und so miteinander reden, dass dauerhafter Fortschritt gefördert wird.

Zusammenfassend, hat das kooperative Lernen das Potenzial Schüler auf die zukünftigen Herausforderungen des Berufslebens vorzubereiten, besonders durch die Fokussierung auf Teamarbeit, Teamgeist, soziale Kompetenz, usw. Die auftretenden Probleme beim praktischen Einsatz von kooperativen Lernen in Gruppen- und Partnerarbeiten, werden in [61, 68] wie folgt zusammengefasst:

Diskussionskultur: Viele Lernende diskutieren meist nur darüber, wie sie die Aufgabe schnell und mit geringstem Aufwand erledigen können, ohne sich tief mit dem eigentlichen Inhalt zu beschäftigen.

Oberflächlichkeit: Das Gespräch wird häufig auf oberflächliche Aspekte der Aufgabe, bzw. des Problems beschränkt, ohne die nötige Verstehenstiefe zu erreichen.

Meinungsverschiedenheiten: Diese werden lediglich durch soziales Aushandeln und nicht durch eine sachbezogene Argumentation gelöst.

Fragenstellung: Von den Lernenden werden nur selten spontan relevante Fragen gestellt, selten argumentiert und erklärt, sondern die Kommunikation wird auf einzelne

Wortfragmente beschränkt, zum Beispiel Aufforderungen.

Dialogbeteiligung: Schwächere Lernende nehmen nur selten aktiv am Dialog teil.

2.5.3 Frontalunterricht

Der Frontalunterricht, wie in Abbildung 2.5 schematisch dargestellt, ist in der modernen Didaktik häufig mit Kritik belegt, wenn er auch die dominierende Unterrichtsart in den heimischen Schulen darstellt. Es dominiert klar der Lehrende, der Lehrstoff wird im Sinne eines (einseitigen) Vortrages und des Lehrgespräches veranschaulicht und vermittelt. In der frontalen Lernsituation wird angenommen, dass alle Lernende gleichzeitig verstehen, lernen, begreifen und aufnahmebereit sind. Die Lernenden sitzen i.d.R. vor dem Lehrenden in parallelen Reihen. Die Kernelemente auf denen der Frontalunterricht basiert sind: vortragen, erzählen, berichten, demonstrieren, veranschaulichen, Lehrgespräche, usw. Die modernen Lehrmittel (Videoprojektor, Filme, Podcasts, ...) wirken unterstützend und lockern den oft monotonen Frontalunterricht auf. Besonders Präsentationen sollen zeitlich begrenzt sein und eine möglichst offene Form der Informationsweitergabe (frei nach dem Prinzip: „small is beautiful“) realisieren, welche die Mitarbeit der Lernenden anregen und fördern soll. Der Lernende selbst kann durch Fragen oder Beteiligungen den Unterricht nur wenig beeinflussen. Aufmerksamkeit, stillsitzen aber auch das sprachliche Verstehen muss gewährleistet sein.

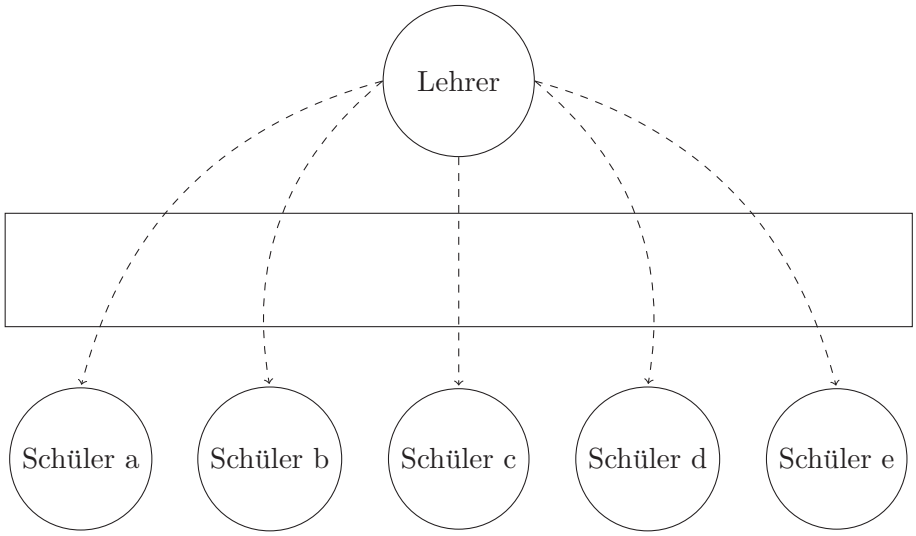


Abbildung 2.5: Der Frontalunterricht.

Als Vorteile des Frontalunterrichts finden sich in [69]:

- Die Lernzielorientierung ist am ehesten zu gewährleisten, wenn das Unterrichtsgeschehen allein vom Lehrer ausgeht.

- Die Kontrolle über den Unterrichtsverlauf ist vollkommener, wenn der Lehrer alles auf sich zentriert.

- Unterrrichtsplanung und offizieller Unterrichtsverlauf sind am ehesten zur Deckung zu bringen.

- Im lernzielorientierten Unterricht ist Lernen eine Form der Unterordnung, die analog zur künftigen Stellung des Lernenden in der Berufswelt geübt wird.

Werden die genannten Vorteile und die durchgängige Verbreitung des Frontalunterrichts im heimischen Schulsystem betrachtet, so lässt sich folgern, dass der Frontalunterricht auch aus einem modernen Unterricht nicht wegzudenken ist. Jedoch soll dieser nicht als alleinige Methode eingesetzt werden, sondern mit den nachfolgenden Weiterentwicklungen so gut als möglich kombiniert werden.

Weiterentwicklungen des Frontalunterrichts wie in [70] beschrieben, sind der darstellend-entwickelnde und der fragend-entwickelnde Unterricht, die im Folgenden kurz skizziert werden.

Darstellend-entwickelnder Unterricht

Die darstellend-entwickelnde Methode erweitert den Frontalunterricht (Vortrag) mit Zwischenfragen, ist dabei aber stets stark lehrerzentriert. Die Zwischenfragen werden verwendet um lange Vortragsettapen aufzulockern und den Lernenden Möglichkeiten zu geben, gerade Gelerntes zu reflektieren und eigene Fragen zu stellen. Um den Unterricht effektiv zu gestalten, bedarf es einen Vortragenden der sympathisch und mit hohen Entertainer Qualitäten einen fesselnden, interessanten Vortrag hält. Dieses Wunschbild des Lehrenden wird in der Realität leider nur von wenigen Individuen erfüllt.

Fragend-entwickelnder Unterricht

Das gelenkte Unterrichtsgespräch (fragend-entwickelnder Unterricht) baut auf Fragen die der Lehrende stellt auf. Diese Fragen sollen von möglichst vielen Lernenden beantwortet werden, um den Unterricht voranzubringen. Durch die Fragestellung kann der Lehrende das Unterrichtsziel steuern. In einigen Fällen wird gänzlich auf einen Vortrag verzichtet. Durch Umwandlung des Lehrstoffs in ein gezieltes Frage-Antwort Spiel sollen die Lernenden zu eigenen Einsichten und Schlussfolgerungen gelangen. Ähnlich zur darstellend-entwickelnden Methode stellt diese Unterrichtsmethode hohe Anforderungen an den Lehrenden. Es stellt sich die Herausforderung teilweise abstrakten Stoff in Fragen zu verpacken und dabei zu vermeiden, suggestiv die erwünschten Antworten bereits vorzugeben, oder nur rhetorische Fragen zu stellen, die der Lehrende letztendlich selbst beantwortet.

2.5.4 Gruppenarbeit

Die Gruppenarbeit hat es zum Ziel eine Organisationsform des Unterrichts zu schaffen, bei der neben dem Sachlernen auch das soziale Lernen im Mittelpunkt steht. Besonders im Informatikunterricht gilt es manchmal auch Konzepte wie Softwareapplikationen oder Programmiersprachen zu vermitteln, auf die in ihrer ganzen Breite aufgrund zeitlichen Einschränkungen und unterschiedlichen Vorkenntnissen der Lernenden nicht im Detail eingegangen werden kann. Eine Definition für Gruppenunterricht liefert Meyer in [71]:

> Gruppenunterricht ist eine Sozialform des Unterrichts, bei der durch die zeitlich begrenzte Teilung des Klassenverbandes in mehrere Abteilungen arbeitsfähige Kleingruppen entstehen, die gemeinsam an der von dem Lehrer, der Lehrerin gestellten oder selbst erarbeiteten Themenstellung arbeiten und deren Arbeitsergebnisse in späteren Unterrichtsphasen für den Klassenverband nutzbar gemacht werden können. Gruppenarbeit ist die in dieser Sozialform von den Schüler/innen und dem Lehrer, der Lehrerin geleistete zielgerichtete Arbeit, soziale Interaktion und sprachliche Verständigung.

Die Methodik Gruppenarbeit erlaubt es, den Unterricht individualisiert zu erstellen und besonders die Methodenkompetenz der Lernenden zu trainieren. In [63] werden fünf Merkmale des Gruppenunterrichts hervorgehoben, die auch auf den Informatikunterricht angewendet werden können [45]:

1. Gruppenarbeiten fördern den Zusammenhalt unter den Lernenden. Entweder geht man gemeinsam unter oder man schwimmt gemeinsam. Gemeinsam „schwimmen“ lernen ist hilfreich. Ein Einzelner kapituliert schneller als eine ganze Gruppe.

2. Die Zusammenarbeit in einer Gruppe wirkt motivierend und beschleunigt den Lernprozess.

3. In einer Gruppe muss jedes Mitglied Verantwortung übernehmen und einen Beitrag zur Zielerreichung leisten.

4. Gruppenarbeiten fördern Sozialkompetenzen wie Kommunikation in einem Team, Vertrauen, Entscheidungsfindung in einer Projektgruppe und Umgang mit Konflikten.

5. Gruppenarbeiten fördern das Nachdenken über gruppendynamische Prozesse und tragen damit zur Effizienzsteigerung bei der Arbeit in Informatikprojekten bei.

In der Literatur [71] und Praxis finden sich viele Formen des Gruppenunterrichts. Für diese Arbeit sind der Puzzleunterricht, die Partnerarbeit und die Projektarbeit von Relevanz und werden daher im Folgenden dargestellt.

Der Puzzleunterricht - The jigsaw teaching technique

Eine spezielle Form der Gruppenarbeit stellt der Puzzleunterricht dar. Diese Methodik [72, 73] geht auf den Psychologen Elliot Aronson zurück, der diese 1971 vorstellte um Probleme zwischen Schülern unterschiedlicher Herkunft zu lösen die gemeinsam unterrichtet wurden. Die n Lernenden werden auf ungefähr $\sqrt{n}$ Gruppen aufgeteilt. Die Lernenden erarbeiten zuerst einen Themenkreis für sich alleine, gleichen in einer sog. Expertengruppe ihre Ergebnisse ab und danach wird jeder dieser Experten zum Lehrenden und unterrichtet die anderen. Dieser Ablauf ist in Abbildung 2.6 schematisch dargestellt. Zuerst werden in einer Gruppe von vier Lernenden ein Thema erarbeitet und diskutiert (Abbildung 2.6 Mitte). Danach kehrt jeder Lerner in seine Ursprungsgruppe zurück und wird vom Lerner zum Lehrer und gibt sein Wissen an die Gruppe weiter.

Es kommen daher drei Phasen zur Anwendung (i) Erarbeiten, (ii) Abgleichen in der Expertenrunde und (iii) Weitergeben an Mitschüler. Die Hauptaufgabe des Lehrers liegt hier in der Vorbereitung. Während des Unterrichts ist es seine Aufgabe den Ablauf zu organisieren und zu koordinieren. Eine der Vorteile dieser Methode liegt in der geforderten Selbstständigkeit unter Verfolgung der Ziele des kooperativen Lernens, einer Förderung des Selbstbewusstseins und Training der Eigenverantwortung.

Um die Puzzlemethode effizient zu realisieren, bedarf es eines guten Zeitmanagements des Lehrers und einem Abgleich der Schnittstellen zwischen den individuellen Lernphasen. Weiters ist zu beachten, dass die Geschwindigkeit der Lernenden beim Erarbeiten des Themas teilweise stark unterschiedlich sein kann. Es gilt daher Zusatzmaterialien vorzubereiten, oder die Themen des Gruppenpuzzles mit unterschiedlicher Schwierigkeit auszuwählen. Ein Nachteil der Puzzlemethode ist, dass die individuellen Lernenden den Stoff nicht gleich gut bearbeiten. Die Experten werden das eigene Thema besser verstehen, als sie es vermitteln können. Daher eignet sich diese Methode besonders gut in Kombination mit Themen die viele Einzelaspekte aufweisen, die nicht alle in ihrer ganzen Tiefe bearbeitet werden müssen. Als Einführung in grundlegende Konzepte eignet sich daher die Puzzlemethode eher schlecht.

Partnerarbeit

Die Partnerarbeit stellt die kleinste Organisationsform des Gruppenunterrichts dar. Der Lehrende verteilt die Aufgaben meist an Paare zu je zwei Lernenden, die diese Aufgabe dann im Team bearbeiten. Typische Partnerarbeiten dauern selten länger als eine Stunde, 10-30 Minuten stellen die Regel dar.

Durch diese kurze Dauer lassen sich Blöcke mit Partnerarbeiten elegant und nahtlos in den Frontalunterricht integrieren. Im Gegensatz zur Puzzlemethode oder zur Projektarbeit erhält jede Gruppe kleine, sehr präzise Aufgabenstellungen. Meist beziehen sich diese Aufgaben auf einen Gedankenaustausch, Diskussion und gegenseitiges Erklären von

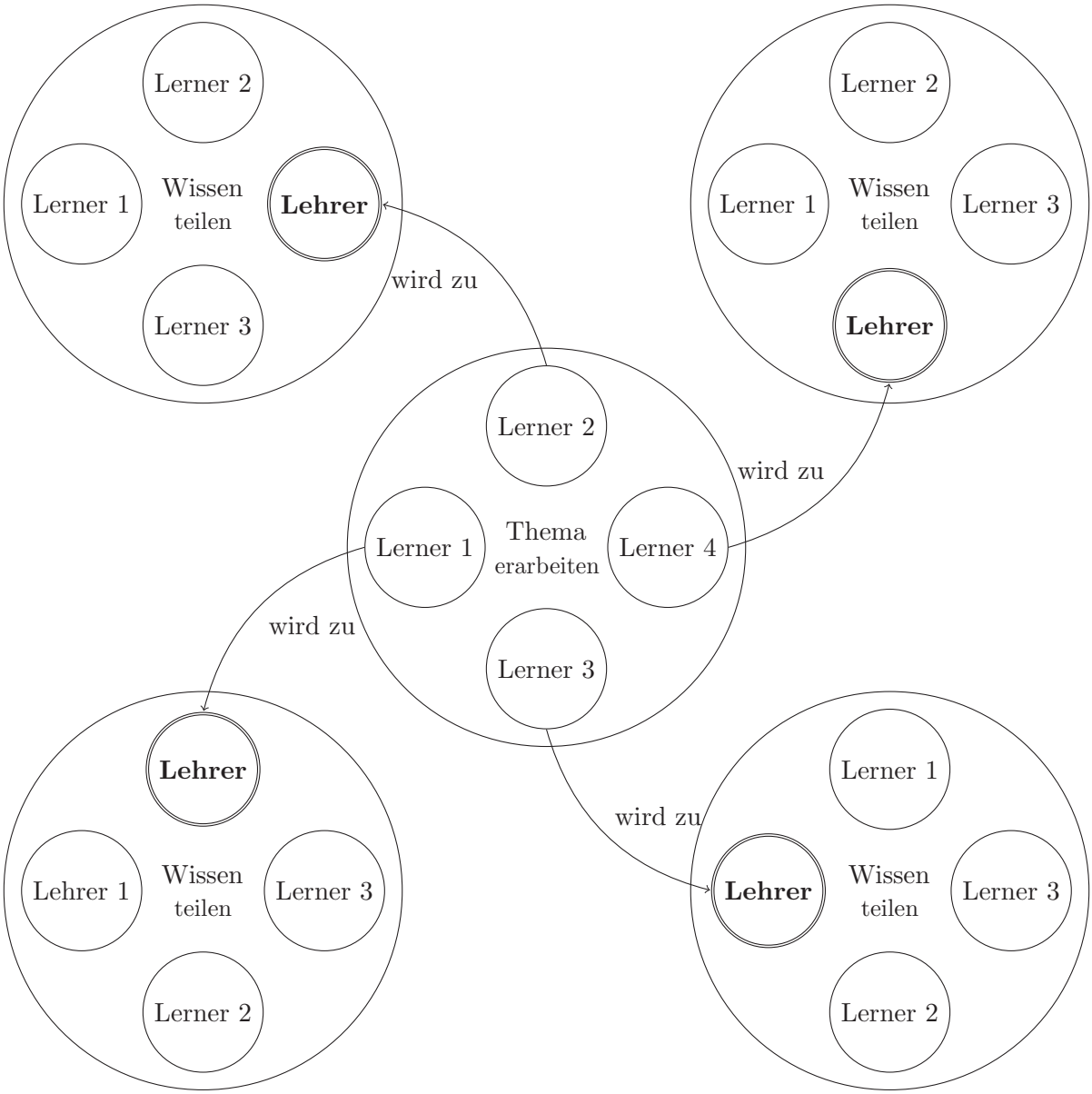


Abbildung 2.6: Der Puzzleunterricht.

Sachinhalten.

Als eine für den Informatikunterricht typische Partnerarbeit wird in [45] das Erklären eines Sortieralgorithmus vorgeschlagen. Hier kommen zwei verschiedene Rollen zum Einsatz. Ein Partner erteilt dem anderen gemäß einem vorgegebenen Programm Anweisungen, während der andere den eigentlichen Sortieralgorithmus anhand von Spielkarten ausführt. Besonders bei Themen der Informatik (Verschlüsselung und Entschlüsselung, Kryptographie, Steuerprogrammen, Mikrocontroller Anweisungen, usw.) tragen unterschiedliche Sichtweisen bedeutend zum Verstehen bei.

Projektarbeit

Eine Projektarbeit ist eine Unterrichtsform, die sich an einer Durchführung eines Industrieprojekts orientiert. Der Fokus liegt auf der eigenständigen Bearbeitung einer Problemstellung durch eine Gruppe, angefangen von Initiierung über Planung und Durchführung bis zur Reflexion. Die einzelnen Phasen der Projektarbeit aus der Gliederung in [74] sind in Abbildung 2.7 dargestellt. Dieses Modell korreliert großteils mit dem siebenstufigen Modell von Frey in [75]. Diese Abläufe können auch ohne große Änderung auf industrielle Projektmanagement Prozesse angewandt werden, die ähnliche Strukturen aufweisen.

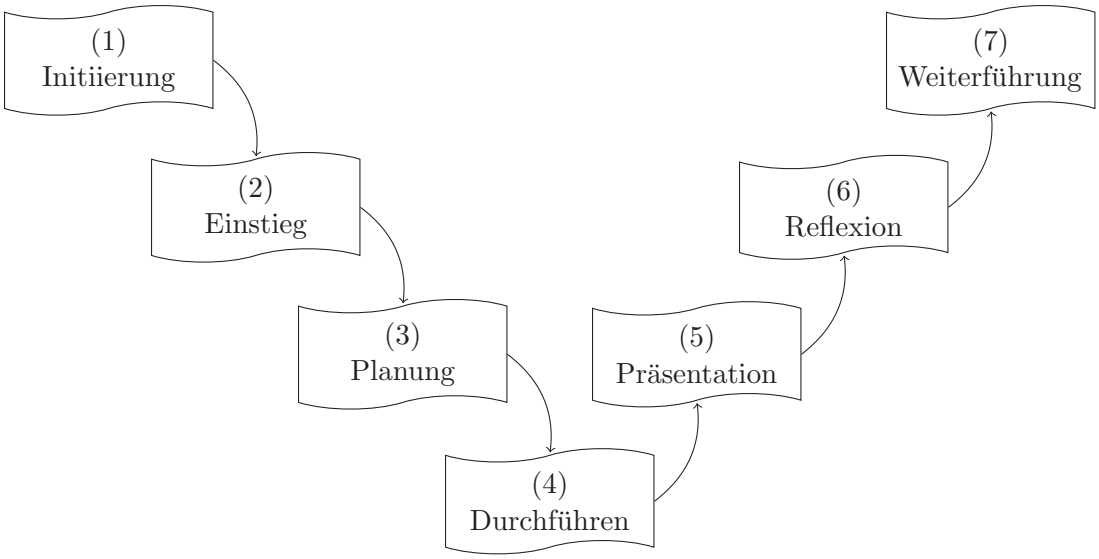


Abbildung 2.7: Der Ablauf der Projektarbeit.

In Phase (1) – Initiierung – werden Probleme, Aufgaben und Ideen aufgeworfen. Die Fragestellung kann vom Lehrer, von den Lernenden oder aus einer dritten Quelle stammen. Das Projekt soll möglichst offen gestaltet sein, es soll vermieden werden, dass sich die Aufgabe auf das Lösen eines klaren und präzisen Problems beschränkt.

In Phase (2) – Einstieg – geht es um das Präzisieren von ersten vagen Projektideen. Aufgrund der beschränkten Ressourcen (Zeit, Vorkenntnisse, Material, Motivation) kann

im realen Schulunterricht meist nicht das Projekt in seiner Gänze verwirklicht werden. In dieser Phase ist der Lehrende gefordert, Projektziele zu bewerten, einzugrenzen und Schwerpunkte zu setzen.

In Phase (3) – Planung – wird ein Arbeitsplan erstellt. Bekannte Ansätze sind hier der Projektstrukturplan, Meilensteinplan, Gantt-Charts, sowie Top-Down und Bottom-Up approach. Es ist wichtig, dass Zielsetzungen, Informationsfluss und Vorgehensweise dokumentiert und niedergeschrieben werden.

In Phase (4) – Durchführung – werden die zuvor in Phase (3) geplanten Aktivitäten auch in die Realität umgesetzt. Zusätzlich werden die erledigten Projektarbeiten mit der Planung verglichen. Je nach Projektfortschritt wird die Planung revidiert.

In Phase (5) – Präsentation – werden die erreichten Projektziele vor einem vorzugsweisen größeren Plenum vorgetragen.

In Phase (6) – Reflexion – sollen die anderen Teilnehmer zur Diskussion und Bewertung der Resultate miteinbezogen werden.

In Phase (7) – Weiterführung – wird über den Nutzen des Projekts für die Schule und eine eventuelle Wiederverwendung von Teilen des Projekts diskutiert. Synergien entdecken und das Beste daraus machen ist hier die Devise. Wird beispielsweise ein Datenbanksystem im Informatikunterricht erarbeitet, kann dies für die Wissenssammlung in einem Intranet Portal für – beispielsweise – den Biologieunterricht eingesetzt werden.

Eine Projektarbeit fordert von den Lernenden demokratisches Handeln und trainiert das handlungsorientierte, problemlösungsbasierte Lernen. Den einzelnen Teilnehmern wird Eigenverantwortung übertragen. Seitens der Lehrperson sind der Projektplanung auch viele Grenzen gesetzt. Meist gibt es nur einen strikt begrenzten Zeitraum, in dem gewisse, punktuelle Informationen vermittelt werden müssen. Auch der Themenbereich muss sich nach den Lehrplänen des Schultyps orientieren und unterliegt somit gewissen Einschränkungen.

Ein signifikantes Kriterium für das Gelingen des Projekts ist die Zusammensetzung der Projektgruppen. Es ergeben sich verschiedene Ansätze (Frei nach [76]):

1. **Autoritäre Zusammenstellung:** Der Lehrende stellt die Projektgruppen zusammen, es ergeben sich folgende Vor- und Nachteile:
   - Hohes Risiko in der Frage Sympathie und Antipathie kann Leistungsbereitschaft und Kreativität hemmen
   - Gute und gerechte Bewertungschancen, da Leistungsniveau ausgeglichen gestaltet werden kann
   - Gefühl der Bevormundung und absoluten Kontrolle
2. **Demokratische Zusammenstellung:** Die Lernenden finden sich selbst zu Projektgruppen zusammen, es ergeben sich folgende Vor- und Nachteile:

- Gutes Arbeitsklima, da Sympathie unter den Projektmitgliedern
- Gruppe hat aber schon a priori eine feste Rollenverteilung
- Hohes Risiko in Fragen der Leistungsbewertung, da Schüler meist unterschiedliches Leistungsniveau haben (Mitschwimmer)
- Gruppe hat von Anfang an das Gefühl der selbstständigen Arbeit und Verantwortung

3. **Anonyme Zusammenstellung:** Das Auswahlverfahren erfolgt per Zufall oder durch individuelle Neigung. Es ergeben sich folgende Vor- und Nachteile:
   - Gruppenbildung erfolgt ohne Zwang des Lehrers oder Betreuers
   - Teams sind meistens leistungsmäßig stark gestreut
   - Es bildet sich aber auf Grund des Drucks der Aufgabenstellung und der Nichteinmischung des Lehrers meist eine gute Arbeitsathmosphäre
4. **Alternative Zusammenstellung:** Die Auswahl der Projektmitglieder ist unter gewissen Bedingungen frei, z.B.: Leistungsklassen. Es ergeben sich folgende Vor- und Nachteile:
   - Gute Voraussetzung für gerechte Bewertung
   - Gute Voraussetzung für eine homogene Gruppenbildung

Der Projektunterricht ist daher prädestiniert den Lernenden einen Einblick in die realen Probleme, die bei Informatikprojekten auftreten, zu ermöglichen.

2.5.5 Portfolio

Unter Portfolio im Bildungszusammenhang vesteht man eine individuell zusammengestellte Sammlung zum persönlichen Nachweis und Dokumentation von Lernleistungen. Das Portfolio hält somit den individuellen Lernfortschritt fest, und ist mehr als die langläufig bekannte „Mitschrift“. Der Portfoliounterricht ist auf zwei Säulen aufgebaut:

Portfolioprozess ist das Sammeln, kritische Sichten, und Auswählen von aussagekräftigen Lerndokumenten zur Steigerung der eigenen (Fach)kompetenzen

Portfolioprodukt ist eine spezifisch zusammengestellte und kommentierte Präsentation von Dokumenten zum Nachweis des Lernfortschritts

Dabei soll der Lernende in die Auswahl der Inhalte und Themen sowie die verwendeten Darstellungsmöglichkeiten und den Beurteilungskriterien aktiv miteinbezogen werden.

Der Mehrwert des Portfoliounterrichts liegt daher beim (i) Fördern von Eigenverantwortung, (ii) Vermitteln von Strukturierungskompetenz und Reflexionskompetenz beim

Erarbeiten neuer Themengebieten, (iii) Offenlegen des eigenen Entwicklungsprozesses, (iv) Ankurbeln des Theorie-Praxis Transfers und (v) Entwicklen von fachübergreifenden Kompetenzen.

Der Lehrende muss für die Ausführung der Portfolios klare Rahmenbedingungen hinsichtlich Ziele, Verbindlichkeit, Umfang, Erwartungen, Bewertung, usw. schaffen. Weiters sind Hilfen bei der Einarbeitung in das Thema sowie der eigentlichen Durchführung anzubieten. Eine umfassende Behandlung des Portfolio Unterricht findet sich in [77]. Eine praktische Betrachtung in [78].

2.5.6 Weitere Unterrichtsmethoden

Neben den bereits vorgestellten Unterrichtsmethoden gibt es eine Vielzahl von alternativen Unterrichtsmethoden die einen individualisierten Unterricht ermöglichen. Ein gesamtheitlicher Überblick findet sich in [79].

3 Der SOLL Zustand

Im folgenden Kapitel wird durch zwei Interviews der Standpunkt der Industrie zum Thema Software Verifikation und -Test einerseits und die Anforderungen an künftige HTL Absolventen andererseits stichprobenartig erfasst.

3.1 Interview mit Vertretern der Industrie – Teil I

3.1.1 Dipl.-Ing. Christian Galbavy – Zur Person

Das folgende Interview wurde mit Dipl.-Ing. Christian Galbavy am 12-03-2010 aufgezeichnet. Christian Galbavy ist Absolvent der TU Graz (Telematik) und ist in einem großen österreichischen Technologieunternehmen Projektleiter für Telematiksysteme. Als Mitarbeiter in Projektteams hat er langjährige Erfahrung in der Bedeutung und dem abgestimmten Zusammenspiel von Systemdesign, Requirements Engineering, Softwareentwicklung und -tests.

Dipl.-Ing. Christian Galbavy
Project Management Officer
Siemens AG Austria
Erdbergerlände 26
A-1030 Wien, Austria
e-mail: christian.galbavy@siemens.com
web: **http://www.mobility.siemens.com**

3.1.2 Interview

Anforderungen an Bewerber

$\mathcal{Q}.1$ Herr Galbavy, Sie sind Projektleiter für Telematiksysteme und arbeiten somit in großen internationalen Teams. Welche Eigenschaften (fachlich und SoftSkills) schätzen Sie bei neuen Teammitgliedern (Absolventen) besonders?
$\mathcal{A}.1$ *Absolventen sollten eine fundierte technische Ausbildung haben sowie bereits Arbeitserfahrung (Werkstudententätigkeit, Ferialpraxis, etc.) mitbringen. Gute Englischkenntnisse sind eine Grundvoraussetzung. Weiters erwarten wir die Fähigkeit in großen Teams arbeiten zu können.*

$\mathcal{Q}.2$ Für ein durchschnittliches SW Projekt in Ihrem Unternehmen: Wie schätzen Sie das Verhältnis von (i) SW Spezifikation, (ii) SW Entwicklung (Implementierung) und (iii) SW Test und Verifikation?
$\mathcal{A}.2$ *30%, 40%, 30%.*

$\mathcal{Q}.3$ Welche Erfahrungen haben Sie mit HTL Absolventen gemacht?
$\mathcal{A}.3$ *Viele unserer Schlüsselleute in Projekten sind HTl-Absolventen mit langjähriger Arbeitserfahrung. Absolventen von Universitäten bringen jedoch meist ausgeprägtere Problemlösungsfähigkeiten mit.*

$\mathcal{Q}.4$ Bitte beschreiben Sie kurz die Qualifikationen die Sie sich von einem HTL Absolventen wünschen, wenn dieser in Ihrem Team mitarbeiten soll? (im Allgemeinen und für SW Entwicklung)
$\mathcal{A}.4$ *Kenntnisse sowohl in Software- und Hardwareentwicklung, Arbeitserfahrung, gute Englischkenntnisse und Teamfähigkeit. Vor allem grundlegendes Verständnis von Softwareentwicklung und objektorientierter Programmierung. Wissen zu Entwicklungsmethodiken und Genauigkeit.*

$\mathcal{Q}.5$ Wenn Sie Softwareentwickler einstellen, wie wichtig ist ihnen dabei Fachwissen im Bereich (formale) Verifikation und Testen von Software?
$\mathcal{A}.5$ *Software Testing wird bei uns vor allem durch erfahrene Entwickler durchgeführt, die vor allem intern trainiert werden. Neu angestellte Mitarbeiter werden eher im Entwicklungsbereich eingesetzt.*

Test und Verifikationspraktiken im Unternehmen

$\mathcal{Q}.6$ Welche Methoden zur Softwareverifikation verwenden Sie in Ihren Produkten?
$\mathcal{A}.6$ *Code Reviews, Component Tests, Integrationstests, Systemtests, Feldtests (Large Scale Testing), Hardware-Tests (EMV, Shock, etc.).*

$\mathcal{Q}.7$ Für die folgenden Begriffe geben Sie bitte kurz an ob und wie Sie diese in Ihren Firmenalltag einsetzen:
$\mathcal{A}.7$ *V Modell: täglich, Integrationstest: mehrmals im Projekt, Unit Testing: täglich, Black Box Testing: täglich, Statische Software Analyse: täglich, Formale Verifikationsmethoden: selten, Theorem Proving: selten, Model Checking: häufig, Formale Spezifikation: selten.*

Europäischer Qualifikationsraster und Vergleichbarkeit der Qualifikationen

$\mathcal{Q}.8$ Sie arbeiten in einem international aufgestellten Unternehmen. Nach welchen Kriterien bewerten Sie die Qualifikationen der Bewerbungen aus dem Ausland? Wie vergleichen Sie Qualifikationen die in verschiedenen Ländern erworben wurden?
$\mathcal{A}.8$ *Personalverantwortung liegt in unserem Unternehmen bei der Geschäftsleitung, wobei Projektleiter sehr wohl bei Personalentscheidungen nach Ihrer Meinung gefragt werden. Ich persönlich beurteile Bewerbungen anhand des Abschlusses (Höhere Schule, Studium, etc.) und vor*

allem Arbeitserfahrung. Dazu erscheint mir der Europäische Lebenslauf gut geeignet. Der Rest entscheidet sich anhand der angegebenen (technischen Kenntnisse) und beim Bewerbungsgespräch.

$\mathcal{Q}.9$ Nach dem Bologna-Abkommen sind Studien nun in einer Bachelor/Master Organisation zu führen. Wenn Sie nun zwei Bewerber haben, einer mit einem Bachelor Abschluss, einer mit einem HTL Abschluss, für wen entscheiden Sie sich tendenziell eher?
$\mathcal{A}.9$ *Das hängt von der zu besetzenden Position ab und wie gesagt von der Arbeitserfahrung. Einen Bachelor, der neben dem Studium bereits als Werkstudent gearbeitet hat, würde ich eher nehmen als einen HTL-Absolvent.*

$\mathcal{Q}.10$ Das bm:ukk implementiert derzeit den von der EU beschlossenen European Qualifications Framework (EQF) zur Transparenz der Ausbildungen im europäischen Raum. Welche Vorteile sehen Sie dafür für Ihr Unternehmen in Bezug auf den Rekrutierungsprozess? Würden Sie sich wünschen Qualifikationen besser vergleichen zu können?
$\mathcal{A}.10$ *Da unser Unternehmen weltweit aufgestellt ist, sind unsere Recruiter tagtäglich damit konfrontiert, unterschiedliche Qualifikationen zu bewerten. Hierbei haben unsere Leute viele Jahre Erfahrung, welche Kriterien dazu einsetzbar sind. Vorteile sehe ich persönlich in einer Bewertungsstandardisierung nicht, da bei einer Anstellung auch sehr viel vom menschlichen Charakter, Motivation und Teamfähigkeit abhängt.*

$\mathcal{Q}.11$ Wenn Sie auf Ihre eigene Ausbildung (AHS, BHS, Studium) zurückblicken, wovon profitieren Sie heute im Berufsleben am meisten?
$\mathcal{A}.11$ *Vor allem durch Berufserfahrung während der Schule und des Studiums. Weiters von mehreren Langzeit-Auslandsaufenthalten (sowohl in der Ausbildung als auch privat). Im Bereich des Studiums war eine Ausbildung zum Generalisten sehr sinnvoll.*

$\mathcal{Q}.12$ Sonstiges, das Sie noch gerne über Software Test und Verifikation in der Ausbildung los werden wollen?
$\mathcal{A}.12$ *Meiner Erfahrung nach erhalten Studenten (zumindest in meinem Studium) bereits eine gute Basis zum Thema „Testen“. In der HTL-Ausbildung sollte man vor allem bei Programmier-/Laborprojekten mehr Wert auf Tests legen und diese in die Aufgabenstellungen mitaufnehmen.*

3.1.3 Interpretation und Schlussfolgerungen

Das Interview mit Dipl.-Ing. Christian Galbavy gibt einen interessanten Einblick in die industrielle Praxis der Entwicklung von sicherheitskritischen Anwendungen, in diesem speziellen Fall Mautsysteme. Aus seiner Einschätzung lassen sich für diese Arbeit mehrere Schlussfolgerungen ziehen:

- Wichtig sind neben den fachlichen Qualifikationen Teamfähigkeit, Arbeitserfahrung und sprachliche Kompetenz (Englisch) (vgl. Q/A1, Q/A11).

- Viele Schlüsselpositionen sind von ehemaligen HTL Absolventen besetzt. (vgl. Q/A3).
- Ein Modell zum internationalen Kompetenzenvergleich wie der Europäische Qualifikations Rahmen ist für das Unternehmen nicht unbedingt erforderlich, erworbene Qualifikationen werden nach Erfahrungen der Personalabteilung beurteilt (vgl. Q/A10).
- Im beruflichen Alltag sind Test und Verifikation der entwickelten Systeme ein Schlüsselfaktor. Der Arbeitsaufwand für Verifikation ist größer als für die eigentliche Programmerstellung (vgl. Q/A2, Q/A6, Q/A7, Q/A12).
- Model Checking, statische Analyse und verschiedene Testverfahren sind Kernelemente der täglichen Softwareentwicklung (Q/A7).

3.2 Interview mit Vertretern der Industrie – Teil II

3.2.1 Dipl.-Ing. Dr. Stephan Grünfelder – Zur Person

Das folgende Interview wurde mit Dipl.-Ing. Dr. Stephan Grünfelder am 17-03-2010 aufgezeichnet. Stephan Grünfelder ist Trainer für Software-Testing, Software-Technologie und Requirements-Engineering. Er stützt sich auf langjährige Erfahrung als Projektleiter, Senior Software Entwickler und Tester. Sein Wissen gibt er gerne an Universitäten, in Seminaren und Workshops für Industriekunden (vorwiegend aus dem Bereich Embedded Systems) und in der Zeitschrift Elektronik weiter.

Dipl.-Ing. Dr. Stephan Grünfelder
Trainer für Software-Testing, Software-Technologie und Requirements-Engineering
RUAG Space GmbH, Stachegasse 16
A-1120 Wien, Austria
e-mail: stephan.gruenfelder@ruag.com
web: **http://www.freewebs.com/methver/**

3.2.2 Interview

Anforderungen an Bewerber

$\mathcal{Q}.1$ Herr Grünfelder, Sie sind für RUAG Aerospace den größten österreichischen Hersteller für Satellitensysteme tätig und arbeiten somit in großen internationalen Teams. Welche Eigenschaften (fachlich und SoftSkills) schätzen Sie bei neuen Teammitgliedern (Absolventen) besonders?
$\mathcal{A}.1$ *Problemlösungsfähigkeit und Teamfähigkeit halte ich neben einer guten fachlichen Ausbildung für sehr wichtig. Weiters müssen alle unsere Mitarbeiter gut Englisch können und von vielen wird*

das Verfassen von Dokumenten erwartet.

$\mathcal{Q}.2$ Für ein durchschnittliches SW Projekt in Ihrem Unternehmen: Wie schätzen Sie das Verhältnis von (i) SW Spezifikation, (ii) SW Entwicklung (Implementierung) und (iii) SW Test und Verifikation?
$\mathcal{A}.2$ *Etwa 20%, 35%, 45%.*

$\mathcal{Q}.3$ Welche Erfahrungen haben Sie mit HTL Absolventen gemacht?
$\mathcal{A}.3$ *Unser bester Mann hatte bis vor 5 Jahren als höchsten Abschluss eine HTL, bis er im Abendstudium, einen Dipl-Ing-FH gemacht hat.*

$\mathcal{Q}.4$ Bitte beschreiben Sie kurz die Qualifikationen die Sie sich von einem HTL Absolventen wünschen, wenn dieser in Ihrem Team mitarbeiten soll? (im Allgemeinen)
$\mathcal{A}.4$ *Kenntnisse von RTOS, Programmiersprache C, Kenntnisse von Mirkroprozessoren, Genauigkeit, fundiertes Englisch.*

$\mathcal{Q}.5$ Wenn Sie Softwareentwickler einstellen, wie wichtig ist ihnen dabei Fachwissen im Bereich (formale) Verifikation und Testen von Software?
$\mathcal{A}.5$ *Formale Verifikation: uninteressant. Verifikation (allgemein): lernt er bei uns gemäß ESA-Richtlinien, wir stellen keine Ansprüche. Test: bislang haben wir wenige Leute einstellen können, die schon Test-Know-How haben. Eine Ausnahme stellt Ihr Kollege Stefan Lechner dar, er hat Embedded Systems an der FH gemacht*[1] *und dort schon über Testing einiges gehört. Er ist jetzt mit Testaufgaben betraut.*

Test und Verifikationspraktiken im Unternehmen

$\mathcal{Q}.6$ Welche Methoden zur Softwareverifikation verwenden Sie in Ihren Produkten?
$\mathcal{A}.6$ *Code Inspection, statische automatische Analyse (PC-Lint), Unit Test, Integrationstest, Software-Systemtest (funktional und nicht-funktional)*

$\mathcal{Q}.7$ Für die folgenden Begriffe geben Sie bitte kurz an ob und wie Sie diese in Ihren Firmenalltag einsetzen:
$\mathcal{A}.7$ *V Modell: täglich, Integrationstest: häufig, Unit Testing: täglich, Black Box Testing: täglich, Statische Software Analyse: täglich, Formale Verifikationsmethoden: gar nicht, Theorem Proving: gar nicht, Model Checking: Zur Zeit nicht. Ggf. eines Tages. Vermutlich aber nicht, Formale Spezifikation: Gar nicht.*

[1] Masterstudiengang: Embedded Systems, Fachhochschule Technikum Wien

Europäischer Qualifikationsraster und Vergleichbarkeit der Qualifikationen

$\mathcal{Q}.8$ Sie arbeiten in einem international aufgestellten Unternehmen. Nach welchen Kriterien bewerten Sie die Qualifikationen der Bewerbungen aus dem Ausland? Wie vergleichen Sie Qualifikationen die in verschiedenen Ländern erworben wurden?
$\mathcal{A}.8$ *Tun wir meines Wissens nur nach Bauchgefühl.*

$\mathcal{Q}.9$ Nach dem Bologna-Abkommen sind Studien nun in einer Bachelor/Master Organisation zu führen. Wenn Sie nun zwei Bewerber haben, einer mit einem Bachelor Abschluss, einer mit einem HTL Abschluss, für wen entscheiden Sie sich tendenziell eher?
$\mathcal{A}.9$ *Ich persönlich würde den HTLer nehmen.*

$\mathcal{Q}.10$ Das bm:ukk implementiert derzeit den von der EU beschlossenen European Qualifications Framework (EQF) zur Transparenz der Ausbildungen im europäischen Raum. Welche Vorteile sehen Sie dafür für Ihr Unternehmen in Bezug auf den Rekrutierungsprozess? Würden Sie sich wünschen Qualifikationen besser vergleichen zu können?
$\mathcal{A}.10$ *Es ist uns wichtiger ob ein Mitarbeiter sich bei uns wohl fühlt und teamfähig ist und dadurch Performance zeigt, als wenn wir Ausbildungsstände vergleichen. Weiters haben wir wenige Bewerber aus dem europäischen Ausland.*

$\mathcal{Q}.11$ Wenn Sie auf Ihre eigene Ausbildung (AHS, BHS, Studium) zurückblicken, wovon profitieren Sie heute im Berufsleben am meisten?
$\mathcal{A}.11$ *Programmierkenntnisse, Kenntnisse über Betriebssysteme und Software-Architektur, Englisch.*

$\mathcal{Q}.12$ Sonstiges, das Sie noch gerne über Software Test und Verifikation in der Ausbildung los werden wollen?
$\mathcal{A}.12$ *Ja: (Software Test und Verfikation) Wird zur Zeit zu wenig unterrichtet. Alle wollen SW schreiben aber niemand testen. Bitte keine neue Ausbildung erfinden, sondern mit Organisationen wie ISTQB[2] zusammenarbeiten und diese Inhalte in HTL-Lehrpläne etc. übernehmen.*

3.2.3 Interpretation und Schlussfolgerungen

Das Interview mit Dipl.-Ing. Dr. Stephan Grünfelder gibt einen interessanten Einblick in die industrielle Praxis der Herstellung von sicherheitskritischen Anwendungen. Daraus lassen sich für diese Arbeit mehrere Schlussfolgerungen ziehen:

- Wichtig ist neben den fachlichen Qualifikationen auch sprachliche Kompetenzen, vor allem Englisch (vgl. Q/A1, Q/A11).
- HTL Absolventen genießen höchstes Ansehen in der Industrie. (vgl. Q/A9, Q/A3).

[2] Das International Software Testing Qualifications Board (ISTQB) hat das Ziel, eine standardisierte Ausbildung für professionelle Softwaretester auszuarbeiten

- Ein Modell zum internationalen Kompetenzenvergleich wie der Europäische Qualifikations Rahmen ist für das Unternehmen nicht unbedingt erforderlich, erworbene Qualifikationen werden nach „Bauchgefühl“ beurteilt (vgl. Q/A10).
- Im beruflichen Alltag der sicherheitskritischen Softwareentwicklung nehmen Test und Verifikation einen wesentlichen Anteil ein. Der Arbeitsaufwand für Verifikation ist größer als für die eigentliche Programmerstellung (vgl. Q/A2, Q/A6, Q/A7).
- Mehr Ausbildung in Richtung Test und Verifikation von Software ist wünschenswert und wäre ein bedeutender Vorteil bei der Bewerbung (vgl. Q/A12).

4 Der IST Zustand

4.1 Die elektrotechnische Ausbildung in der HTL

Der aktuell gültige Lehrplan [80] für die höhere Abteilung der HTL für Elektrotechnik (ET) bietet drei verschiedene Ausbildungsschwerpunkte – sog. Spezialisierungen – an, die der Lernende typischerweise am Anfang des 3ten Jahrgangs wählen kann:

Energietechnik und Industrielle Elektronik (ETE) bietet eine Spezialisierung auf wirtschaftliche und umweltgerechte Erzeugung, Verteilung und Anwendung von elektrischer Energie sowie die industrielle Nutzung elektronischer Bauteile und Geräte. Weitere Schwerpunkte sind die elektrische Antriebstechnik, deren Leistungselektronik und Regelungstechnik sowie Beleuchtungs- und Wäremetechnik [80, 27].

Regelungstechnik (ETR) bietet eine Spezialisierung auf die Erfassung, Aufbereitung und Verarbeitung von Messdaten für die Automatisierung industrieller Prozesse. Weitere Schwerpunkte sind der Entwurf und Dimensionierung von Steuerungs- und Regelungseinrichtungen, Antriebssystemen sowie der Einsatz von zugehöriger Software [80, 27].

Informationstechnik (ETI) bietet eine Spezialisierung auf die Informationserfassung, Informationsverarbeitung und Informationsdarstellung sowie der Prozessautomatisierung. Weitere Schwerpunkte liegen dabei auf der Prozessdatenverarbeitung, Leittechnik von Industrieanlagen, Vernetzung industrieller Systeme und deren Anbindungen an übergeordnete Informationsnetze [80, 27].

4.1.1 Der aktuelle Lehrplan

Tabelle 4.1 stellt den aktuell gültigen Lehrplan dar. Für die in dieser Arbeit vorgestellten Konzepte zur Lehre von (formaler) Verifikation von Software ist besonders das Fach *Industrielle Informationstechnik* (INIT) interessant. Die Jugendlichen lernen im Fach Angewandte Informatik in der ersten und zweiten Klasse bereits die Grundlagen der Informatik kennen, somit stellt INIT die informatische Vertiefung dar.

In der aktuell gültigen Fassung des Lehrplans für HTL sind verschiedene Ausbildungsschwerpunkte festgelegt. Einer davon ist z.B. der Ausbildungsschwerpunkt Informationstechnik, auf dessen Lehrplan im Folgenden genauer eingegangen werden soll. Wie jeder

Tabelle 4.1: Lehrplan einer Elektrotechnik HTL.

#	Gegenstand	Jahrgang					$\sum$
		I	II	III	IV	V	
Pflichtgegenstände							
1	Religion	2	2	2	2	2	10
2	Deutsch	2	2	2	2	2	10
3	Englisch	2	2	2	2	2	10
4	Geschichte und politische Bildung	-	-	-	2	2	4
5	Leibesübungen	2	2	2	1	1	8
6	Geographie und Wirtschaftskunde	2	2	-	-	-	4
7	Wirtschaft und Recht	-	-	-	2	2	5
8	Angewandte Mathematik	4	3	3	3	2	15
9	Angewandte Physik	2	2	2	-	-	6
10	Angewandte Chemie und Ökologie	2	2	-	-	-	4
11	Darstellende Geometrie	2	-	-	-	-	2
12	Angewandte Informatik	2	2	-	-	-	4
13	Grundlagen des Maschinenbaus	2	4	-	-	-	6
14	Allgemeine Elektrotechnik	3	5	2	-	-	10
15	Elektronik	2	2	2	2	2	4
16	Konstruktionsübungen	2	2	-	-	-	4
17	Laboratorium	-	-	3	-	-	3
18	Werkstättenlaboratorium	-	-	4	-	-	4
19	Werkstätte	8	7	3	-	-	18
$\sum$	Gesamtwochenstundenzahl	37	37	37	37	37	185
Schulautonomer Schwerpunkt: Informationstechnik							
1	Automatisierungstechnik	-	-	2	2	2	6
2	Elektronik	-	-	-	-	-	2
3	Elektrische Antriebe und Anlagen	-	-	2	2	2	6
4	Betriebssysteme und Netzwerke	-	-	2	2	3	7
5	*Industrielle Informationstechnik*	-	-	2	3	3	8
6	Projektengineering	-	-	2	3	2	7
7	Qualitäts- und Produktmanagement	-	-	-	2	-	2
8	Laboratorium	-	-	-	4	6	10
9	Werkstättenlaboratorium	-	-	-	3	3	6
$\sum$	Gesamtwochenstundenzahl	-	-	10	21	23	54

moderne Lehrplan ist auch jener für den Ausbildungsschwerpunkt Informationstechnik [80] sehr abstrakt gehalten und enthält eine kurze, allgemeine Formulierung des Lehrstoffs.

Die vorliegende Arbeit stellt in Kapitel 6 Unterrichtsblöcke und Beispiele mit einer Einordnung in das vom bm:ukk entwickelte Kompetenzmodell vor. Soll eine Einordnung in den bestenden Lehrplan erfolgen, so ist das Fach Industrielle Informationstechnik (INIT) – basierend auf den geforderten Lehrinhalt – eine passende Wahl. Im Lehrplan sind folgende Bildungs- und Lehraufgaben für das Unterrichtsfach Industrielle Informationstechnik (INIT) festgeschrieben:

Der Schüler/die Schülerin soll:

- unter Berücksichtigung praxisüblicher Vorschriften und Qualitätsstandards Informationstechnik in der Industrie einsetzen können;
- die Grundlagen der Programmierung, Visualisierung und den Austausch von Prozessdaten beherrschen;
- industrielle Bussysteme kennen, verstehen, planen und einsetzten;

Für den eigentlichen Lehrstoff sind folgende Inhalte vorgeschrieben:

III Jahrgang:

- Programmiersprachen der Informationstechnik
- Prozessrechnergrundlagen
- Mikrokontroller

IV Jahrgang:

- Industrielle Bussysteme
- Automatisierte Messsysteme
- Anwendung: Prozessrechentechnik, Echtzeitprogrammierung und hardwarenahe Programmierung

V Jahrgang:

- Leittechnik
- Prozessdatenbanken
- Anwendung von Standardsoftware auf Probleme der Prozessautomatisierung und -visualisierung unter Einbindung von Internet und Intranet

4.1.2 Kritik

Der aktuelle Lehrplan hat den Charakter eines Rahmenlehrplans und in Kombination mit den schulautomomen Schwerpunkten unterscheidet sich die tatsächliche Ausbildung teils erheblich. Dieses, durch die Autonomie, hervorgerufene Ungleichgewicht ist sowohl Fluch als auch Segen. Einerseits wird innovativen HTL's die Möglichkeit gegeben gezielt auf gewisse Inhalte einzugehen und sich mit dieser Spezialisierung am Arbeitsmarkt einen guten Namen zu machen. Andererseits fehlt Absolventen weniger innovativen und engagierten HTL's genau dieses Fachwissen und hinken somit ihren Kollegen hinterher.

4.2 Interview mit einem Vertreter der Lehre

4.2.1 Dipl.-Ing. Dr. Walter Führer – Zur Person

Das folgende Interview wurde mit Dipl.-Ing. Dr. Walter Führer am 23-02-2010 aufgezeichnet. Walter Führer ist selbst HTL Hollabrunn Absolvent und nach langjähriger Beschäftigung in der Industrie unterrichtet er nun bereits mehr als 10 Jahre an der HTL Hollabrunn. Seine Spezialgebiete sind Angewandte Informatik, Betriebssysteme und Netzwerke, Projektmanagement und die industrielle Informationstechnik welche er auch unterrichtet.

Dipl.-Ing. Dr. Walter Führer
Lehrender INIT, HTL Hollabrunn
e-mail: walter.fuehrer@htl-hl.ac.at, +43(2952)3361-214
web: **www.htl-hl.ac.at**
Dissertation'07: Auswirkungen verschiedener zeitlicher Ausgangssituationen auf das Testergebnis eines Multiple Choice Tests einschließlich einer didaktischen Bewertung und Reflexion der Konsequenzen für einen zeitgemäßen Mathematikunterricht

4.2.2 Interview

Allgemeines

$\mathcal{Q}.1$ Herr Führer, welche Fächer unterrichten Sie an der HTL Hollabrunn?
$\mathcal{A}.1$ *Ich unterrichte im Moment: Angewandte Informatik (AINF, 2te Klasse, höhere Abteilung + Fachschule) und im Ausbildungsschwerpunkt Informationstechnik: Industrielle Informationstechnik (INIT, 3te + 5te Klasse, höhere Abteilung), Laboratorium (LA1, 5te Klasse, höhere Abteilung), Projektengineering (PRE, 3te Klasse, höhere Abteilung) sowie in der 4ten Klasse Fachschule Betriebssysteme und Netzwerke (BUN). Insgesamt summiert sich mein Unterricht auf ca. 25 Stunden pro Woche.*

$\mathcal{Q}.2$ Wie lange unterrichten Sie bereits das Fach INIT?
$\mathcal{A}.2$ *Seit 5 Jahren.*

$\mathcal{Q}.3$ Haben Sie Industrieerfahrung, wenn ja wie sehr hilft ihnen diese beim Unterrichten?
$\mathcal{A}.3$ *Ja, ich war langjährig in der Softwareentwicklung und später als Projektleiter in einem großen österreichischen IT-Unternehmen tätig. Meine Aufgabenbereiche stellten neben dem Managen von Projekten auch die Organisation von firmeninternen Messen, das Erstellen von Abteilungsproduktfolder, sowie zahlreiche Dienstreisen dar. Ich glaube, durch meine Erfahrung kann ich in meinem Unterricht genau jenes weitergeben was ich mir selbst im Berufsleben mühsam erlernen musste und mir in meiner eigenen Ausbildung gefehlt hat. Weiters hilft mir die Berufserfahrung beim Vorbereiten der Schüler auf die reale Situation am Arbeitsmarkt, die nicht mit der Schulsituation vergleichbar ist. Generell stellt für mich die berufliche Praxiserfahrung eine solide Basis für einen guten und erfolgreichen Unterricht dar.*

$\mathcal{Q}.4$ Was tun Sie um am „Puls-der-Informatik“ zu bleiben, wie bilden Sie sich selbst weiter?
$\mathcal{A}.4$ *Ich entwickle schon seit längeren eine eigene Testsoftware und ein Content Management System (IWIS®) und bleibe somit ständig am Puls der Zeit bzgl. Datenbanken, Programmiersprachen, zugehörige Tools usw. Ich versuche auch regelmäßig Kontakte zu Firmen, Unternehmen, und ehemaligen Absolventen zu pflegen um somit neue Entwicklungen in der Industrie zu verfolgen. Durch Fachzeitschriften, Expertengesprächen und universitäre Weiterbildung (Dissertation im Bereich Assessment und Feedback in Schulen) versuche ich auch fachlich soweit wie möglich am Ball zu bleiben.*

Unterrichtsform

$\mathcal{Q}.5$ Welche Maßnahmen setzen Sie ein um den Praxisbezug Ihres Unterrichts herzustellen?
$\mathcal{A}.5$ *In technischer und allgemeinbildender Hinsicht führe ich mit den Schülern der Abschlussklasse ein Bewerbungsprojekt mit den Schwerpunkten Bewerbung, Assessment, Teambuilding und Bewerbungsgespräch durch. Ich finde es ist eine – neben den fachlichen – der wichtigsten Qualifikationen, dass sich Absolventen vernünftig bewerben und sich selbst gut verkaufen können. Um eine reale Projektsituation zu simulieren führe ich eine technische Ausschreibung in Zuge eines Firmenprojekts in der 4ten Klasse durch. Hier konkurrieren jeweils zwei Projektteams um einen einzelnen Auftrag. Bei Projektarbeiten versuche ich stets eine reale Situation zu konstruieren, d.h. es gibt Projektleiter und den Teammitgliedern werden verschiedene Verantwortungen übertragen. Weiters versuche ich reale Projektsituationen aus meinem Berufsalltag mit den Schülern nachzuspielen und meine persönlichen Best Practices weiterzugeben (bsp. Debugger, Entwurfsmodell, Realisierungsmöglichkeiten, Autogenerierung, Testzyklen, Abnahmeprüfungen, ...) Für den eigentlichen Unterricht versuche ich soweit als möglich aufeinander aufbauende Unterrichtssequenzen zu entwerfen. Beispielsweise wird eine Datenbankapplikation so entwickelt, dass sie zuerst nur über die Konsole zugänglich ist, später wird dann ein Web Interface entsprechend hinzugefügt und somit die Anwenderorientiertheit in den Vordergrund gestellt.*

Q.6 Wie glauben Sie kommt Ihr Unterricht bei den Schülern an?
A.6 *Ich glaube gut. Ich habe die Erfahrung gemacht, dass speziell jene Inhalte die auch von den Schülern privat 1:1 genutzt werden können besonders interessant sind. Beispielsweise Webpage-Erstellung mit Datenbanksysteme, Audio/Videotechnik ...*

Q.7 Welchen Führungscharakter legen Sie beim Unterrichten an den Tag (Demokratisch, Autoritär, Laissez-faire)?
A.7 *Ich adaptiere meinen Führungscharakter nach dem Unterrichtsthema und Form. Bei Frontalunterricht autoritär, bei Projektarbeiten eher demokratisch.*

Q.8 Welche Unterrichtsform bevorzugen Sie? (Frontalunterricht, Gruppenunterricht, Partnerunterricht, Projektunterricht, Portfoliounterricht)
A.8 *Für die Theorievermittlung setze ich Frontalunterricht ein. Gruppen- und Projektunterricht verwende ich für die begleitenden Übungen, wie z.B.: im Laboratorium. Portfoliounterricht verwende ich nicht.*

Q.9 Bitte gewichten Sie prozentual welche Unterrichtsformen Sie im INIT Unterricht einsetzen (Frontalunterricht, Gruppenunterricht, Partnerunterricht, Projektunterricht, Portfoliounterricht)
A.9 *Ich verwende zu 2/3 der Zeit Frontalunterricht (Theorie, erklären, besprechen) und zu 1/3 der Zeit Projektunterricht.*

Q.10 Wie wichtig sehen Sie den INIT Unterricht im Vergleich zu andern naturwissenschaftlichen Fächern im HTL Curriculum?
A.10 *Ich erachte ihn als sehr wichtig, da viele Unterrichtsinhalte einen Schwerpunkt der Ausbildung Informationstechnik darstellen (Grundlagenfach).*

Q.11 Verwenden Sie ein Skript oder ein begleitendes Buch für Ihren Unterricht?
A.11 *Es ist ein eigenes Skript vorhanden, das ich im Laufe der Jahre immer wieder überarbeite.*

Q.12 Welche Erfahrungen haben Sie mit Gruppenarbeiten & Projektarbeiten gemacht?
A.12 *Sowohl Gute als auch Schlechte. Gute Erfahrungen: Zusammenarbeit im Team vermitteln mit allen Konsequenzen (Eigenverantwortung). Schlechte Erfahrungen: Es ist schwierig Mitschwimmer zu vermeiden und die Gruppierung von „Guten“ zu vermeiden.*

Q.13 Haben Sie genug Möglichkeiten sich mit Fachkollegen über Ihren Unterricht auszutauschen?
A.13 *Ja.*

Q.14 Wie stellen Sie sicher, dass abgegebene Software vom Schüler selbst angefertigt ist und nicht bloß aus dem www kopiert ist?
A.14 *Durch Erfahrung ist vieles offensichtlich und ein Abgabegespräch gekoppelt mit einer Codedurchsicht deckt schnell auf ob es sich um eigene Arbeit handelt oder kopiert worden ist.*

Q.15 Wie handhaben Sie Plagiate?
A.15 *Entweder ich bewerte beide schlecht, oder versuche durch das Miteinbeziehen des*

Abgabedatums und Verständnisfragen zu klären wer von wem kopiert hat.

$\mathcal{Q}.16$ Für den INIT Unterricht, welche Arten der Leistungsbeurteilung verwenden Sie?
$\mathcal{A}.16$ *Ich verwende drei Arten der Leistungsfeststellung (i) Stundenwiederholung an der Tafel, (ii) selbst entwickelte, vollautomatische Multiple Choice Tests und (iii) schriftliche Wiederholungen (Programmentwicklung). Diese Leistungsfeststellungen stellen die Basis zur Leistungsbeurteilung dar.*

$\mathcal{Q}.17$ Welche Didaktik-Lektüre(n) haben Sie zuletzt gelesen?
$\mathcal{A}.17$ *In letzter Zeit habe ich unter anderem folgendes gelesen:*
Didaktik der Informatik, Hubwieser [41]
Fachdidaktische Studien, Karl Josef Fuchs [81]
Theorie des Unterrichtens und Prüfens, Werner Schwendenwein [82]

$\mathcal{Q}.18$ Inwiefern ist Ihr Unterricht fächerübergreifend?
$\mathcal{A}.18$ *Ich versuche folgende Fächer miteinander zu kombinieren: PRE + INIT in der 3ten Klasse sowie INIT + Labor in der 5ten Klasse. Weiters versuche ich in der 3ten und 5ten Klasse Bezüge zum Mathematikunterricht herzustellen. Beispiele dafür sind: INIT (3te Klasse): Bei der Informationstheorie nach Shannon werden verschiedene Arten des Logarithmus' verwendet; beim fehlererkennender Code wird die Polynomdivision angewandt. Beim Arithmetisches Codieren kommen halboffene Intervalle zum Einsatz. Beim Huffmancode werden Häufigkeiten, Auftrittswahrscheinlichkeiten und dergleichen angewandt. Beim Kryptographieverfahren RSA kommen große Primzahlen zum Einsatz, etc ... INIT(5te Klasse): Datenbanken: Mengenlehre, Funktionen, optimierbare Funktionen, ...*

$\mathcal{Q}.19$ Welche Tricks haben Sie um die Schüler zu motivieren?
$\mathcal{A}.19$ *Ich versuche zum Beispiel das Programmieren an sich zu motivieren. Ich mache den Schülern klar, dass sie ihre Programmierfähigkeiten auf eigene, persönliche Problemstellungen anwenden können, wie etwa einer meiner Schüler für ein Auslosungssystem eines Schützenvereinwettbewerbs.*

$\mathcal{Q}.20$ Sonstiges, das Sie gerne über Ihren Unterricht anmerken möchten?
$\mathcal{A}.20$ *Ich passe den Unterricht jedes Jahr dynamisch an und somit ist das Vorbereiten des Unterrichts sehr aufwendig. Somit kann ich aber sicherstellen einen zeitgemäßen Unterricht zu halten.*

Schwerpunkte des Unterrichts - Fachinhalte

$\mathcal{Q}.21$ Welches Wissen hätten Sie gerne das Ihre Schüler aus Ihrem Unterricht „mitnehmen" und auch in 5 Jahren noch abrufen können?
$\mathcal{A}.21$ *Wichtig ist mir, dass die Absolventen gut auf den Bewerbungsprozess vorbereitet sind. Weiters wünsche ich mir, dass Sie noch lange von den Projektmanagmentfähigkeiten (Teamfähigkeit, Problemlösung und Kommunikation) zehren können.*

Q.22 Welche Lehrinhalte/Themen sind für Sie besonders schwierig zu vermitteln?
A.22 *Lehrinhalte bei denen kein unmittelbarer Praxisbezug hergestellt werden kann sind besonders schwierig zu vermitteln.*

Q.23 Ist Software Test ein Teil Ihrer Lehrinhalte?
A.23 *Teilweise versuche ich den Schülern auch das Softwaretesten beizubringen, jedoch ist in Schulprojekten dafür meist zu wenig Zeit. (Im Vgl. zu meinem Berufsleben wo es drei Testphasen gab und dies den größten Teil der SW Entwicklung einnahm . . .)*

Q.24 Ist Software Verifikation ein Teil Ihrer Lehrinhalte?
A.24 *Nein, nur eine teilweise Validierung.*

Q.25 Wenn (23) oder (24) nein, warum?
A.25 *Zeitmangel, evtl. auch schwierig zu vermitteln aufgrund mangelnder Motivation der Schüler*

Q.26 Welche Softwareentwicklungsprozesse vermitteln Sie?
A.26 *Ich versuche den Entwicklungsprozess im größeren Kontext eines Projekts zu vermitteln, für die SW Entwicklung nehme ich meist das Wasserfallmodel heran.*

Q.27 Welche Programmiersprachen setzen Sie ein? (Argumentieren Sie bitte kurz dafür)
A.27 *Für das Fach INIT: PHP und MySQL und für das Fach AINF: C jedoch ohne Objektorientierung*

Q.28 Im Gegensatz zu anderen Fächern, finden Sie, dass Ihr Unterricht von Ihren Kollegen bzgl. der Fachinhalte respektiert wird?
A.28 *Meistens, da die Kollegen einen großen Praxisbezug darin sehen. Sie finden es vielleicht manchmal mit zu großem Aufwand verbunden.*

Q.29 Sonstiges, das Sie gerne über Ihre Fachinhalte anmerken möchten?
A.29 *Der Praxisbezug ist mir sehr wichtig für meinen Unterricht.*

Sonstiges

Q.30 Wenn Sie Bundesminister für Bildung wären, was würden Sie an Ihrem ersten Amtstag am gegenwärtigen HTL Curriculum für Informationstechnik ändern wollen?
A.30 *Es sollten mehr finanzielle Mittel für Bildung aufgebracht werden (das will natürlich jeder). Es gibt bereits Studien, die belegen, dass andere EU Staaten (auch neue, wie z.B. Polen) viel mehr in die Bildung investieren und somit Österreich in absehbarer Zeit „überholen". Die Ausbildung an berufsbildenden höheren Schulen muss unbedingt langfristig auf dem gleichen hohen Niveau gehalten werden. Sparmaßnahmen sind hier fehl am Platz. Lehrende die sich bereit erklären an Schulveranstaltungen außerhalb der Schule, wie Skikurse, Sommersportwochen, motiviert teilzunehmen sollten aufgrund der „erweiterten" Aufsichtspflicht und Verantwortung besser entlohnt werden. Momentan werden diese Zusatzstunden aus Sparmaßnahmen sogar den bestehenden*

Überstunden abgezogen.

$\mathcal{Q}.31$ Haben Sie schon von dem neuen Kompetenzenmodell des bm:ukk gehört, wenn ja was halten Sie davon?
$\mathcal{A}.31$ *Ja, habe ich. Prinzipiell finde ich den Ansatz interessant auch im späteren Hinblick auf die Vergleichbarkeit der Qualifikationen auf EU Ebene. Aber ich sehe die Schulautonomie dabei sehr gefährdet. Der Lehrende vermittelt immer genau jene Inhalte auf höchster Ebene auf die er sich persönlich spezialisiert hat (bsp. durch Berufserfahrung). Soll er nun aber jetzt andere, zentral vorgegebene Inhalte vermitteln und Beispiele verwenden, so wird seine eigene Motivation sinken und somit auch die Gesamtqualität des Unterrichts tendenziell abnehmen. Es ist wichtig, dass auch dem Lehrenden der Sinn der zentral vorgegebenen Komptenzen klar gemacht werden kann und sie somit mit auf das Bildungsstandards-Boot geholt werden.*

$\mathcal{Q}.32$ Wie setzen Sie sich mit dem Feedback der Schüler auseinander? Inwiefern sind Sie bereit Ihren Unterricht grundlegend zu ändern?
$\mathcal{A}.32$ *Nach jeder Leistungsfeststellung erfolgt ein vollautomatisiertes, elektronisches Feedback samt Auswertung. Im Anschluss werden daraus besondere Items mit der gesamten Klasse diskutiert um nähere Informationen zu erhalten. Aus dem Feedback leite ich Anregungen ab und versuche dadurch einen Qualitätssicherungskreislauf für meinen Unterricht und Leistungsfeststellungen zu erhalten.*

$\mathcal{Q}.33$ Finden Sie, dass die Schüler genügend Motivation mitbringen?
$\mathcal{A}.33$ *Teilweise ja, aber stark vom Individuum abhängig.*

$\mathcal{Q}.34$ Viele Lehrkräfte beklagen „früher war alles besser“ wie sehen Sie das?
$\mathcal{A}.34$ *Meiner Meinung nach führen geburtenschwächere Jahrgänge unvermeidlich zur Aufnahme von Schülern die früher nicht den Anforderungen einer HTL erfüllt hätten. Dieses Phänomen ist aber nicht nur an HTL's zu beobachten.*

$\mathcal{Q}.35$ Sonstiges, das Sie gerne anmerken möchten?
$\mathcal{A}.35$ *Ich befürworte die Evaluation des Unterrichts durch Feedbacks der Schüler. Nach jedem Test führe ich das selbst durch zur Selbstevaluation. Wenn die Evaluation aber nun für alle Lehrenden eingeführt werden soll, ist jedoch anzudenken den Lehrern eine Möglichkeit zu geben auch die Schüler zu beurteilen, bzw. Feedbacks abgeben zu können.*

4.2.3 Interpretation und Schlussfolgerungen

Das Interview mit Dipl.-Ing. Dr. Walter Führer gibt einen interessanten Einblick in die Unterrichtspraxis. Daraus lassen sich für diese Arbeit mehrere Schlussfolgerungen ziehen:

- Softwareverifikation und Softwaretesten kommen im momentanen Unterricht nicht vor. Ein Grund dafür ist offensichtlich Zeitmangel (vgl. Q/A23, Q/A24, Q/A25).

- Moderner Informatikunterricht in einer höheren technischen Lehranstalten ist stark projektorientiert. Dies ist auch durch den beruflichen Background des Lehrenden geprägt (vgl. Q/A5, Q/A9, Q/A21).
- Bildungsstandards und Kompetenzmodelle sind Begriffe die den Lehrenden nicht fremd sind. Eine gewisse Skepsis bleibt, besonders dass bei der tatsächlichen Implementierung die langjährig entwickelte Schulautomomie verloren geht (vgl. Q/A31).
- Schüler sind besonders für Lehrstoff zu begeistern den sie auch selbst für eigene, praktische Projekte unmittelbar weiterverwenden können (vgl. Q/A6, Q/A19).

5 Bildungsstandards und Kompetenzmodelle

> The nice thing about standards is that there are so many of them to choose from.
>
> *(Andrew S. Tanenbaum)*

Die Europäische Union versucht erworbene Qualifikationen in ihren einzelnen Mitgliederstaaten durch einen gemeinsamen Rahmen vergleichbar zu machen. Dazu wurde der Europäische Qualifikationsrahmen für lebenslanges Lernen (EQR) [83] erarbeitet, der es ermöglicht erworbene Kenntnisse, Fertigkeiten und Kompetenzen in ein 8 stufiges Modell einzuordnen. Dieses sehr abstrakte Modell definiert eine Reihe von Deskriptoren, die für die Erlangung der diesem Niveau entsprechenden Qualifikationen in allen Qualifikationssystemen erforderlich ist. So ist beispielsweise Stufe 6 mit einem erfolgreichen abgeschlossenen ersten Studienzyklus (Bachelor), Stufe 7 mit einem erfolgreich absolvierten zweiten Studienzyklus (Master) und Stufe 8 mit einem erfolgreich absolvierten dritten Studienzyklus (PhD) zu erreichen.

Mit dieser Initiative versucht die Europäische Union einen Übersetzungsraster zwischen den individuellen Qualifikationssystemen der Mitgliederstaaten zu schaffen. Dieser Raster ermöglicht es Arbeitgeber, EU-Bürger und Einrichtungen EU-weite Qualifikationen vergleichbar und verständlich zu machen. Im Moment wird der EQR als unverbindliche Empfehlung für die Mitgliederstaaten geführt.

Das Projekt „Bildungsstandards in der Berufsbildung" [26] des bm:ukk hat es sich zum Ziel gesetzt diese europäischen Pläne nach transparenten Darstellungsformen von Lernergebnissen sowie nach Systemvergleichbarkeit zu unterstützen. Die Unterstützung erfolgt konkret durch die Erarbeitung von sog. Bildungsstandards für die Berufsbildung. Diese sollen zur Erhöhung der Transparenz im nationalen Bildungssystem beitragen und somit eine bessere Vergleichbarkeit von Bildungsabschlüssen gewährleisten.

Zusammenfassend orientieren sich Bildungsstandards an nationalen oder internationalen Bildungszielen, denen schulisches Lernen folgen soll, und setzen diese in konkrete Anforderungen um.

Aus einer Veröffentlichung des bm:ukk [26] geht eindeutig hervor, dass ein „Teaching to the Test" nicht beabsichtigt ist und dies unter allen Umständen vermieden werden muss.

5.1 Taxonomie bei bildungstechnologischen Standards

Der kognitive[1] Teil des Kompetenzmodells des bm:ukk baisert auf den wissenschaftlichen und theoretischen Arbeiten von Benjamin Bloom's „Taxonomie von Lernzielen im kognitiven Bereich" [84]. Unter Taxonomie versteht man ein Modell das versucht Begriffe eines Themengebietes einerseits zu definieren und andererseits untereinander in Beziehung zu setzen. Es ist daher ein Modell um eine Thematik präzise zu beschreiben und zu repräsentieren. Bloom wendete dieses Konzept auf die Lerntheorie an und definiert somit ein Klassifikationschema für Lernziele. Der kognitive Teil beinhaltet all jene Ziele die mit Wissen, Denken und dem Problemlösen zu tun haben. Die jeweiligen Kategorie sind Begriffe, die die Art des Verhaltens beschreiben, das man von den Studierenden erwarten kann. Bloom ordnet Lernziele für den kognitiven Bereich in sechs Kategorien mit aufsteigender Komplexität:

1. Wissen ↪ Informationen möglichst wortgetreu erinnern und wiedergeben können. Kenntnisse konkreter Einzelheiten wie Definitionen, Regeln, Daten, Fakten, Theorien, Kriterien, Abläufe, ...

2. Verstehen ↪ Sachverhalte und Informationen sinngemäß umformen können. In eigenen Worten wiedergeben, zusammenfassen, Beispiele finden und graphisch darstellen.

3. Anwenden ↪ Abstraktionen (Regeln, Methoden, etc) in konkreten Situationen anwenden können. Problemlösender Wissenstransfer, bereits Gelerntes kann in neuen Situationen abgerufen werden.

4. Analyse ↪ Ideen, Problemstellungen in ihre Einzelkomponenten zerlegen und vergleichen können, Unterschiede herausarbeiten und Folgerungen ableiten. Unterscheiden zwischen Fakten und Interpretationen.

5. Synthese ↪ Einzelne Elemente zu einer Ganzheit formen. Neue Schemata entwerfen und Hypothesen begründen.

6. Beurteilen ↪ Ein bewertendes Urteil abgeben können. Alternativen gegeneinander abwägen, auswählen, Entschlüsse fassen und begründen. Bewusst Wissen transferieren.

Bloom stellt somit eine Einteilung bereit um beliebige Lernziele nach Schwierigkeits- und Komplexitätsgraden einzuordnen. Diese Einteilung ist in einer etwas abgewandten Form in die Handlungsdimension des Kompetenzmodells eingeflossen (siehe Abschnitt 5.2.2).

[1] Wahrnehmung, Lernen, Erinnern und Denken, menschliche Erkenntnis- und Informationsverarbeitung

5.1.1 Der Komptenzbegriff

Eine umfassende Definition des Begriffs Kompetenz gib Weinert in [85]:

> Unter Kompetenzen versteht man die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können.

5.2 Das Kompetenzmodell

Das Kompetenzmodell ist fachspezifisch und wesentlicher Teil eines Bildungsstandards. Es besteht aus einer Handlungs- und einer Inhaltsdimension. Abbildung 5.1 stellt das Kompetenzmodell für die HTL Fachrichtung Elektrotechnik dar. Die Schnittpunkte dieses Koordinatensystems bilden maximal einen Deskriptor der zu genau einem Unterrichtsbeispiel korreliert. Über diese Deskriptoren soll sichergestellt werden, dass alle Absolventen eines Schultyps gemeinsame Kernkomptenzen erreichen. Das Kompetenzmodell liefert einen Rahmen um die erwarteten Lernergebnisse zu beschreiben und sie transparent darzustellen.

5.2.1 Inhaltsdimension

Die Inhaltsdimension des Kompetenzmodells bündelt Qualifikationen die aus beruflicher Sicht besonders relevant für künftige Elektrotechnik Absolventen sind. Sie umfasst lt. [27] folgende Kernthemen:

1. Energietechnik
2. Automatisierungstechnik
3. Antriebstechnik
4. Industrielle Elektronik
5. Fachbezogene Informationstechnik

Die spezifischen Lehrinhalte sind in [27] nachzulesen und werden in dieser Arbeit nicht weiter behandelt.

5.2.2 Handlungsdimension

Die Handlungsdimension spezifiziert kognitive, persönliche und soziale Fähig- und Fertigkeiten, die sich aus dem allgemeinen Bildungsziel und dem spezifischen Ausbildungsprofil

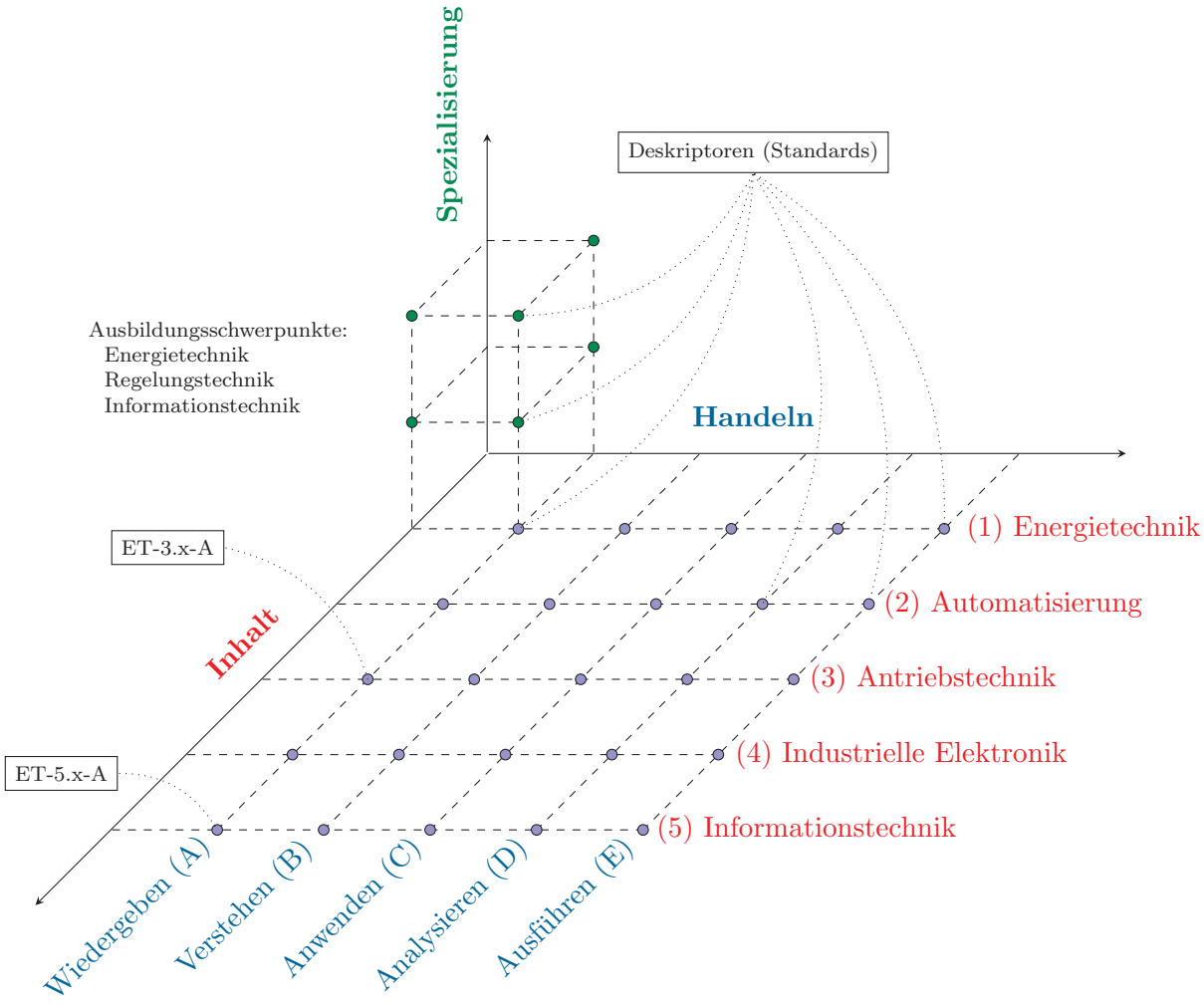


Abbildung 5.1: Das Kompetenzmodell für die HTL Fachrichtung Elektrotechnik (Eigendarstellung nach [28, 26]).

hinsichtlich des späteren Berufslebens ableiten lassen. Die Anordnung ist dabei nicht beliebig. Im Folgenden werden die einzelnen Komponenten der Handlungsdimension basierend auf dem Bildungsstandardentwurf für die Elektrotechnik HTL (wie in [27, 28, 26]) in einer kurzen Zusammenfassung dargestellt.

Wiedergeben

Beschreibt die Kompetenz, elektrotechnische Grundkenntnisse wiederzugeben und sich der geeigneten Fachterminologie zu bedienen. Zugeordnet werden hier die Handlungen „kennen, reproduzieren, angeben und beschreiben". Das eigentliche Wiedergeben von Fachwissen stellt die Kernfähigkeit dar. Dies kann passieren durch Wiedergeben eines verbal ausformulierter Problemstellung im Hinblick auf einen geeigneten Lösungsansatz oder durch das Wiedergeben und Zusammenfassen von Wissensinhalten.

Verstehen

Beschreibt die Kompetenz, Zusammenhänge von elektrotechnischen Grundkenntnissen zu erkennen. Zugeordnet werden hier: Handlungen systematisch ordnen, aus Beobachtung erfassen, charakterisieren, erklären, schlussfolgern, Fachsprache verwenden, usw. Gerade Gelerntes wird auf einen bestimmten Sachverhalt übertragen oder damit verglichen.

Anwenden

Beschreibt die Kompetenz, elektrotechnische Sachverhalte zu bearbeiten, Informationen auszuwerten und Aufgabenstellungen mit Hilfe des elektrotechnischen Wissens zu lösen und in geeigneter Symbolik und Methodik darzustellen. Zugeordnet werden hier die Handlungen messen, ausrechnen, auswerten, durchführen, umsetzen, testen, auswählen, präsentieren. Im Vordergrund steht die Nutzung sowie die Anwendung eines gelernten Verfahrens, die Abarbeitung vorgegebener Schritte und das Erarbeiten einer Lösung aus einer gegeben Struktur.

Analysieren (Interpretieren)

Beschreibt die Kompetenz, elektrotechnische Sachverhalte zu erkennen, zu interpretieren und zu bewerten, sowie Vergleiche anzustellen. Zugeordnet werden hier die Handlungen interpretieren, ableiten, Modelle abgrenzen, Prinzipien übertragen, evaluieren, modellhaft darstellen und auswerten, usw. Besonders wichtig ist die Fähigkeit etwas Gelerntes neu zu strukturieren, eigene Kriterien und Gesichtspunkte zu entwickeln und zu übertragen.

Ausführen (Entwickeln)

Beschreibt die Kompetenz, berufsspezifische und praxisnahe Aufgabenstellungen der Elektrotechnik mit Hilfe geeigneter Methoden und Verfahren zu lösen sowie Entwürfe, Berechnungen und Auslegungen anzufertigen. Zugeordnet werden die Handlungen konstruieren, Entwürfe konzipieren, planen, Modelle entwerfen, Prognosen stellen, Lösungskonzepte erarbeiten.

5.2.3 Deskriptoren

Deskriptoren konkretisieren die Bildung- und Lehraufgaben, in ihrer Summe erfassen sie sämtliche Fähigkeiten eines Absolventen. In [27, 28] werden diese nach folgendem Schemata mit Hilfe fünf Tags definiert:

<**Fachrichtung**><**Schwerpunkt**>-<**Inhalt**>.<**Nummerierung**>-<**Handeln**>

Die Tags <Inhalt>:= $\{1, 2, 3, 4, 5\}$ und <Handeln>:= $\{A, B, C, D, E\}$ korrelieren zu den Achsenbeschriftungen des Kompetenzmodells (vgl. Abbildung 5.1). So steht Inhalt 3 für Antriebstechnik und Handeln C für Anwenden. Die beiden Tags können beliebig kombiniert werden und bilden somit ein kartesisches Produkt <Inhalt> × <Handeln>. Der Tag <Schwerpunkt>:= $\{ET, ETE, ETR, ET\}$ entspricht den angebotenen Ausbildungsschwerpunkten die eine HTL für Elektrotechnik anbietet. Die einzelnen Inhalte wurden bereits in Abschnitt 4.1 behandelt.

Der Tag <Nummerierung>:= $\{1, 2, \ldots, 198, 199\}$ nummeriert die einzelnen Deskriptoren für einen Knoten im Kompetenzmodell. Wendet man nun diese Notation auf das Kompetenzmodell in Abbildung 5.1 an, so lassen sich folgende Beispieldeskriptoren [27] formulieren:

`ETR-5.102-A` Ich kann die Organisation von Betriebssystemen wiedergeben.
`ETE-1.105-B` Ich kann Aufbau und Einsatz der HGÜ erklären.
`ETI-2.105-C` Ich kann Automatisierungssysteme visualisieren.
`ETI-5.107-D` Ich kann Bussysteme analysieren.
`ETE-1.118-E` Ich kann Blitzschutzanlagen planen.
`ET-3.2-A` Ich kenne die Bauelemente und grundlegenden Schaltungen von leistungselektronischen Einrichtungen.
`ET-4.7-B` Ich kann die Funktionsweise von Bauelementen und deren Kennwerte erklären.
`ET-4.10-C` Ich kann elektronische Schaltungen realisieren, überprüfen und warten.
`ET-5.17-D` Ich kann Fehler in informationstechnischen Einrichtungen mit geeigneten Werkzeugen analysieren.
`ET-1.21-E` Ich kann Kabel und Leitungen sowie deren Schutz dimensionieren.

5.2.4 Adaption der Deskriptoren

Der bestehende Entwurf [27] des Kompetenzmodells für Elektrotechnik HTLs sieht keine Kompetenzen im Bereich der Software und System Verifikation vor.

Motiviert durch die Zustandserfassung der Anforderung der Industrie an HTL Absolventen in Kapitel 3 wird das Kompetenzmodell um folgende Deskriptoren für die Kernthemen formale Software und System Verifikation erweitert:

`ETI-5.114-A` Ich kenne Methoden zur Modellierung und formalen Verifikation von Software und IKT Systemen.

`ETI-5.115-B` Ich kann Modellierung und Abstraktion von Software und IKT Systemen erklären.

`ETI-5.116-C` Ich kann Software und IKT Systeme modellieren.

`ETI-5.117-D` Ich kann Software und IKT Systeme mit (formalen) Verifikationsmethoden analysieren.

`ETI-5.118-E` Ich kann Software und IKT Systeme mit Hilfe von Modellierung, Spezifikation und formaler Verifikation planen.

5.2.5 Implementation

Auf der Website der Bildungsstandards [86] heißt es:

> Gehen wir gemeinsam einen Schritt in Richtung kompetenzorientierten Unterricht.

Es bleibt abzuwarten, ob mit dem Kompetenzmodell auch tatsächlich dieser Schritt in Richtung einer ganzheitlichen Veränderung des Unterrichts einhergeht, oder es eventuell „nur“ für eine Feststellung und Einordnung der berufsbildenden Schulen in den Europäischen Qualifikationsrahmen verwendet wird.

6 Unterrichtsplanung

> The best way to teach somebody something is to have them think they're learning something else.
>
> *(Randy Pausch)*

Im folgenden Kapitel werden die Ergebnisse der Unterrichtsplanung dargestellt. Es werden insgesamt sechs zusammenhängende Lehrblöcke/Beispiele vorgestellt. Jeder dieser Lehrblöcke beinhaltet eine Einordnung in das Kompetenzmodell. Ein weiteres Beispiel stellt eine inhaltliche Zusammenfassung der vorhergehenden Inhalte dar.

Die inhaltliche Ausrichtung ist vor allem der Bereich Modellierung, kombiniert mit formaler Spezifikation und automatischen Verifikationsmethoden, sowie der praktischen Fehlersuche in Systeme und Software. Die Beispiele wurden durch die qualitativen Interviews in den Kapitel 3 und 4 motiviert. Eine Argumentation für die Signifikanz von Modellierung im Informatikunterricht unter dem Aspekt von Bildungsstandards findet sich auch in [87].

6.1 Block I – Do it yourself Kaffeemaschine

6.1.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Do it yourself Kaffeemaschine
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.116-C` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, Kripke Struktur, Spezifikation, formale Notation
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 100 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.1.2 Aufgabenstellung

Sie arbeiten in einem von der globalen Wirtschaftskrise stark betroffenen Unternehmen. Als Sie Ihren Vorgesetzten auf die schon tagelang kaputte Kaffeemaschine in der Firmenküche aufmerksam machen und damit auch den von Ihrem Chef beklagten Motivationsmangel begründen, erteilt Ihnen dieser erbost eine klare Absage. Für den neuen vollautomatischen Kaffeeautomaten ist kein Geld übrig.

Das wollen Sie aber so nicht auf sich sitzen lassen. Als HTL Ingenieur haben Sie früher auch schon immer gerne gebastelt, gelötet und programmiert. Um den Firmenfrieden zu retten, beginnen Sie den Kaffeeautomaten (in Ihrer Freizeit natürlich) höchstpersönlich auf eigene Faust von Null an zu entwickeln.

Ihre Arbeitskollegen sind von der ersten Sekunde an hellauf begeistert und unterstützen Sie. In einer ersten Gesprächsrunde kristallisieren sich folgende Anforderungen an die neue self-made Kaffeemaschine heraus, die Sie kurz und bündig niederschreiben:

- Nach dem Münzeinwurf kann der Kunde seinen gewünschten Kaffee wählen.
- Ein Kaffee wird nur dann gebrüht, wenn eine korrekte Auswahl vorausgeht.
- Der Vorgang kann zu jeder Zeit abgebrochen werden.

Aus diesen Anforderungen leiten Sie für die erste Modellierungsphase folgende fünf Events ab:

coin	...	wenn Münze eingeworfen ist.
select	...	wenn Kaffee gewählt wurde.
brew	...	wenn Kaffee gebrüht wird.
abort	...	wenn Abbruch gedrückt wird.
fin	...	wenn Kaffee ausgegeben wurde.

Zu lösende Aufgaben:

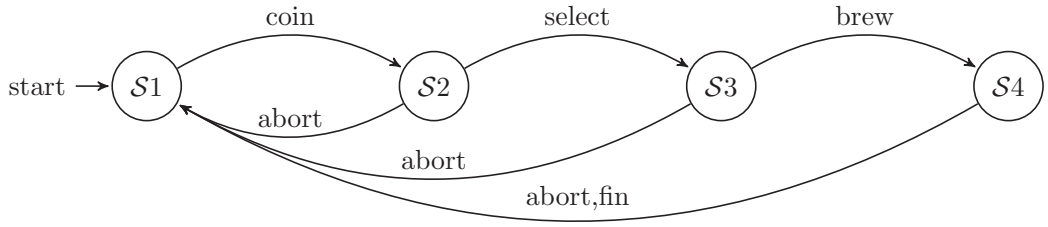
1. Modellieren Sie die Kaffeemaschine, verwenden Sie dazu die Semantik eines Zustandsübergangsdiagramms.

2. Erstellen Sie aus Ihrem Modell einen Abarbeitungsbaum für die ersten 5 Ebenen. Beschriften Sie die einzelnen Knoten mit den Zustandsnamen.

3. Schreiben Sie mindestens zwei mögliche Ablaufsequenzen des Modells an.

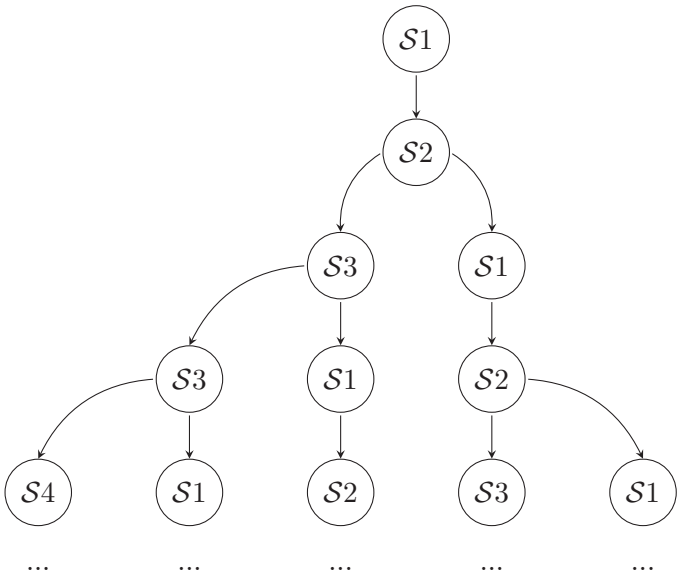
Lösungsvorschläge:

1. Zustandsübergangsdiagramm (Systemmodell)



   Dieses Modell stellt eine der einfachsten Lösungen für die Aufgabenstellung dar. Kompliziertere Modelle könnten natürlich verschiedene Kaffeesorten usw. berücksichtigen. Auch die Rückgabe des Wechselgeldes könnte man detaillierter modellieren. Die tatsächliche Komplexität hängt von der Wahl des Abstraktionsgrades ab.

2. Abarbeitungsbaum



3. Mögliche Ablaufsequenzen:
   (i) $\mathcal{S}1 \rightarrow \mathcal{S}2 \rightarrow \mathcal{S}1 \rightarrow \mathcal{S}2 \rightarrow \mathcal{S}1 \ldots$
   (ii) $\mathcal{S}1 \rightarrow \mathcal{S}2 \rightarrow \mathcal{S}3 \rightarrow \mathcal{S}1 \rightarrow \mathcal{S}2 \ldots$
   (iii) $\mathcal{S}1 \rightarrow \mathcal{S}2 \rightarrow \mathcal{S}3 \rightarrow \mathcal{S}3 \rightarrow \mathcal{S}1 \ldots$
   (iv) $\mathcal{S}1 \rightarrow \mathcal{S}2 \rightarrow \mathcal{S}3 \rightarrow \mathcal{S}3 \rightarrow \mathcal{S}4 \ldots$

Anmerkungen, Erläuterungen und Ergänzungen zum Lerninhalt:

Das hier angeführte Übungsbeispiel gibt eine nachvollziehbare Einführung in die Modellierung und Abstraktion von Systeme. Diese Fähigkeiten sind allesamt bedeutende Kernkonzepte der formalen, automatisierten Verifikation von Hardware und Software Systeme. Die Abstraktion ist die Ableitung des Wesentlichen/Charakteristischen/Gesetzmäßigen aus einer Menge von realen Beobachtungen. Sie ist ein zentrales Konzept für das Erstellen und Verstehen von Modellen im Allgemeinen.

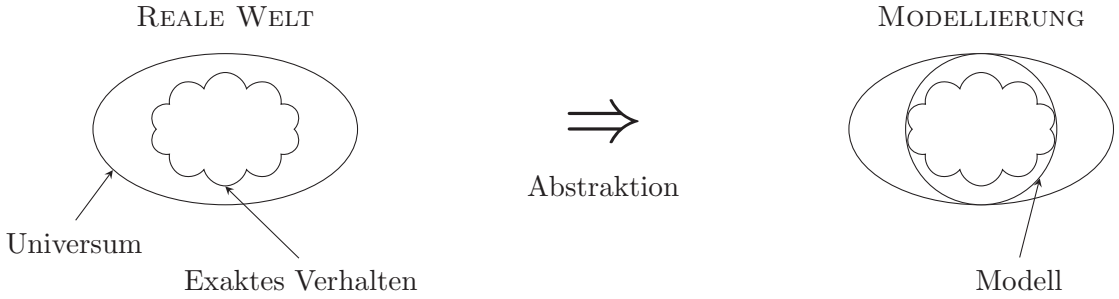


Abbildung 6.1: Abstraktion in der Modellierung. Frei nach [88].

Große Modelle sind ohne einen klar strukturierten Aufbau nicht verstehbar. Dies muss auch den Lernenden vermittelt werden. Die Abstraktion ist daher notwendig zur Beherrschung der Komplexität des Kaffeeautomaten im Ganzen. Würde man einen modernen Kaffeeautomaten mit allen Details modellieren, so würde dies sehr aufwendig und komplex werden. Durch die Vorgabe, dass nur fünf Events *coin, select, brew, abort, fin* verwendet werden, wird die Komplexität durch den Lehrenden während der Aufgabenstellung eingeschränkt.

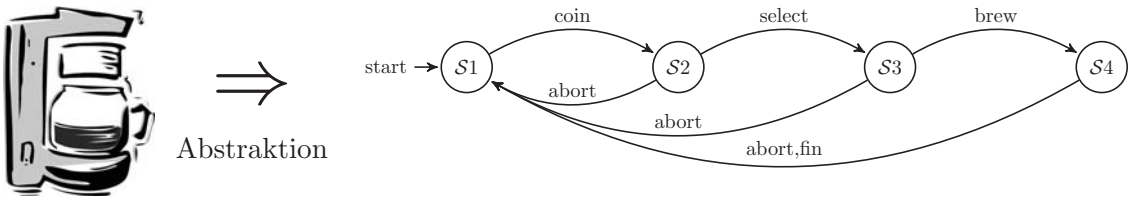


Abbildung 6.2: Abstraktion in der Modellierung.

Weiterführung des Beispiels, Begabtenförderung:

Eine mögliche Weiterführung des Beispiels ist die formale Notation des Systemmodells. Dazu kann zum Beispiel eine 4-Tupel Kripke Struktur Notation verwendet werden:

$$\mathcal{M} = (\mathcal{S}, s_0, \mathcal{R}, \mathcal{L})$$

$\mathcal{S}$: *endliches Set an States*
s_0: *Ausgangszustand* $s_0 \subseteq \mathcal{S}$

$$\mathcal{R}\text{: } \textit{Übergangsrelation } \mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$$
$$\mathcal{L}\text{: } \textit{Interpretationsfunktion } \mathcal{L} :\rightarrow 2^{\mathcal{AP}}$$

Die *Übergangsrelation* $\mathcal{R}$ spezifiziert für jeden Zustand die möglichen Nachfolgezustände, konkret: für jeden Zustand $s \subseteq \mathcal{S}$ gibt es einen Nachfolgezustand $s' \subseteq \mathcal{S}$. Die *Interpretationsfunktion* $\mathcal{L}$ markiert jeden Zustand mit einem Set von $\mathcal{AP}$ die in diesem Zustand zu true evaluiert werden. Ein Pfad π in der Kripke Struktur $\mathcal{M}$ von einem Zustand s ist ein Ablauf von Zuständen $\pi = s_0 s_1 s_2 ...$, gegeben, dass $s_0 = s$ und $\mathcal{R}(s_i, s_{i+1})$ für alle $i \geq 0$ gilt [23].

Zu lösende Zusatzaufgaben:

1. Beschriften Sie die einzelnen Zustände mit den aktuellen Werten der Bool'schen Ausdrücke *payed, selection* und *brewing*. Diese haben die folgende Bedeutung:

$$\textit{payed} = \begin{cases} \texttt{true} & \text{wenn Münze eingeworfen ist,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$\textit{selection} = \begin{cases} \texttt{true} & \text{wenn Kaffee gewählt wurde,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

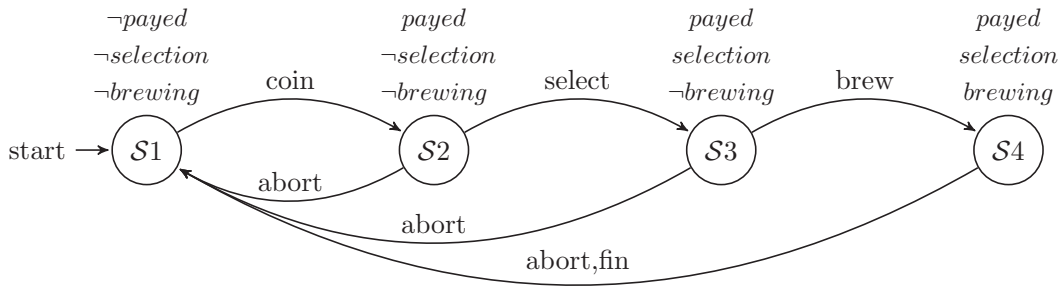
$$\textit{brewing} = \begin{cases} \texttt{true} & \text{wenn Kaffee gebrüht wird,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

2. Schreiben Sie Ihr gefundenes Systemmodell formal nach dem allgemeinen Modell der Kripke Struktur an.

Lösungsvorschläge:

1. Systemmodell

   Schreibt man die aktuellen Werte der Bool'schen Variablen *payed, selection* und *brewing* an, so wird das Systemmodell zu:



2. Formales Systemmodell

$$\mathcal{M}_{Kaffee} = (\mathcal{S}, s_0, \mathcal{R}, \mathcal{L})$$

$$\mathcal{S} := \{\mathcal{S}1, \mathcal{S}2, \mathcal{S}3, \mathcal{S}4\}$$
$$s_0 := \{\mathcal{S}1\}$$
$$\mathcal{R} := \{(\mathcal{S}1, \mathcal{S}2), (\mathcal{S}2, \mathcal{S}1), (\mathcal{S}2, \mathcal{S}3), (\mathcal{S}3, \mathcal{S}1), (\mathcal{S}3, \mathcal{S}4), (\mathcal{S}4, \mathcal{S}1\}$$
$$\mathcal{L}(s_1) := \{\neg payed, \neg brewing, \neg selection\}$$
$$\mathcal{L}(s_2) := \{payed, \neg brewing, \neg selection\}$$
$$\mathcal{L}(s_3) := \{payed, \neg brewing, selection\}$$
$$\mathcal{L}(s_4) := \{payed, brewing, selection\}$$

$\pi_0 = s_1 s_2 s_1 s_2 s_1 ...$ $\pi_1 = s_1 s_2 s_3 s_1 s_2 ...$ $\pi_2 = s_1 s_2 s_3 s_4 s_1 ...$

Dieses formale Systemmodell stellt nun die Grundlage für eine weitere Verifikation dar. Die Umwandlung vom Systemmodell in ein formales Modell erfolgt bei modernen Tools vollautomatisiert, jedoch verstärkt deren manuelle Durchführung das grundlegende Verständnis und hilft die Zusammenhänge besser zu verstehen.

6.2 Block II – Kaffeesatz und Leibniz's Traum

6.2.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Kaffeesatz und Leibniz's Traum
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.117-D` `ETI-5.118-E` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Bool'sche Aussagenlogik, Spezifikationsproblem
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 60 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.2.2 Aufgabenstellung

Sie sind vom Eifer gepackt und nach der erfolgreichen Modellierungsphase der Abteilungskaffeemaschine machen Sie sich Gedanken darüber, welche Eigenschaften die Kaffeemaschine denn erfüllen soll. Ihre vom Kaffeemangel getriebenen Arbeitskollegen erklären sich bereit Ihnen Ihre tatkräftige Unterstützung anzubieten. Sie arbeiten in einem internationalen Team mit Experten aus verschiedenen Kulturkreisen. Verschiedene Kulturkreise, verschiedene Sprachen. Etwa 80% Ihrer Kollegen sprechen vernünftiges Englisch, jedoch dürfen Sie sich glücklich schätzen auch einen brasilianischen Hobby Kaffeeexperten aus *Minas Gerais* zu Ihrem Team zu zählen, der zwar eine unumstrittene Koryphäe auf dem Gebiet der schwarzen Bohnen ist, leider spricht er nur Portugiesisch, dass Sie wiederum nicht beherrschen.

Sie, als selbst ernannter Kaffeemaschinen Projektleiter, entscheiden sich nun dazu nicht auf die Expertise Ihres brasilianischen Arbeitskollegens zu verzichten und überlegen nun wie Sie eine gemeinsame Spezifikation schreiben können, die für alle verständlich ist. Doch wie machen Sie das? Das Ergebnis des Team-Brainstormings zeigt Abbildung 6.3.

Einer Ihrer Teamkollegen erinnert sich daran einmal von der von Leibniz[1] angedachten Universalsprache (lingua characteristica universalis) gehört zu haben mit der es möglich war alles vorhandene zu beschreiben. Das funktioniert sicher auch für eine einfache Kaffeemaschine. Genau das würden Sie jetzt brauchen! Aber leider, die *lingua characteristica universalis* war ein Problem das Leibniz zeitlebens beschäftigt hat, ohne jedoch eine systematische Darstellung dieser zu finalisieren. Sie überlegen kurz die Leibniz'sche Universalsprache zu finalisieren, erkennen dabei aber schnell, dass dies ein sehr schwieriger und steiniger Weg ist, an dem viele vor Ihnen schon gescheitert sind. Bei Ihren Recherchen finden Sie jedoch heraus, dass die moderne Logik diese Anforderungen in den Grundzügen realisiert.

Besonders die hauptsächlich von Amir Pnueli eingeführte „Temporale Logik" [89] scheint für Ihr Kaffeemaschinenmodell brauchbar. Es soll also eine temporale Logik werden, eine Logik die keine kontinuierliche Zeit betrachtet, jedoch eine Vorher-Nachher Beziehung implementiert. Obwohl Ihnen die Bool'sche Logik aus Ihrer Ausbildung weitläufig bekannt ist, scheint die temporale Logik noch weitere Verknüpfungen zu haben, die Sie sich erst erarbeiten müssen. Sie wollen diese temporale Logik ja auch Ihren Teamkollegen weitergeben, in der Hoffnung, dass diese Sie bei der Erstellung der Spezifikation unterstützen. Einige Abende intensiven Studiums der temporalen Logiken [90, 91, 92, 93, 94, 95, 23, 9, 21] später, halten Sie folgendes fest:

Die in der Praxis am häufigsten verwendeten temporalen Logiken sind die Linear Temporal Logic (LTL) [90] und die Computation Tree Logic (CTL) [93]. Beide haben eine gemeinsame Obermenge die man CTL* nennt. LTL lässt sich gut auf Pfaden eines

[1] Gottfried Wilhelm Leibniz (1646 – 1716) war ein bedeutender deutscher Philosoph und Wissenschaftler, Mathematiker, Diplomat, Physiker, Historiker, Politiker, Bibliothekar und Doktor des Weltlichen und des Kirchenrechts.

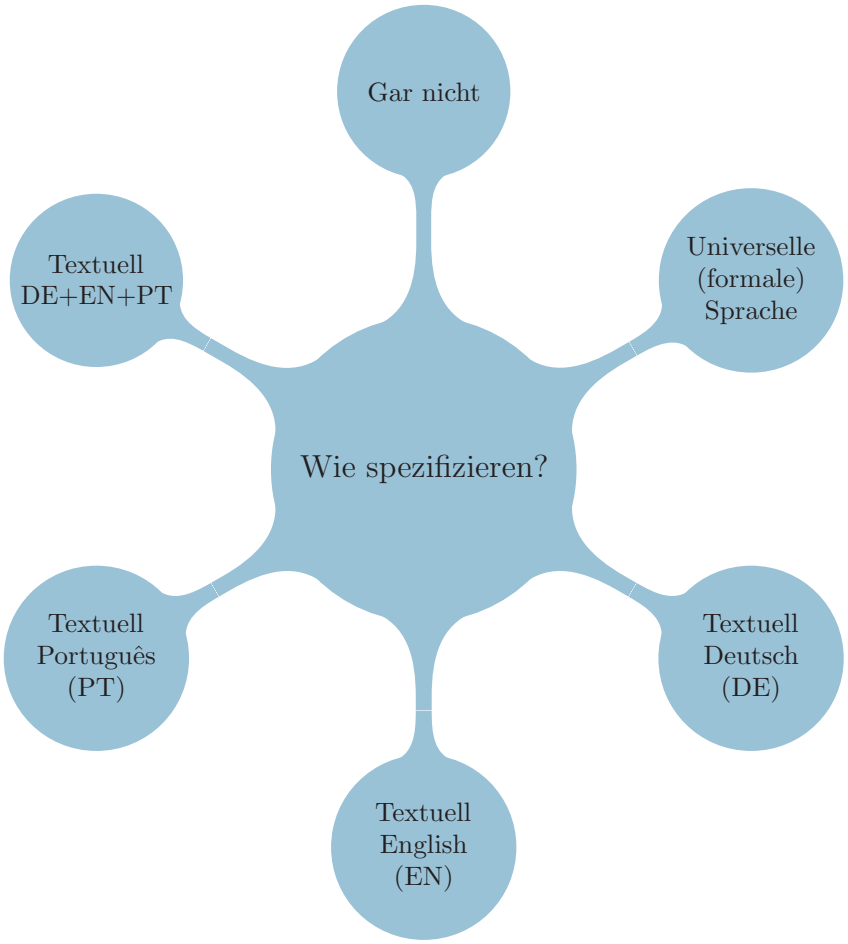


Abbildung 6.3: Spezifikationsarten – Das Ergebnis Ihres Brainstormings.

Systemmodells anwenden, wobei CTL sich besonders für Abarbeitungsbäume eignet. Sie entschließen sich für das Kaffeemaschinenmodell – aus persönlicher Präferenz – auf CTL zurückzugreifen. CTL ist eine temporale Logik, die von den beiden späteren Turing-Award Gewinnern Clarke & Emerson [96] schon 1980 eingeführt wurde [92].

Sie kommen zum Schluss, dass CTL im Grunde einfach eine um eine zeitliche Komponente erweiterte (Bool'sche) Logik ist. Das verwendete Zeitmodell ist die Baumstruktur eines Abarbeitungsbaumes. Es gibt nun neben den Bool'schen Operatoren weitere Operatoren die es ermöglichen, etwas über einen oder mehere Pfade in diesem Abarbeitungsbaumes auszusagen, bzw. formal zu spezifizieren.

Sie glauben fest an die Anwendbarkeit und Brauchbarkeit von CTL für Ihr Kaffeemaschinenmodell. Je mehr Sie sich in temporale Logiken einlesen, desto stärker zweifeln Sie jedoch daran, diese auch Ihren Teamkollegen und besonders dem brasilianischen Kaffeeexperten vermitteln zu können. Sie versuchen sich an ein paar einfachen Beispielen und schreiben die gefundenen Operatoren nacheinander auf:

Bool'sche Operatoren in CTL

$\boxed{\neg}$ NICHT

$\mathcal{A}$	$\neg\mathcal{A}$
0	1
1	0

Verneinung *(Wenn $\mathcal{A}$ wahr, dann $\neg\mathcal{A}$ falsch und umgekehrt)*

$\mathcal{A}$: Kaffee wird aus gemahlenen Kaffeebohnen hergestellt.
$\neg\mathcal{A}$: Kaffee wird *nicht* aus gemahlenen Kaffeebohnen hergestellt.

$\boxed{\wedge}$ UND

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \wedge \mathcal{B}$
0	0	0
0	1	0
1	0	0
1	1	1

Konjunktion *(Nur wenn $\mathcal{A}$ und $\mathcal{B}$ wahr sind)*
$\mathcal{A} \wedge \mathcal{B}$: Kaffee ist schwarz *und* koffeinhaltig.

$\boxed{\vee}$ ODER

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \vee \mathcal{B}$
0	0	0
0	1	1
1	0	1
1	1	1

Disjunktion *(Wenn mindestens eines ($\mathcal{A}$,$\mathcal{B}$) wahr ist)*
$\mathcal{A} \vee \mathcal{B}$: Die gerösteten Bohnen werden grob gemahlen *oder* im Mörser zerstampft.

$\boxed{\rightarrow}$ IMPLIKATION

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \rightarrow \mathcal{B}$
0	0	1
0	1	1
1	0	0
1	1	1

Hinreichende Bedingung *(Aus $\mathcal{A}$ folgt $\mathcal{B}$)*
$\mathcal{A} \rightarrow \mathcal{B}$: Wenn du Kaffee getrunken hast, bist du wach.

$\boxed{\leftrightarrow}$ ÄQUIVALENZ

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \leftrightarrow \mathcal{B}$
0	0	1
0	1	0
1	0	0
1	1	1

Bikondition *(Genau dann $\mathcal{A}$, wenn auch $\mathcal{B}$, bzw. $\mathcal{A}$ gleich $\mathcal{B}$)*
$\mathcal{A} \leftrightarrow \mathcal{B}$: Der Kaffee schmeckt nur dann gut, *wenn* ausgelesene Bohnen höchster Qualität verwendet werden.

Zu lösende Aufgaben:

1. Bool'sche Operatoren sind ein wichtiger Bestandteil der temporalen Logik CTL. Verwenden Sie jeden der Bool'schen Operatoren $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ und schreiben Sie eine Aussage in textueller Form an. Beispiel $\neg$: (Heute regnet es nicht).

Lösungsvorschläge:

1. Bool'sche Operatoren – mögliche Aussagen:

   $\boxed{\neg}$ NICHT

   $\mathcal{A}$: Es regnet.

   $\neg\mathcal{A}$: Es regnet nicht ($\neg$).

   $\boxed{\wedge}$ UND

   $\mathcal{A}$: 2010 ist das chinesische Jahr des Tigers.

   $\mathcal{B}$: 2010 ist das europäische Jahr zur Bekämpfung von sozialer Ausgrenzung und Armut.

   $\mathcal{A} \wedge \mathcal{B}$: 2010 ist das chinesische Jahr des Tigers und ($\wedge$) das europäische Jahr zur Bekämpfung von sozialer Ausgrenzung und Armut.

   $\boxed{\vee}$ ODER

   $\mathcal{A}$: Der sibirische Tiger wird auch Amurtiger genannt.

   $\mathcal{B}$: Der sibirische Tiger wird auch Ussuritiger genannt.

   $\mathcal{A} \vee \mathcal{B}$: Der sibirische Tiger wird auch Amur- oder ($\vee$) Ussuritiger genannt.

   $\boxed{\rightarrow}$ IMPLIKATION

   $\mathcal{A}$: Es wird wärmer.

   $\mathcal{B}$: Die Sonne scheint.

   $\mathcal{A} \rightarrow \mathcal{B}$: Wenn es wärmer wird, ($\rightarrow$) scheint die Sonne.

   $\boxed{\leftrightarrow}$ ÄQUIVALENZ

   $\mathcal{A}$: Heute ist Freitag.

   $\mathcal{B}$: Morgen ist Samstag.

   $\mathcal{A} \leftrightarrow \mathcal{B}$: Heute ist genau dann Freitag, ($\leftrightarrow$) wenn morgen Samstag ist.

   $\mathcal{B} \leftrightarrow \mathcal{A}$: Genau dann wenn morgen Samstag ist, ($\leftrightarrow$) ist heute Freitag.

Weiterführung des Beispiels, Begabtenförderung:

Eine mögliche Weiterführung des Beispiels stellt die Vorstellung der Kombination der Bool'schen Aussagenlogik untereinander dar. Wie zum Beispiel:

Erweiterte Bool'sche Operatoren in CTL

$\boxed{\neg \rightarrow \neg}$ Neg. der IMPLIKATION

$\mathcal{A}$	$\mathcal{B}$	$\neg\mathcal{B} \rightarrow \neg\mathcal{A}$
0	0	1
0	1	1
1	0	0
1	1	1

Hinreichende Bedingung *($\neg\mathcal{A}$ ist Voraussetzung für $\neg\mathcal{B}$)*

$\mathcal{A}$: Es wird wärmer.

$\mathcal{B}$: Die Sonne scheint.

$\mathcal{A} \rightarrow \mathcal{B}$: Wenn es wärmer wird, ($\rightarrow$) scheint die Sonne.

$\neg\mathcal{B} \rightarrow \neg\mathcal{A}$: Wenn nicht($\neg$) die Sonne scheint, ($\rightarrow$) wird es nicht($\neg$) wärmer.

$\boxed{\neg \leftrightarrow}$ Ausschließendes ODER

$\mathcal{A}$	$\mathcal{B}$	$\neg(\mathcal{A} \leftrightarrow \mathcal{B})$
0	0	0
0	1	1
1	0	1
1	1	0

Antivalenz *(Entweder $\mathcal{A}$ oder $\mathcal{B}$)*
$\mathcal{A}$: Kaffee ist schwarz.
$\mathcal{B}$: Kaffee ist braun.
$\neg(\mathcal{A} \leftrightarrow \mathcal{B})$: Kaffee ist *entweder* schwarz *oder* braun.

$\boxed{\neg\wedge}$ Neg. des UND

$\mathcal{A}$	$\mathcal{B}$	$\neg(\mathcal{A} \wedge \mathcal{B})$
0	0	1
0	1	1
1	0	1
1	1	0

Verneinung der Konjunktion *(Nicht dann wenn $\mathcal{A}$ und $\mathcal{B}$ wahr ist)*
$(\mathcal{A} \wedge \mathcal{B})$: Kaffee ist schwarz *und* koffeinhaltig.
$\neg(\mathcal{A} \wedge \mathcal{B})$: Es ist nicht der Fall, dass Kaffee schwarz und koffeinhaltig ist.

Zu lösende Zusatzaufgaben:

1. Verwenden Sie jeden der erweiterten Bool'schen Operatoren/Zusammenhänge $\{(\neg \rightarrow \neg), (\neg \leftrightarrow), (\neg\wedge)\}$ und schreiben Sie eine Aussage in textueller Form an.

Lösungsvorschläge:

1. Bool'sche Operatoren – mögliche Aussagen:

$\boxed{\neg \longrightarrow \neg}$ Neg. der IMPLIKATION
$\mathcal{A}$: Es wird kälter.
$\mathcal{B}$: Es ist Winter.
$\mathcal{A} \rightarrow \mathcal{B}$: Wenn es kälter wird, $(\rightarrow)$ ist Winter.
$\neg\mathcal{A} \rightarrow \neg\mathcal{B}$: Wenn es nicht $(\neg)$ kälter wird, $(\rightarrow)$ ist nicht $(\neg)$ Winter.
(Gleichwertig zu: $\mathcal{B} \rightarrow \mathcal{A}$: Wenn Winter ist, $(\rightarrow)$ wird es kälter[2].)

$\boxed{\neg \leftrightarrow}$ Ausschließendes ODER
$\mathcal{A}$: Wein ist rot.
$\mathcal{B}$: Wein ist weiß.
$\neg(\mathcal{A} \leftrightarrow \mathcal{B})$: Wein ist *entweder* rot *oder* weiß.

$\boxed{\neg\wedge}$ Neg. des UND
$(\mathcal{A} \wedge \mathcal{B})$: Wein ist süß *und* rot.
$\neg(\mathcal{A} \wedge \mathcal{B})$: Es ist nicht der Fall, dass Wein süß und rot ist.

[2] Weil: $\mathcal{B} \rightarrow \mathcal{A} = \neg\mathcal{B} \vee \mathcal{A} = \mathcal{A} \vee \neg\mathcal{B} = \neg\neg\mathcal{A} \vee \neg\mathcal{B} = \neg\mathcal{A} \rightarrow \neg\mathcal{B}$, aus Gries [97, p. 23]

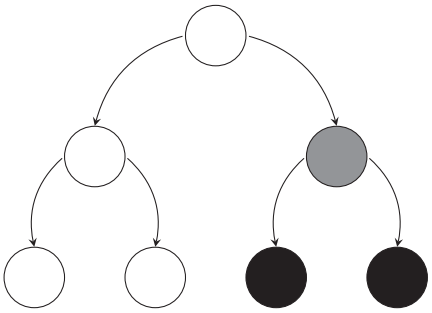
6.3 Block III – Kaffee mit Milch, Pfaden und temporalen Operatoren

6.3.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Kaffee mit Milch, Pfaden und temporalen Operatoren
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.117-D` `ETI-5.118-E` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Bool'sche Aussagenlogik, temporale Operatoren
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 120 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.3.2 Aufgabenstellung

Nachdem Sie sich durch den Dschungel der Bool'schen Operationen gekämpft haben, bleibt „nur“ mehr das Erklären für Ihre Kollegen. Sie sind aber voller Optimismus, dass Sie das mit Hilfe der vorbereiteten Beispiele schaffen werden. Weiters stellen Sie fest, dass Sie mit Aussagelogik (Bool'sche Logik) zwar ganz gut formulieren können was jetzt gerade im momentanen Zustand passiert, jedoch nicht, was in den Zuständen davor oder danach zu passieren hat. Nun wird Ihnen auch klar warum es in CTL neben den Bool'schen Operatoren auch noch so genannte temporale Operatoren gibt. Mit den temporalen Operatoren kann man angeben, was später einmal passieren soll. Also zum Beispiel, nachdem eine Münze in den Kaffeeautomaten eingeworfen wurde, müssen Sie im nächsten Zustand des Modells immer Kaffee bekommen. Um das genauer zu untersuchen, fertigen Sie folgende Skizze an:



Gedankenexperiment

Sie nehmen jetzt an, Ihr selbstgebauter Kaffeeautomat befindet sich in dem grauen Zustand. Sie wissen, dass in dem Zustand bereits Geld eingeworfen wurde. Jetzt möchten Sie gerne in Ihre Spezifikation folgendes schreiben:

> *Wenn ich in dem grauen Zustand bin, dann bekomme ich im nächsten Zustand (einer der beiden schwarzen Zustände) sicher Kaffee.*

Ihnen als HTL Ingenieur wird rasch klar, es wird ein Operator benötigt der dieses Verhalten ausdrücken kann. Und mit den temporalen Operatoren für CTL finden Sie genau das, was Sie suchen.

Temporale Operatoren in CTL (Computational Tree Logic)

$\boxed{\mathcal{X}}$ Im nächsten Zustand

(ne$\mathcal{X}$t)

Annahme/Aussage φ: Kaffee wird gemahlen.
$\mathcal{X}\varphi$: Im nächsten Zustand gilt die Aussage φ.
$\mathcal{X}\varphi$: Kaffee wird im nächsten Zustand (vom akutellen aus gesehen) gemahlen.

$\boxed{\mathcal{F}}$ Irgendwann in der Zukunft

($\mathcal{F}$uture)

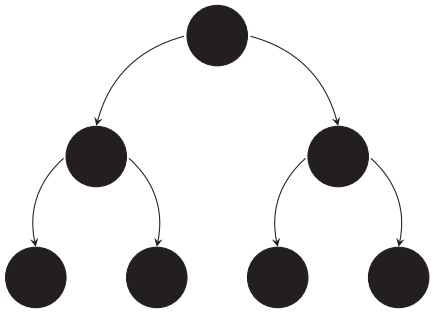
Annahme/Aussage φ: Kaffee wird gemahlen.
$\mathcal{F}\varphi$: Irgendwann in der Zukunft (aber mindestens einmal) gilt die Aussage φ.
$\mathcal{F}\varphi$: Irgendwann in der Zukunft (aber mindestens einmal) wird Kaffee gemahlen.

$\boxed{\mathcal{G}}$ Immer in der Zukunft

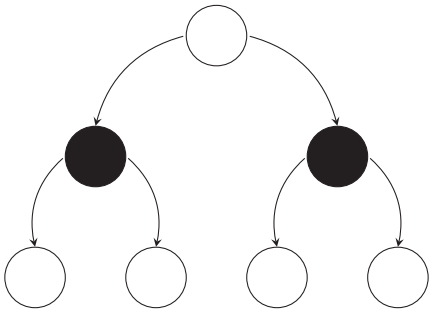
($\mathcal{G}$lobally)

Annahme/Aussage φ: Kaffee wird gemahlen.
$\mathcal{G}\varphi$: Von jetzt an gilt immer die Aussage φ.
$\mathcal{G}\varphi$: Von jetzt an wird immer Kaffee gemahlen.

Die Kombination aus Bool'schen und temporalen Operatoren steigert Ihre Zuversicht die temporale Logik verstanden zu haben. Sie fertigen noch einmal eine Skizze an, um die Operatoren zu wiederholen.

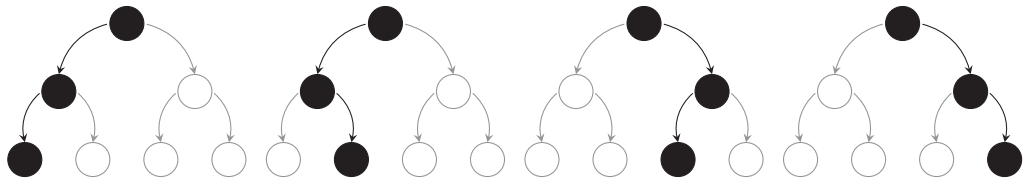


($\mathcal{G}$lobally) – Für immer (in aller Zukunft)



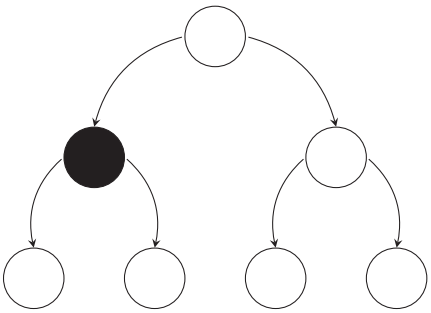
($\mathcal{F}$inally) – Irgendwann in der Zukunft

Bei dem $\mathcal{G}$lobally Operator muss daher die Aussage von jetzt an immer erfüllt sein (in allen Nachfolgerzuständen). Für den $\mathcal{F}$inally Operator muss die Aussage irgendwann in der Zukunft (mindestens einmal) erfüllt sein. Sie beginnen jetzt den von Ihnen gezeichneten Abarbeitungsbaum genauer zu analysieren. Für das $\mathcal{G}$lobally Beispiel setzen Sie sich auf den ersten Knoten – den Wurzelknoten – des Abarbeitungsbaums. Von dort aus sehen Sie zwei Verzweigungen (links, rechts) und jede dieser zwei Verzweigungen hat wiederum zwei Verzweigungen, somit ergeben sich insgesamt sieben Knoten. Sie suchen sich nun genau einen Weg durch den Abarbeitungsbaum, d.h., links oder rechts. Genau „ein Weg“ stellt einen Pfad dar. Insgesamt hat das Beispiel vier mögliche Wege, die Sie übersichtshalber nochmal nebeneinander darstellen:

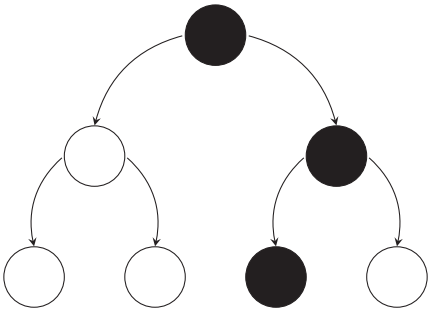


(a) Pfad 1 (b) Pfad 2 (c) Pfad 3 (d) Pfad 4

Sie erkennen also, dass es noch eine Möglichkeit geben muss um eine Aussage entweder für den gesamten Abarbeitungsbaum oder nur für mindestens einen Pfad niederzuschreiben. Sie brauchen daher ein spezielles „$\mathcal{F}$inally“ und ein spezielles „$\mathcal{G}$lobally“ das sich wie folgt darstellt:



„spezielles“ $\mathcal{F}$inally – Irgendwann auf einem Pfad



„spezielles“ ($\mathcal{G}$lobally) – Immer auf einem Pfad

Immer noch besessen vom Traum der eigenen Kaffeemaschine tauschen Sie nocheinmal mittelklassige Fernsehinhalte mit einschlägiger Fachliteratur und stoßen auf die Pfadoperatoren in CTL. Bingo! Genau das brauchen Sie jetzt.

Pfadoperatoren in CTL

$\boxed{\mathcal{A}}$ Für alle Pfade

($\mathcal{A}$lle)

Annahme/Aussage φ: Kaffee wird gemahlen.
$\mathcal{AG}\varphi$: In allen folgenden Zuständen gilt immer Aussage φ.
$\mathcal{AF}\varphi$: In allen folgenden Zuständen gilt irgendwann Aussage φ.
$\mathcal{AG}\varphi$: In allen folgenden Zuständen wird immer Kaffee gemahlen.
$\mathcal{AF}\varphi$: In allen folgenden Zuständen wird irgendwann Kaffee gemahlen.

$\boxed{\mathcal{E}}$ Für mindestens einen Pfad

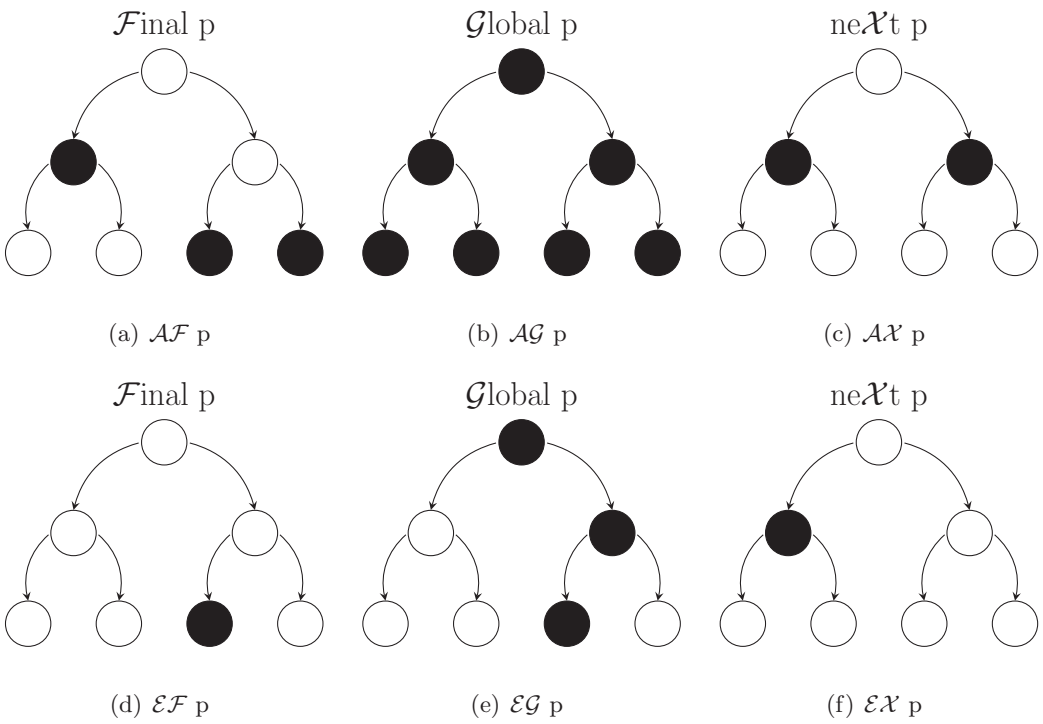
($\mathcal{E}$in)

Annahme/Aussage φ: Kaffee wird gemahlen.
$\mathcal{EG}\varphi$: Auf mindestens einem folgenden Pfad gilt immer die Aussage φ.
$\mathcal{EF}\varphi$: Auf mindestens einem folgenden Pfad gilt irgendwann die Aussage φ.
$\mathcal{EG}\varphi$: Auf mindestens einem folgenden Pfad wird immer Kaffee gemahlen.
$\mathcal{EF}\varphi$: Auf mindestens einem folgenden Pfad wird irgendwann Kaffee gemahlen.

Bool'sche Operatoren, Temorale Operatoren und dann auch noch Pfadoperatoren. Wie sollen Sie das nur Ihrem Team erklären? Ihre Kollegen wollen doch nur Melange, Verlängerten und Espresso, ob die Verständnis für $\mathcal{AF}$, $\mathcal{AG}$, $\mathcal{AX}$, $\mathcal{EF}$, $\mathcal{EG}$, $\mathcal{EX}$ haben? Bilder sagen bekanntlich mehr also 1000 CTL Formeln, daher erarbeiten Sie die Gegenüberstellung in Abbildung 6.4.

Zu lösende Aufgaben:

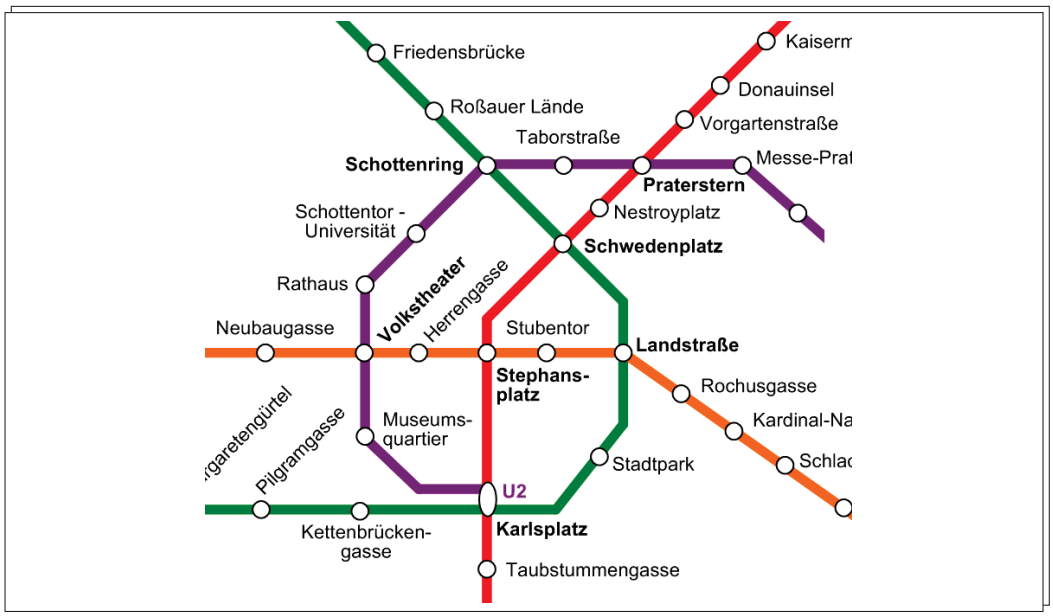
1. Während einer Fahrt mit den Wiener Linien entdecken Sie, dass das Wiener U-Bahnnetz auch mit nicht mehr als ein paar Zuständen und Übergängen modelliert werden kann. In Ihrem Gedankenmodell vereinfachen Sie das Modell, sodass sämtliche Züge nur in folgende Richtungen verkehren (i) U4 von Schottenring Richtung Karlsplatz, (ii) U2 von Praterstern bis Karlsplatz, (iii) U1 von Praterstern Richtung Karlsplatz und (iv) U3 von Volkstheater Richtung Landstraße. Sie beginnen nun Ihre Fahrt in der Taborstraße und wollen in die Universitätsbibliothek der TU Wien die sich am Karlsplatz befindet. Welche möglichen Pfade können Sie identifizieren?



(a) $\mathcal{AF}$ p (b) $\mathcal{AG}$ p (c) $\mathcal{AX}$ p

(d) $\mathcal{EF}$ p (e) $\mathcal{EG}$ p (f) $\mathcal{EX}$ p

Abbildung 6.4: Übersicht der CTL Operatoren.



2. Für Ihre Fahrt von Taborstraße bis Karlsplatz, welche (vernünftige) Aussagen in CTL können Sie dafür treffen? Geben Sie die CTL Formeln und auch Ihre textuelle Repräsentation an.

3. Warum machen für Ihre U-Bahnfahrt folgende Aussagen bezüglich der Annahmen in (1) keinen Sinn? Annahme/Aussage φ: Ich bin am Karlsplatz. (i) $\mathcal{AG}\varphi$ und (ii) $\mathcal{EG}\varphi$ Was bedeuten diese Aussagen eigentlich?

Lösungsvorschläge:

1. U-Bahn Pfade
   - {*Taborstraße, Schottenring, Schottentor, Rathaus, Volkstheater, Museumsquartier, Karlsplatz*}
   - {*Taborstraße, Schottenring, Schottentor, Rathaus, Volkstheater, Herrengasse, Stephansplatz, Karlsplatz*}
   - {*Taborstraße, Schottenring, Schottentor, Rathaus, Volkstheater, Herrengasse, Stephansplatz, Stubentor, Landstraße, Stadtpark, Karlsplatz*}
   - {*Taborstraße, Schottenring, Schwedenplatz, Landstraße, Stadtpark, Karlsplatz*}
   - {*Taborstraße, Schottenring, Schwedenplatz, Stephansplatz, Karlsplatz*}
   - {*Taborstraße, Schottenring, Schwedenplatz, Stephansplatz, Stubentor, Landstraße, Stadtpark, Karlsplatz*}
2. U-Bahn Fahrt Aussagen
   - Annahme/Aussage φ: Ich bin am Karlsplatz.
     $\mathcal{GF}\varphi$: Ich komme auf allen Wegen irgendwann zum Karlsplatz.
     $\mathcal{EF}\varphi$: Ich komme auf mindestens einem Weg irgendwann zum Karlsplatz.
     $\mathcal{AF}\varphi$: Alle Wege (Pfade) passieren den Karlsplatz. (Alle U-Bahn Pfade kommen am Karlsplatz vorbei).
3. U-Bahn Fahrt Falschaussagen
   - Annahme/Aussage φ: Ich bin am Karlsplatz.
     $\mathcal{AG}\varphi$: Ich bin immer am Karlsplatz (Alle Stationen sind der Karlsplatz).
     $\mathcal{EG}\varphi$: Auf mindestens einem Weg (Pfad) ist immer der Karlsplatz. (Die Stationen mindestens einer U-Bahnlinie heißen alle Karlsplatz).

Anmerkungen, Erläuterungen und Ergänzungen zum Lerninhalt:

U-Bahnpläne sind etwas reelles, angreifbares mit dem die Schüler bereits vertraut sind. Die Idee den Wiener U-Bahnplan[3] als Zustände und Übergangsrelationen zu interpretieren, ermöglicht den spielerischen Zugang zur Modellierung. Die Namen der einzelnen U-Bahnstationen können als Bool'sche Variablen angesehen werden, die genau dann true sind, wenn man sich in der jeweiligen Station befindet. Auf eine Verschachtelung einzelner

[3] Eine nette unterstützende Hands-On Übung wäre mit dem mobile Tool qando (http://www.qando.at/) der Wiener Linien denkbar.

Formeln wird bewusst verzichtet. Die temporale Logik CTL umfasst neben den vorgestellten Operatoren auch noch weit schwierigere Konstrukte, wie den $\mathcal{R}$elease und den $\mathcal{U}$ntil Operator, die hier bewusst weggelassen wurden[4].

Um den Schülern die Sinnhaftigkeit einer formalen Spezifikation glaubhaft zu machen, können im Unterricht Beispiele aus realen Industrieprojekten besprochen werden. Ein paar Beispiele unterschiedlicher Schwierigkeitsgrade aus dem Paper [98] seien hier angeführt:

- Einfach

  AG(EF *mark*=`MARK_SENDVERSION`)
  Auf allen Pfaden muss immer gelten, dass es mindestens einen Pfad gibt auf dem irgendwann die Variable *mark* den Wert `MARK_SENDVERSION` annimmt.

- Mittel

  AG (*RxBuffer_1*<`4` & *RxBuffer_0*=`0` & *RxBuffer_3*<`4` & *RxBuffer_4*=`0`)
  RxBuffer_1 und *RxBuffer_3* muss auf allen Pfaden immer kleiner 4 sein und *RxBuffer_0* und *RxBuffer_4* muss immer 0 sein.

- Schwierig

  AG (*RxBuffer_6*=`0x24` & *RxBuffer_7*=`0x5A` & *RxBuffer_8*=`0x23`) ⇒ AF (*TxBuffer_5*=`0x24` & *TxBuffer_6*=$\texttt{CNT}_1$ & *TxBuffer_7*=$\texttt{CNT}_2$ & *TxBuffer_8*=`0x23`) Auf allen Pfaden (immer) muss gelten, dass wenn *RxBuffer_6*=`0x24` und *RxBuffer_7*=`0x5A` und *RxBuffer_8*=`0x23`, dann muss auf allen folgenden Pfaden irgendwann *TxBuffer_5*=`0x24` und *TxBuffer_6*=$\texttt{CNT}_1$ und *TxBuffer_7*=$\texttt{CNT}_2$ und *TxBuffer_8*=`0x23` gelten.

Weiterführung des Beispiels, Begabtenförderung:
Zu lösende Zusatzaufgaben:

1. Sie bekommen die Aufgabe ein Systemmodell für einen für Sie optimierten U-Bahn Betrieb aufzustellen. Die Spezifikation liest sich wie folgt: *Es kommt eine U-Bahn Garnitur zum Einsatz, die Sie direkt von Ihrem Wohnort (Taborstraße) non-stop zur TU Bibliothek am Karlsplatz bringt. Das heißt Sie können nur in der Taborstraße einsteigen und am Karlsplatz aussteigen. Bei allen anderen Stationen erklingt das vertraute „Achtung Zug fährt durch".* Für diese individuelle Verbindung sollen Sie nun Systemeigenschaften mit Hilfe von CTL spezifizieren. Sie definieren folgende

[4] In der industriellen Praxis kommen ohnehin meist nur Invariante Ausdrücke (z.B.: AG φ) zum Einsatz.

Bool'sche Variablen:

$$station\_Taborstraße = \begin{cases} \texttt{true} & \text{wenn Zug in Station Taborstraße,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Schottenring = \begin{cases} \texttt{true} & \text{wenn Zug in Station Schottenring,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Schottentor = \begin{cases} \texttt{true} & \text{wenn Zug in Station Schottentor,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Rathaus = \begin{cases} \texttt{true} & \text{wenn Zug in Station Rathaus,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Volkstheater = \begin{cases} \texttt{true} & \text{wenn Zug in Station Volkstheater,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Museumsquartier = \begin{cases} \texttt{true} & \text{wenn Zug in Station Museumsquartier,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$station\_Karlsplatz = \begin{cases} \texttt{true} & \text{wenn Zug in Station Karlsplatz,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$zug\_stopp = \begin{cases} \texttt{true} & \text{wenn Zug stehenbleibt,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$zug\_durchsage = \begin{cases} \texttt{true} & \text{bei Durchsage „Achtung Zug fährt durch”,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

- Übersetzen Sie die oben angegebene textuelle Spezifikation in eine temporale Spezifikation.

Lösungsvorschläge:

1. Temporale Spezifikation

- Textuelle Spezifikation: *Es kommt eine U-Bahn Garnitur zum Einsatz die Sie direkt von Ihrem Wohnort (Taborstraße) non-stop zur TU Bibliothek am Karlsplatz bringt.* Aussage: Der Zug bleibt irgendwann in der Taborstraße und irgendwann beim Karlsplatz stehen.

$$\mathrm{EF}(station\_Karlsplatz{=}\texttt{true}) \wedge \mathrm{EF}(station\_Taborstraße{=}\texttt{true})$$

Interpretation:
Es gibt mindestens einen Pfad auf dem der Zug in der Taborstraße stehen bleibt und mindestens einen Pfad auf dem der Zug am Karlsplatz stehen bleibt.
Anmerkungen/Probleme:
Die Variable *zug_stopp* wird nicht berücksichtigt.

Verbesserung:

$$\mathrm{AG}((station\_Karlsplatz{=}\texttt{true} \vee station\_Taborstraße{=}\texttt{true}) \rightarrow zug\_stopp{=}\texttt{true})$$

Interpretation:
Auf allen Pfaden (immer) muss gelten, dass wenn *station_Karlsplatz* `true` oder *station_Taborstraße* `true`, dann muss auch *zug_stopp* `true` sein.
Anmerkungen/Probleme:
Die Variable *zug_durchsage* wird nicht berücksichtigt.

Verbesserung:

$$\mathrm{AG}(station\_Karlsplatz{=}\texttt{true} \vee station\_Taborstraße{=}\texttt{true}) \rightarrow (zug\_stopp{=}\texttt{true} \wedge zug\_durchsage{=}\texttt{false})$$

Interpretation:
Auf allen Pfaden (immer) muss gelten, dass wenn *station_Karlsplatz* `true` oder *station_Taborstraße* `true`, dann muss auch *zug_stopp* `true` und *zug_durchsage* `false` sein.
Anmerkungen/Probleme:
Wie kann man sicherstellen, dass der Zug mindestens einmal in den Stationen Karlsplatz und Taborstraße kommt?

Verbesserung:

$$\mathrm{AG}((station\_Karlsplatz{=}\texttt{true} \vee station\_Taborstraße{=}\texttt{true}) \rightarrow (zug\_stopp{=}\texttt{true} \wedge zug\_durchsage{=}\texttt{false})) \wedge \mathrm{EF}\ station\_Karlsplatz{=}\texttt{true} \wedge \mathrm{EF}\ station\_Taborstraße{=}\texttt{true}$$

Interpretation:
Auf allen Pfaden (immer) muss gelten, dass wenn *station_Karlsplatz* `true` oder *station_Taborstraße* `true`, dann muss auch *zug_stopp* `true` und *zug_durchsage* `false` sein. Weiters muss *station_Karlsplatz* und *station_Taborstraße* mindestens einmal `true` sein.

Anmerkungen/Probleme:
Wie kann man sicherstellen, dass der Zug non-stop zwischen den Stationen Taborstraße und Karlsplatz verkehrt?

Verbesserung:

$$\begin{aligned}
&\mathrm{AG}((station_Karlsplatz{=}\texttt{true} \vee station_Taborstraße{=}\texttt{true}) \rightarrow (zug_durchsage{=}\texttt{false} \\
&\wedge zug_stopp{=}\texttt{true})) \wedge \mathrm{EF}\ station_Karlsplatz{=}\texttt{true} \wedge \mathrm{EF}\ station_Taborstraße{=}\texttt{true} \\
&\wedge \mathrm{AG}((station_Schottenring{=}\texttt{true} \rightarrow (zug_durchsage{=}\texttt{true} \wedge zug_stopp{=}\texttt{false})) \\
&\wedge (station_Schottentor{=}\texttt{true} \rightarrow (zug_durchsage{=}\texttt{true} \wedge zug_stopp{=}\texttt{false})) \\
&\wedge (station_Rathaus{=}\texttt{true} \rightarrow (zug_durchsage{=}\texttt{true} \wedge zug_stopp{=}\texttt{false})) \\
&\wedge (station_Volkstheater{=}\texttt{true} \rightarrow (zug_durchsage{=}\texttt{true} \wedge zug_stopp{=}\texttt{false})) \\
&\wedge (station_Museumsquartier{=}\texttt{true} \rightarrow (zug_durchsage{=}\texttt{true} \wedge zug_stopp{=}\texttt{false})))
\end{aligned}$$

Interpretation:
Auf allen Pfaden (immer) muss gelten, dass wenn *station_Karlsplatz* `true` oder *station_Taborstraße* `true`, dann muss auch *zug_stopp* `true` und *zug_durchsage* `false` sein. Weiters muss *station_Karlsplatz* und *station_Taborstraße* mindestens einmal `true` sein. Weiters muss auf allen Pfaden gelten, dass wenn sich der Zug nicht in der Station Karlsplatz oder Taborstraße befindet (in einer der andern ist), dann muss *zug_durchsage* `true` und *zug_stopp* `false` sein.

Bei diesem Beispiel wird ersichtlich, dass eine durchdachte Spezifikation durchaus aufwendig werden kann und auch Erfahrung mit dem Umgang und Interpretation der temporalen Operatoren erfordert.

6.4 Block IV – Kaffee in Sicht

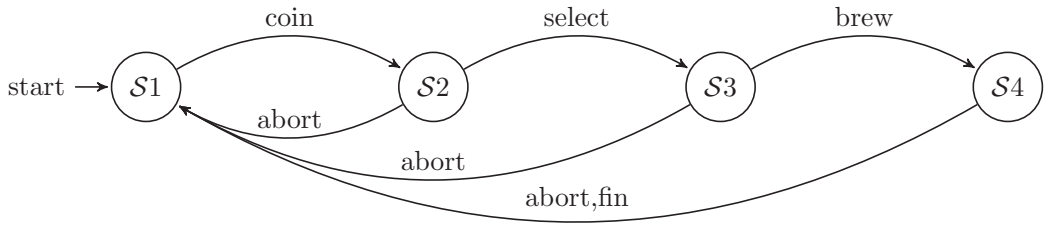
6.4.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Kaffee in Sicht
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.117-D` `ETI-5.118-E` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Bool'sche Aussagenlogik, temporale Operatoren
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 50 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.4.2 Aufgabenstellung

Das Arbeitsklima ist aufgrund des akuten Kaffeemangels der letzten Woche schwer angeschlagen. Viele Ihrer Kollegen verbringen die Hälfte Ihrer Mittagspause bei US amerikanischen, international tätigen Kaffeekonzernen. Andere wiederum kommen mit den Taschen voller Instant-Coffee ins Büro. Ihr brasilianischer Kollege hat sogar begonnen selbst geröstete Kaffeebohnen zu kauen. Die Lage ist brenzlig. Den Koffeinengpass können Sie definitiv nicht länger mitansehen. Sie haben den Firmenfrieden in der Hand. Ihre selbstgebaute Kaffeemaschine wird in der Lage sein die Stimmung in der Abteilung wieder herumzureißen. Wenn diese nur schon fertig wäre. Nur noch schnell spezifizieren und den Entwurf testen/verifizien ...

Nachdem Sie Ihren Kollegen in einem Crashkurs in das Gebiet der temporale Logiken einführen, rufen Sie sich das ursprüngliche Modell der Kaffeemaschine nochmals ins Gedächtnis:



Mit gemeinsamen Kräften machen Sie sich nun an die Spezifikation Ihrer Kaffeemaschine.

Zu lösende Aufgaben:

1. Für das Kaffeemaschinenmodell stellen Sie mindestens fünf CTL Formeln auf und beschreiben Sie in textueller Form was Sie damit aussagen möchten.

Lösungsvorschläge:

EF(*state*=S2)

Interpretation: Es gibt mindestens einen Pfad auf dem das Systemmodell in den Zustand S2 kommt.

EF(*state*=S3)

Interpretation: Es gibt mindestens einen Pfad auf dem das Systemmodell in den Zustand S1 kommt.

EF(*state*=S4)

Interpretation: Es gibt mindestens einen Pfad auf dem das Systemmodell in den Zustand S4 kommt.

AG EF(*state*=S1)

Interpretation: Es ist immer möglich in den Startzustand des Systemmodells zu kommen.

AG (*abort*=true → AF *state*=S1)

Interpretation: Wenn abort aktiv ist (Abbruch gedrückt wurde) dann ist es immer möglich in den Startzustand des Systemmodells zu kommen.

AG (*state*=S3 → !(EX *state*=S2))

Interpretation: Wenn das System im Zustand S3 ist, dann darf es im nächsten Zustand nie im Zustand S2 sein (Es gibt keine Transition von S3 nach S2).

AG (*state*=S4 → !(EX *state*=S3))

Interpretation: Wenn das System im Zustand S4 ist, dann darf es im nächsten Zustand nie im Zustand S3 sein (Es gibt keine Transition von S4 nach S3).

AG (*state*=S4 → !(EX *state*=S2))

Interpretation: Wenn das System im Zustand S4 ist, dann darf es im nächsten Zustand nie im Zustand S2 sein (Es gibt keine Transition von S4 nach S2).

Anmerkungen, Erläuterungen und Ergänzungen zum Lerninhalt:

Es gibt hier keine einzig richtige Aussage über das Modell oder falsche Aussage über das Modell. Genauso ist das auch in den Industrieprojekten mit der richtigen Spezifikation oder der falsche Spezifikation. Über das was spezifiziert wurde, muss absolute Klarheit herrschen. Dies spart spätere Unklarheiten in der Implementierungsphase. Als Unterrichtsmethode bietet sich hier das Gruppenpuzzle an, bei dem sich die Lernenden die aufgestellten Spezifikationen gegenseitig erklären sollen.

6.5 Block V – Kaffeemaschine in Perfektion

6.5.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Kaffeemaschine in Perfektion
Relevante(r) Deskriptor(en)	ETI-5.114-A ETI-5.115-B ETI-5.116-C ETI-5.117-D ETI-5.118-E (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Tools zur formalen Verifikation, Model Checker
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 120 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.5.2 Aufgabenstellung

Sie entscheiden sich die Verifikation Ihres Kaffeemaschinenmodells mit Hilfe eines vollautomatischen Tools durchzuführen. Sie machen sich schlau und finden heraus, dass ein sog. Modellprüfungstool (Model Checker) genau das ist, was Sie brauchen können. Sie haben ein Modell und wollen dieses irgendwie checken ⇒ Model Checker. Doch wie funktioniert denn so ein Model Checker genau? Nach dem Studium einschlägiger Fachliteratur [21, 23] halten Sie folgendes fest:

Ein Model Checker ist ein Tool zur automatisierten, formalen Verifikation. Im Wesentlichen benötigt ein Model Checker ein Systemmodell und eine Spezifikation. Die Spezifikation wird dann gegen das Systemmodell geprüft. Wird die Spezifikation erfüllt so wird eine Bestätigung ausgegeben, ist dies nicht der Fall so erhält man ein Gegenbeispiel, das genau aufzeigt warum die Spezifikation nicht eingehalten wurde. Das Gegenbeispiel kann man sich so vorstellen, als ob einem ein allwissender Guide (der Model Checker) auf eine Reise durch das Systemmodell mitnimmt und genau zeigt warum das Systemmodell fehlerhaft ist bzw. die Spezifikation nicht erfüllt wird. Die Spezifikation selbst ist für die meisten Tools eine temporale Spezifikation, wie etwa CTL. Für Ihren Anwendungsfall würde der Ablauf zur Verifikation der Kaffeemaschine wie in Abbildung 6.5 aussehen.

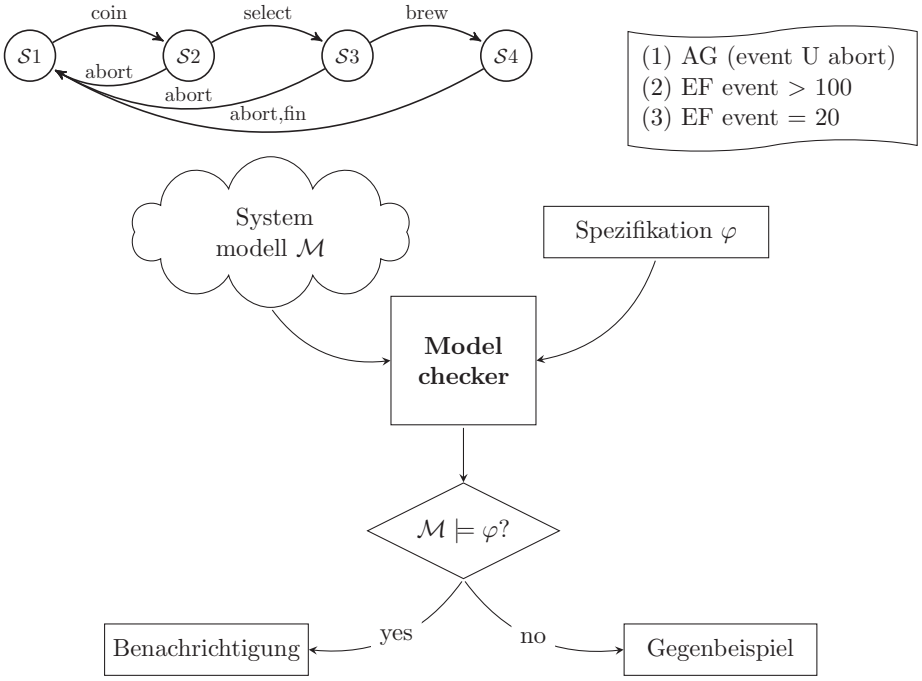


Abbildung 6.5: Die Verifikation der Kaffeemaschine mittels Model Checker.

Sie überlegen sich nun wie denn der Ablauf für die Kaffeemaschine aussehen könnte. Sie stellen fest, dass Sie das Systemmodell bereits fertig haben. Eine Spezifikation sollte sich mit Hilfe von CTL auch erstellen lassen, doch woher denn einen Model Checker nehmen?

Sie wissen, im Prinzip ist der Model Checker nur dafür verantwortlich alle möglichen Zustände und Kombinationen Ihres Systemmodells zu erstellen und dabei zu überprüfen ob Ihre Spezifikation in jedem einzelnen Zustand gültig ist. Ihre Anforderungen an den Model Checker sind daher:

- Es muss möglich sein das Systemmodell als State Machine (SM) angeben zu können.
- Es muss möglich sein die Spezifikation in CTL angeben zu können.
- Sie wollen keine seitenlangen Manuals lesen.
- Besonders wichtig: Sie wollen ein OpenSource Tool verwenden um Kosten zu sparen.

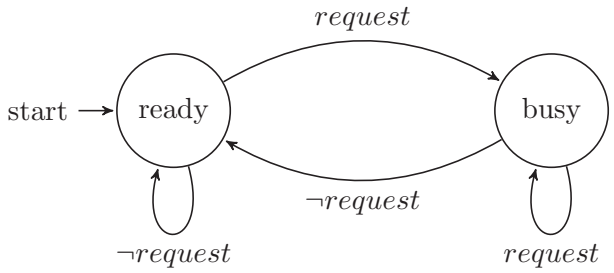
Die Thematik model checking ist ein noch immer sehr belebtes in der gegenwärtigen Forschung und Entwicklung von sicherheitskritischen Systemen. Neben den vielen kommerziellen Tools am Markt gibt es auch eine Reihe von brauchbaren Model Checker, die aus Forschungsprojekten entstanden sind. Ein Tool, das Ihre oben genannten Anforderungen erfüllt ist der NuSMV [99, 100] Model Checker.

Das Tool arbeitet konsolenbasiert, d.h. es verwendet keine graphische Benutzeroberfläche (GUI). Es wird über Kommandos gesteuert. Ein erster Aufruf macht einen ernüchternden Eindruck:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV
*** This is NuSMV 2.4.3 (compiled on Tue May 22 13:50:47 UTC 2007)
*** For more information on NuSMV see <http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson
```

Sie suchen nach einer Möglichkeit Ihre Kaffemaschine in NuSMV zu laden. Sie versuchen es mit einem trivialen Beispiel zum Einstieg. Sie probieren es mit diesem Modell das aus zwei Zuständen und dem Event *request* besteht:



Um dieses Modell in NuSMV zu verwenden erstellen Sie mit einem Texteditor die Datei `sm.smv` mit folgendem Inhalt:

```
1   MODULE main
2   VAR
3       request : boolean;
4       state   : {ready,busy};
5   ASSIGN
6       init(state) := ready;
7
8       next(state) := case
9             state = ready & request   : busy;
10            1                         : {ready,busy};
11      esac;
```

Die Bedeutung der einzelnen Zeilen:

Zeile 1 : Das Schlüsselwort `MODULE` kennzeichnet den Beginn des Files und somit der Beschreibung des Systemmodells.

Zeile 2 : Nach dem Schlüsselwort `VAR` werden „Variablen“ und Zustände definiert. Mögliche Datentypen sind zum Beispiel `boolean` und `integer`.

Zeile 3 : Hier wird die Variable `request` vom Datentyp boolean angelegt. Der Initialwert der Variable wird nicht spezifiziert, kann daher sowohl `true` als auch `false` sein.

Zeile 4 : Anlegen der Zustandsvariable des Automaten. Die Zustandsvariable beinhaltet alle möglichen Zustände in denen sich das Modell befinden kann. Dieses Modell hat zwei Zustände, nämlich `ready` und `busy`.

Zeile 5 : Nach dem Schlüsselwort `ASSIGN` stehen Zuweisungen.

Zeile 6 : Der Zustand `ready` wird als Initialzustand festgelegt.

Zeile 8 : Wie aus anderen Programmiersprachen bekannt, wird ein `case` Befehl verwendet um eine Mehrfachauswahl für den nächsten Zustand zu ermöglichen.

Zeile 9 : Wenn der aktuelle Zustand `ready` ist und `request true` ist, dann ist der nächste Zustand `busy`.

Zeile 10 : Für alle anderen Fälle kann der nächste Zustand entweder `ready` oder `busy` sein.

Zeile 11 : Abschluss des `case` Befehls mit `esac`.

Mit dem folgenden Befehl laden Sie Ihr Systemmodell in NuSMV:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV -int sm.smv
NuSMV> go
NuSMV>
```

Somit ist der Einstieg in die Welt der formalen Verifikation vorerst geschafft und Sie sind bereit den Model Checker für sich arbeiten zu lassen!

Zu lösende Aufgaben:

1. Organisieren Sie sich eine kostenlose Kopie des Model Checkers NuSMV und installieren Sie diese auf Ihrem Arbeitsplatz (Linux oder Windows Version).
2. Starten Sie eine Online-Recherche und beschreiben Sie mindestens drei industrielle Anwendungen bei denen NuSMV zur Verifikation eingesetzt wurde. Mögliche Ausgangspunkte Ihrer Recherche: http://nusmv.irst.itc.it/ und http://de.wikipedia.org/wiki/Model_Checking
3. Machen Sie sich mit den Kommandozeilenoptionen {`go`, `print_reachable_states -v`, `pick_state -r`, `print_current_state -v`, `simulate -r`, `show_traces -v`} von NuSMV vertraut. Verwenden Sie das `sm.smv` Beispiel und probieren Sie die Befehle aus.
4. Wieviele mögliche Zustände hat das `sm.smv` Beispiel? Schreiben Sie erst Ihre Überlegungen auf einem Blatt Papier auf und überprüfen Sie diese mit NuSMV.

Lösungsvorschläge:

1. NuSMV kann gegenwärtig von http://nusmv.irst.itc.it/ bezogen werden. Es sind Versionen für Linux und Windows Betriebssysteme vorhanden.
2. Aus http://nusmv.fbk.eu/examples/examples.html: A model for the PCI Bus protocol, A model of a robotics controller, Shuttle Digital Autopilot engines out (3E/O) contingency guidance requirements ...
3. Die Befehle können direkt in NuSMV ausprobiert werden, oder auch im User Manual nachgeschlagen werden.
4. Das System hat zwei Zustände und ein Event (Bool'sche Variable). Die Gesamtanzahl der Zustände – der Zustandsraum – umfasst somit folgende Konfigurationen:

$$\mathcal{S} := \{s_0, s_1, s_2, s_3\}$$

$$s_0 := \{state = \texttt{ready},\ request = \texttt{false}\}$$
$$s_1 := \{state = \texttt{ready},\ request = \texttt{true}\}$$
$$s_2 := \{state = \texttt{busy},\ request = \texttt{false}\}$$
$$s_3 := \{state = \texttt{busy},\ request = \texttt{true}\}$$

In NuSMV kann genau diese Ausgabe mit folgendem Befehl geladen werden:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV -int sm.smv
NuSMV> go
NuSMV> print_reachable_states -v
######################################
system diameter: 2
reachable states: 4 (2^2) out of 4 (2^2)
------- State 1 ------
request = 1
state = ready
state.0 = 1
------- State 2 ------
request = 1
state = busy
state.0 = 0
------- State 3 ------
request = 0
state = ready
state.0 = 1
------- State 4 ------
request = 0
state = busy
state.0 = 0
--------------------------
######################################
NuSMV>
```

Anmerkungen, Erläuterungen und Ergänzungen zum Lerninhalt:

Der Model Checker NuSMV wurde für das Übungsbeispiel ausgewählt weil er (i) ein in der Industrie tatsächlich verwendetes Tool ist und (ii) eine der am einfachsten zu verwendenden Lösungen darstellt. Besonders die Steuerung über die Konsole ist einfach zu verstehen und die Formulierungssprache erlaubt es Automaten direkt zu spezifizieren.

Viele der modernen Tools zum Softwaretest und der Verifikation sind über die Konsole steuerbar und können somit durch Kommandozeilen Skripte zu Testumgebungen zusammengebaut werden. Es ist daher wichtig, dass sich die Lernenden mit einem dieser Tools genauer auseinandersetzen.

Weiterführung des Beispiels, Begabtenförderung:

Auf der Website des NuSMV Projektes (http://nusmv.fbk.eu/examples/examples.html) gibt es eine Menge an industriellen Beispielen die mit Hilfe dieses Model Checkers modelliert und verifiziert wurden. Begabte Lernende können sich selbst in diese einarbeiten.

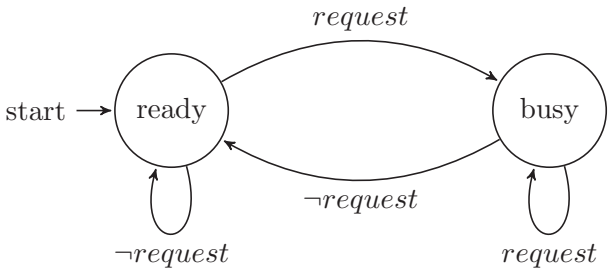
6.6 Block VI – Alles hat seine Ordnung

6.6.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Alles hat seine Ordnung
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.116-C` `ETI-5.117-D` `ETI-5.118-E` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Tools zur formalen Verifikation, Model Checker
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [24]. Abstraktionsbegriff frei nach [88].
Zeitbedarf in Minuten	Richtwert: 100 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.6.2 Aufgabenstellung

Nachdem Sie sich mit dem Model Checker NuSMV vertraut gemacht haben, fühlen Sie sich mit dem nötigen Rüstzeug ausgestattet, die Kaffeemaschine fertig zu spezifizieren, und somit die Spezifikation und Designphase abzuschließen. Sie suchen aber noch nach einer Möglichkeit NuSMV Ihre CTL Spezifikation mitzugeben. Für das Minimalbeispiel



wollen Sie folgende CTL Spezifikation überprüfen (auf allen Pfaden (immer) muss gelten, dass irgendwann der Automat in den Zustand `busy` übergeht):

AF(*state*=`busy`)

Nach Studium des Manuals stellen Sie fest, dass es möglich ist CTL Spezifikationen direkt in die Modellierungsdatei mit dem Schlüsselwort `SPEC` einzubringen:

```
1   MODULE main
2   SPEC AF (state=busy)
3   VAR
4       request : boolean;
5       state : {ready,busy};
6   ASSIGN
7       init(state) := ready;
8
9       next(state) := case
10            state = ready & request    : busy;
11            1                          : {ready,busy};
12      esac;
```

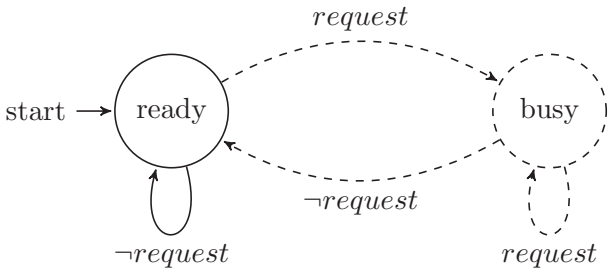
Der Verifikationslauf mit NuSMV führt zu folgendem Ergebnis:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV sm.smv
-- specification AF state = busy is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-- Loop starts here
-> State: 1.1 <-
request = 0
state = ready
-> Input: 1.2 <-
-> State: 1.2 <-
NuSMV>
```

Der Model Checker gibt ein Gegenbeispiel (Counterexample) aus, das es Ihnen ermöglicht, zu verstehen warum Ihre CTL Spezifikation von dem Modell nicht eingehalten wurde. Sie befinden sich im state `ready` und die logische Variable `request` ist 0. Gibt es nun keine Änderung von `request` so kommt das Modell nie in den Zustand `busy` und somit wird Ihre Spezifikation nicht erfüllt. Folgendes Bild verdeutlicht dies:

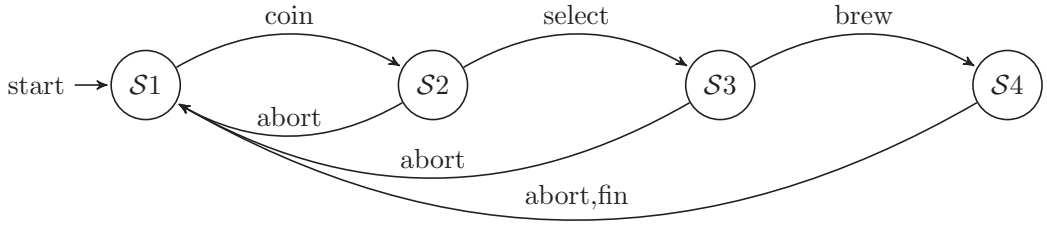
Zu lösende Aufgaben:

1. Modellieren Sie Ihre Kaffeemaschine in NuSMV.
2. Verifizieren Sie das Modell gegen die von Ihnen aufgestellten CTL Systemeigenschaften.
3. Wie viele verschiedene Zustände hat das Systemmodell? Zeigen Sie wie diese Anzahl erreicht wird.



Lösungsvorschläge:

1. Kaffeemaschine als NuSMV Modell:
   Der zuvor festgelegte Automat



   wird wie folgt in ein NuSMV Modell übergeführt. Die Bool'schen Events *coin, select, brew, abort, fin* werden durch den Datentyp boolean modelliert. Die Statemachine enthält vier States *S1, S2, S3* und *S4* wobei in den `case` Zweigen die Übergangsrelationen modelliert werden:

```
1   MODULE main
2   SPEC EF (state=S1)
3   SPEC EF (state=S2)
4   SPEC EF (state=S3)
5   SPEC EF (state=S4)
6   SPEC AG (abort=true -> AF state=S1)
7   SPEC AG (state=S3 -> !(EX state=S2))
8   SPEC AG (state=S4 -> !(EX state=S3))
9   SPEC AG (state=S4 -> !(EX state=S2))
10  VAR
11      coin    : boolean;
12      brew    : boolean;
13      select  : boolean;
14      abort   : boolean;
15      fin     : boolean;
16
17      state   : {S1,S2,S3,S4};
18  ASSIGN
19      init(state) := S1;
20
```

```
21        next(state) := case
22              state = S1 & coin          : S2;
23              state = S2 & select        : S3;
24              state = S3 & brew          : S4;
25              state = S4 & fin           : S1;
26              abort                      : S1;
27              1                          : S1;
28        esac;
```

2. Automatische Verifikation mit NuSMV:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV coffee.smv
-- specification AG (EF state = S1) is true
-- specification AG (EF state = S2) is true
-- specification AG (EF state = S3) is true
-- specification AG (EF state = S4) is true
-- specification AG (abort -> AF state = S1) is true
-- specification AG (state = S3 -> !(EF state = S2)) is true
-- specification AG (state = S4 -> !(EF state = S3)) is true
-- specification AG (state = S4 -> !(EF state = S2)) is true
```

3. Anzahl der möglichen Systemzustände:
   Das System hat vier Event Variablen. Jede davon kann entweder 0 oder 1 sein. Insgesamt gibt es vier Zustände in denen sich die Statemachine befinden kann, daher ist die Gesamtanzahl der Zustände: Kombination der Events (Variablen) $\times$ Anzahl der Zustände des Automaten $= 2^5 \times 4 = 128$, oder mit NuSMV:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV -int
coffee.smv
NuSMV> go
NuSMV> print_reachable_states
######################################
system diameter: 4
reachable states: 128 (2^7) out of 128 (2^8)
######################################
NuSMV>
```

6.7 Block VII – Ein Auto, ein Computer, ein Mann

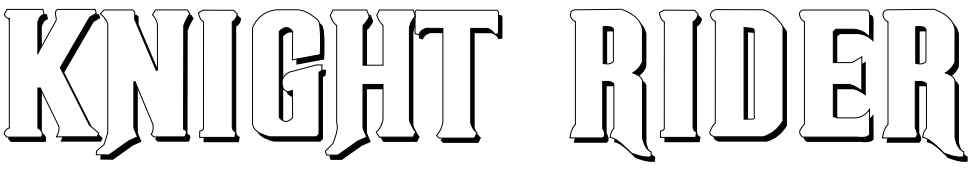
6.7.1 Einordnung in das Kompetenzmodell

Fachgruppe	Elektrotechnik – Informationstechnik
Titel	Ein Auto, ein Computer, ein Mann
Relevante(r) Deskriptor(en)	`ETI-5.114-A` `ETI-5.115-B` `ETI-5.116-C` `ETI-5.117-D` `ETI-5.118-E` (Aus Abschnitt 5.2.4)
Themenbereiche und Fertigkeiten	Modellierung, Systeme verstehen, Abstraktion, Statemachines, Automaten, formale Spezifikation, temporale Logiken, Tools zur formalen Verifikation, Model Checker
Methodisch/Didaktische Hinweise	Siehe Hinweise in Abschnitt 6.8.
Hilfsmittel	evtl. Computer, Internet, Fachliteratur – ausgewähle Kapitel aus [9, 88, 23, 21].
Quelle	Idee aus [101].
Zeitbedarf in Minuten	Richtwert: 180 Minuten (Abhängig von Unterrichtsform, siehe 6.8).

6.7.2 Aufgabenstellung

Sie haben es geschafft! Der Firmenfrieden ist wiederhergestellt. Nachdem Sie die Kaffeemaschine für Ihre Abteilung fertiggestellt haben genießen Sie höchtses Ansehen bei Ihren Kollegen. Davon hat auch Ihr Chef Wind bekommen und bestellt Sie in sein Büro, um mit Ihnen über etwas „Größeres“ zu sprechen. Er habe einen dicken Fisch an Land gezogen, sagt er. Er meint, er brauche jemanden auf den er sich verlassen kann!

Es geht um ein Steuergerät für ein Automobil. Und es handelt sich nicht um irgendein Automobil. Es geht nicht um eine zweitklassige Familienkutsche. Es geht um DEN Automobiltraum Ihrer Jugend. Es handelt sich um den legendären `K.I.T.T.` den `Knight Industries Two Thousand`.



Für eine Neuverfilmung der TV-Serie Knight Rider rund um den Helden Michael Knight wird ein neuer `K.I.T.T.` gebaut und Ihre Firma hat den Zuschlag für den Auftrag bekommen. Ihr Unternehmen will in die Filmbranche einsteigen und kann es sich keinesfalls leisten mit etwaigen Fehlfunktionen schlechte Kritiken einzufangen. Es steht viel am Spiel.

Einerseits lastet der Druck Ihres Chefs auf Ihren Schultern, andererseits bekommen Sie butterweiche Knie, wenn Sie nur daran denken, dass Millionen Fans auf dem Globus von dem neuen `K.I.T.T.` bitter enttäuscht werden könnten.

Ihre konkrete Aufgabe ist es, ein Steuergerät für den neuen `K.I.T.T.` zu entwickeln, das den `SUPER PURSUIT MODE` unterstützt. Nachdem die erste Nervosität geschwunden ist, sehen Sie Ihrer Aufgabe gelassen entgegen, denn Sie kennen sich aus mit Modellierung und formaler Verifikation. Kein Problem meinen Sie! Eigentlich ist alles ganz einfach, Sie haben früher jede einzelne Folge von Knight Rider gesehen und wissen genau wie der `SUPER PURSUIT MODE` funktioniert:

> Es gibt zwei Bool'sche Eingänge welche die Entscheidungen von Michael Knight darstellen. Michael kann beschleunigen und bremsen. Wenn beschleunigt wird, dann bewegt sich `K.I.T.T.` erst mit langsamer Beschleunigung. Wird weiter beschleunigt, so aktiviert `K.I.T.T.` den `SUPER PURSUIT MODE` (SPM) und kann dann bis zu 300 Mph (ca. 480 km/h) schnell fahren. Falls Michael bremst bleibt das Fahrzeug sofort stehen.

Für Ihre Modellierung verwenden Sie folgende Event-Variablen:

$$accelerate = \begin{cases} \texttt{true} & \text{wenn Michael beschleunigt,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

$$brake = \begin{cases} \texttt{true} & \text{wenn Michael bremst,} \\ \texttt{false} & \text{andernfalls.} \end{cases}$$

Natürlich sind auch dem `Knight Industries Two Thousand` gewisse Grenzen basierend auf den physikalischen Grundgesetzen gesetzt. Luftwiderstand (c_w Wert) und Reibung (μ) zum Beispiel. Wird vom Stillstand beschleunigt, so geht `K.I.T.T.` erst in einen normalen Beschleunigungszustand über und dann bei anhaltender Beschleunigung in den `SUPER PURSUIT MODE`.

Für die Neuverfilmung hat die langjährige Chefmechanikerin Dr. Bonnie Barstow (Die Erfinderin des `SPM`) monatelang im Labor und in der Werkstatt des SEM-I (mobiler Einsatztruck) verbracht um ein neuartiges Bremssystem zu entwickeln. Das Prinzip selbst unterliegt strengster Geheimhaltung (Klassifizierung Top Secret der Foundation für Recht und Verfassung), nur soviel ist bekannt: Es nennt sich `Zero Brake Delay (ZBD)` und vermag es `K.I.T.T.` innerhalb eines Bruchteils einer Sekunde zum Stehen zu bringen. Wenn Michael nun bremst, so bleibt `K.I.T.T.` sofort stehen.

Sie machen sich ans Werk und überlegen sich folgende Eigenschaften, die das System mindestens erfüllen muss:

Immer dann, wenn gebremst wird, muss `K.I.T.T.` stehen bleiben.

AG(*brake*=`true` → AX *state*=`stop`)

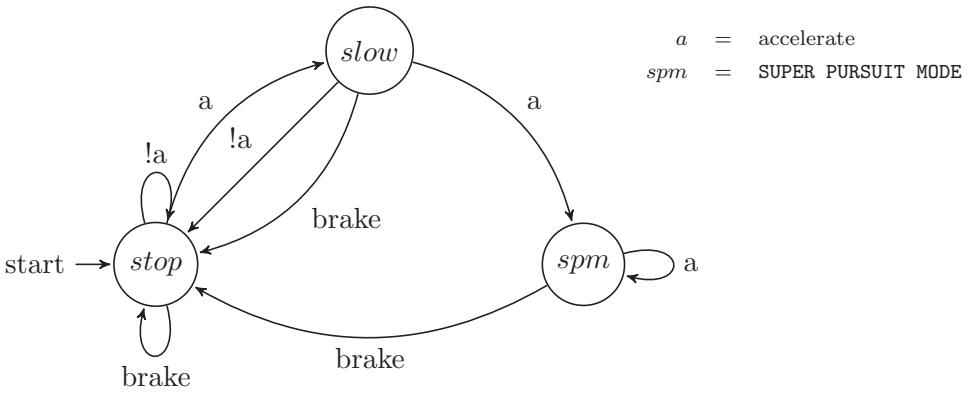
Immer wenn beschleunigt und nicht gebremst wird, dann muss die Beschleunigung erst langsam und dann schnell sein (`SUPER PURSUIT MODE`).

(i) AG(¬*brake* ∧ *accelerate* ∧ *state*=`stop` → AX *state*=`slow`)
(ii) AG(¬*brake* ∧ *accelerate* ∧ *state*=`slow` → AX *state*=`spm`)

Immer wenn nicht beschleunigt und nicht gebremst wird, dann muss die Beschleunigung erst langsam sein bis `K.I.T.T.` stehenbleibt, aufgrund der Gleitreibung zwischen Reifen und Asphalt.

(i) AG(¬*brake* ∧ ¬*accelerate* ∧ *state*=`spm` → AX *state*=`slow`)
(ii) AG(¬*brake* ∧ ¬*accelerate* ∧ *state*=`slow` → AX *state*=`stop`)

Einer Ihrer Arbeitskollegen hat bereits eine Statemachine dazu gezeichnet die er später gerne in Software implementieren möchte. Stolz präsentiert er Ihnen seine Überlegungen:



Sein Automat hat insgesamt drei Zustände:

- *stop*: `K.I.T.T.` steht.
- *slow*: `K.I.T.T.` beschleunigt langsam.
- *spm*: `K.I.T.T.` beschleunigt schnell und befindet sich im `SUPER PURSUIT MODE`.

Sie sprechen Ihren Kollegen auf die Thematik formaler Verifikation an und ob er davon schon einmal was gehört hätte. Er verneint und meint für die paar Zustände brauchen Sie keine aufwendige Verifikation. Er ist sich sicher, dass alles stimmt – er hat schließlich Erfahrung. Formale Verifikation haben wir noch nie verwendet, brauchen wir auch nicht. Meint er zumindest. Das wollen Sie so nicht auf sich sitzen lassen ...

Zu lösende Aufgaben:

1. Modellieren Sie das K.I.T.T. Steuergerät das Ihr Kollege entworfen hat in NuSMV.
2. Wie viele verschiedene Zustände hat das Systemmodell? Zeigen Sie wie diese Anzahl erreicht wird. Überlegen Sie zunächst selbst und überprüfen Sie Ihr Ergebnis mit NuSMV.
3. Verifizieren Sie das Modell gegen die von Ihnen aufgestellten CTL Eigenschaften.
4. Offensichtlich hat die formale Verifikation gegen die Erfahrung Ihres Kollegens gesiegt. Interpretieren Sie das erzeugte Gegenbeispiel und überlegen Sie wie man das Modell richtigstellen kann und wie Sie Ihren Kollegen den Fehler in seinem Modell anhand des Gegenbeispiels erklären.
5. Überprüfen Sie das von Ihnen verbesserte Modell und überprüfen Sie nochmals die Spezifikation.

Lösungsvorschläge:

1. Die Modellierung des K.I.T.T. Automatens Ihres Kollegens in NuSMV:

```
1  MODULE main
2  VAR
3    accelerate : boolean;
4    brake : boolean;
5    state : {stop, slow, spm};
6
7  ASSIGN
8    init(state) := stop;
9    next(state) := case
10         accelerate & !brake & state = stop : slow;
11         accelerate & !brake & state = slow : spm;
12        !accelerate & !brake & state = slow : stop;
13
14        brake : stop;
15        TRUE  : state;
16 esac;
```

2. Der gesamte Zustandsraum berechnet sich wie folgt: Das System hat zwei Bool'sche Events (Variablen). Jede davon kann entweder 0 oder 1 sein. Insgesamt gibt es drei Zustände in denen sich die Statemachine befinden kann, daher ist die Gesamtanzahl der Zustände: Kombination der Events (Variablen) $\times$ Anzahl der Zustände des Automaten $= 2^2 \times 3 = 12$, oder mit NuSMV:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV -int kitt.smv
NuSMV> go
NuSMV> print_reachable_states
######################################
system diameter: 3
reachable states: 12 (2^3.58496) out of 12 (2^3.58496)
######################################
NuSMV>
```

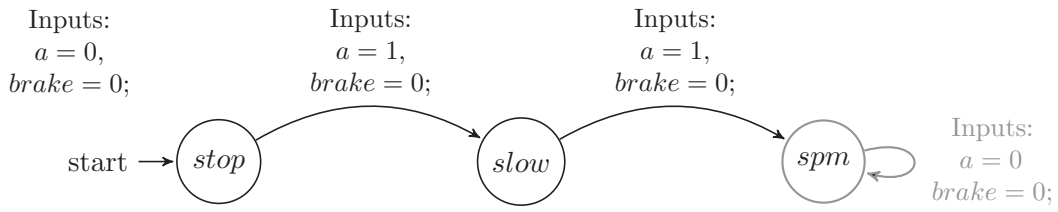
3. Verifikation des K.I.T.T. Modells Ihres Kollegen durch Ihre CTL Spezifikation: Zuerst wird die CTL Spezifikation in das Systemmodell eingefügt mit dem SPEC Schlüsselwort:

```
1   MODULE main
2   SPEC AG(brake -> AX state=stop)
3   SPEC AG(!brake &  accelerate & state=stop -> AX state=slow)
4   SPEC AG(!brake &  accelerate & state=slow -> AX state=spm)
5   SPEC AG(!brake & !accelerate & state=spm  -> AX state=slow)
6   SPEC AG(!brake & !accelerate & state=slow -> AX state=stop)
7   VAR
8     accelerate : boolean;
9     brake : boolean;
10    state : {stop, slow, spm};
11
12  ASSIGN
13    init(state) := stop;
14    next(state) := case
15         accelerate & !brake & state = stop : slow;
16         accelerate & !brake & state = slow : spm;
17        !accelerate & !brake & state = slow : stop;
18
19        brake : stop;
20        TRUE : state;
21  esac;
```

Mithilfe SPEC Anweisungen direkt im NuSMV Modellcode können Sie nun die Verifikation starten, die zu folgendem Resultat führt:

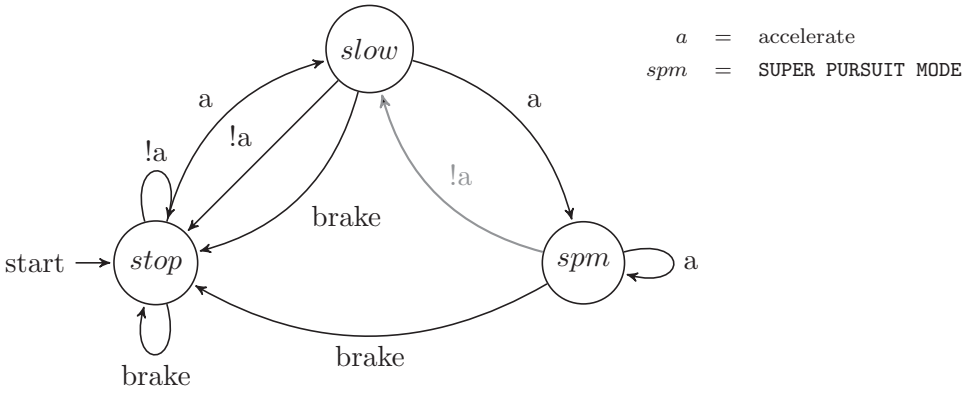
```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV kitt.smv
-- spec AG (brake -> AX state = stop) is true
-- spec AG (((!brake & accelerate) & state = stop) -> AX state = slow) is true
-- spec AG (((!brake & accelerate) & state = slow) -> AX state = spm) is true
-- spec AG (((!brake & !accelerate) & state = spm) -> AX state = slow) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
accelerate = 0
brake = 0
state = stop
-> Input: 1.2 <-
-> State: 1.2 <-
accelerate = 1
-> Input: 1.3 <-
-> State: 1.3 <-
state = slow
-> Input: 1.4 <-
-- Loop starts here
-> State: 1.4 <-
accelerate = 0
state = spm
-> Input: 1.5 <-
-> State: 1.5 <-
-- spec AG (((!brake & !accelerate) & state = slow) -> AX state = stop) is true
```

Betrachtet man das Gegenbeispiel grafisch, so wird folgendes ersichtlich: Immer dann, wenn sich `K.I.T.T.` im `SPM` befindet und die Beschleunigung von 1 auf 0 übergeht gibt es keine Möglichkeit wieder in den vorherigen Zustand *slow* überzugehen. Das bedeutet, wenn Michael nur vom Gaspedal geht und nicht bremst wird `K.I.T.T.` immer schneller werden.



Das Modell wäre somit nur dann richtig, wenn es keine Reibung zwischen den Reifen und der Straße gibt (oder sich `K.I.T.T.` im freien Fall befindet). Das bedeutet, wenn

sich K.I.T.T. im SPM befindet und nicht beschleunigt und nicht gebremst wird muss es eine Transition (Übergang) im Modell vom Zustand *spm* in den Zustand *slow* geben. Sie zeichnen nun diese zusätzliche Transiton (!a) in das Modell ein:



4. Verifikation des richtiggestellten K.I.T.T. Modells durch Ihre CTL Spezifikation: Das bestehende Modell wird durch die Zeile 17 erweitert, die die zusätzliche Transition vom Zustand *spm* in den Zustand *slow* darstellt:

```
1   MODULE main
2   SPEC AG(brake -> AX state=stop)
3   SPEC AG(!brake &  accelerate & state=stop -> AX state=slow)
4   SPEC AG(!brake &  accelerate & state=slow -> AX state=spm)
5   SPEC AG(!brake & !accelerate & state=spm  -> AX state=slow)
6   SPEC AG(!brake & !accelerate & state=slow -> AX state=stop)
7   VAR
8     accelerate : boolean;
9     brake : boolean;
10    state : {stop, slow, spm};
11
12  ASSIGN
13    init(state) := stop;
14    next(state) := case
15         accelerate & !brake & state = stop : slow;
16         accelerate & !brake & state = slow : spm;
17        !accelerate & !brake & state = spm  : slow;
18        !accelerate & !brake & state = slow : stop;
19
20        brake : stop;
21        TRUE : state;
22  esac;
```

Die Verifikation mit NuSMV ergibt folgendes Resultat:

```
thomas@tianShi:~/NuSMV-2.4.3-i686-pc-linux-gnu/bin> ./NuSMV kitt.smv
-- spec AG (brake -> AX state = stop) is true
-- spec AG (((!brake & accelerate) & state = stop) -> AX state = slow) is true
-- spec AG (((!brake & accelerate) & state = slow) -> AX state = spm) is true
-- spec AG (((!brake & !accelerate) & state = spm) -> AX state = slow) is true
-- spec AG (((!brake & !accelerate) & state = slow) -> AX state = stop) is true
```

Anmerkungen, Erläuterungen und Ergänzungen zum Lerninhalt:

Das Knight Rider Beispiel ist von dem Car Controller (CC) Beispiel in dem Paper „Testing with model checkers: A survey“ [101] inspiriert. Interessierte Lehrende können aus diesem Paper nach Belieben weitere Beispiele ableiten und für den eigenen Unterricht aufbereiten.

6.8 Leitfaden zum konkreten Einsatz der Unterrichtsbeispiele im Unterricht

Die in dieser Arbeit gezeigten Unterrichtsbeispiele sind bewusst sehr generisch gehalten. Sie sind nicht als reine Unterrichtsbeispiele zum Abprüfen von Fachinhalten zu verstehen, sie sind in einem universielleren Kontext zu sehen. Einerseits liegt ihnen das Kompetenzmodell des bm:ukk zugrunde, andererseits versuchen sie den Lernenden einen Einstieg in den Themenkreis (formale) Verifikation von Software zu ermöglichen.

Neben einer Definition von Kompetenzen für die Ausbildung im Bereich der (formalen) Verifikation von Software sind die vorgestellten Lehrblöcke auch verwendbar als:

Projektaufgabenstellung: Die Blöcke I – VI eignen sich als Projektaufgabenstellung, die von Teams zu zwei bis drei Lernenden bearbeitet werden können. Der Abschlussblock VII kann für eine Wiederholung der Lehrinhalte von jedem einzelnen Lernenden herangezogen werden, oder aber auch als Prüfungsangabe. Durch die, in die Aufgabenstellung verpackte, breite Einführung in die Thematik ist die Duchführung eines solchen Projekts mit minimaler Supervision des Lehrenden möglich. Es kann hierbei auf die Eigendynamik der Lerngruppe gesetzt werden.

Lehrinhalte für den Frontalunterricht: Durch die aufeinander aufbauende, kumulitive Auslegung der Unterrichtsblöcke eignen sich diese besonders für den Frontalunterricht für ca. sechs bis acht Doppelstundeneinheiten. Durch die angeführte Begabtenförderung kann der Lehrende, bestimmte Lehrinhalte vertiefen, andere dafür zum Beispiel nur in ihrer elementarsten Form anwenden. Es ist anzumerken, dass der Fron-

talunterricht unbedingt mit begleitenden Übungen zu kombinieren ist. Modellierung und Verifikation sind Fähigkeiten die großteils Erfahrung und Übung benötigen.

Portfolio: Strebt der Lehrende den Einsatz von Portfolio Unterricht an, so geben ihm die ausgearbeiteten Unterrichtsblöcke einen Fahrplan für die Vermittlung der (formalen) Software- und Systemverifikation.

Gruppenpuzzle: Einzelne Fachgebiete und Fragestellungen der Unterrichtsreihe können auch in der Form eines Gruppenpuzzles ausgearbeitet werden. Mögliche Fragestellungen für das Gruppenpuzzles können sein:

- Erarbeiten der Bool'schen Operatoren
- Erarbeiten der temporalen Operatoren
- Erarbeiten einer weiteren temporalen Logik (z.b. LTL, CTL$*$)
- Lösen von Logikrätsel mit Model Checker
- Gegenseitige Vorstellung von weiteren Verifikationstools

Den Unterricht begleitendes Skriptum: Die Blöcke I – VII eignen sich für ein begleitendes Skriptum, dass den Unterricht unterstützt.

Prüfungsaufgaben: Besonders Block VII eignet sich als die Thematik umfassendes Prüfungsbeispiel.

7 Conclusio

> The reverse side also has a reverse side.
>
> *(Japanisches Sprichwort)*

7.1 Zusammenfassung

In der vorliegenden Arbeit wurde eine erste konkrete Anwendung des Kompetenzmodell des bm:ukk auf die Lehre von (formaler) Verifikation von Software und Systeme für die berufsbildende höhere Schule (HTL) erarbeitet. Didaktische Grundsätze, Möglichkeiten, Absichten und Potenziale von Bildungsstandards wurden erläutert und dargestellt.

Die Basisidee der Arbeit, formale Verifikation von Software und Systeme zum konkreten Unterrichtsinhalt zu machen, wurde durch qualitative Interviews motiviert. Diese Interviews stellen einerseits den IST Zustand in der Schule (HTL Hollabrunn) sowie den SOLL Zustand (Anforderungen der Industrie an künftige HTL Absolventen) stichprobenartig dar. Aus den Interviews lassen sich folgende Kernaussagen ableiten:

1. Formale Software- und Systemverifikation sind gegenwärtig keine Inhalte des Informatikunterrichts.
2. Der Fokus in der Schule liegt weiterhin auf dem „Programmieren im Kleinen“ Prinzip.
3. In der Industrie sind Test und Verifikation mindestens genauso wichtig wie das eigentliche Implementieren.
4. Für sicherheitskritische Applikationen ist der Anteil an Testen und Verifikation >50%.
5. (Formale) Software- und Systemverifikation ist Teil des „Alltagsgeschäfts“ in der Industrie.
6. Um den Puls der Industrie zu folgen, ist es unumgänglich formale Software- und Systemverifikation zum Unterrichtsinhalt zu machen.
7. Die Industrie wünscht sich Absolventen mit Fähigkeiten im Themenkreis (formale) Verifikation und Test.

8. Die Industrie würde sich für die Ausbildung eine Zusammenarbeit mit bestehenden Institutionen wie dem International Software Testing Qualifications Board (ISTQB) wünschen.

9. Für die Industrie scheint eine internationale Vergleichbarkeit von erworbenen Qualifikationen eher uninteressant, weil Qualifikationen anhand von Erfahrung und „Bauchgefühl“ bewertet werden.

10. Für die Industrie sind sprachliche Kompetenzen (vor allem Englisch) neben einer fundierten technischen Ausbildung besonders wertvoll.

11. HTL Absolventen mit entsprechender Arbeitserfahrung nehmen Schlüsselpositionen in Unternehmen ein.

Um den Ungleichgewichtszustand zwischen Lehrinhalten und Anforderungen der Industrie entgegenzuwirken, präsentiert diese Arbeit sieben Unterrichtsblöcke die einerseits in das Kompetenzmodell eingeordnet sind, anderseits auch völlig frei von diesem für den alltäglichen Unterricht angewendet werden können. Die Blöcke sind aufeinander aufbauend und bieten großzügige Zusatzinformationen und Tipps zur Begabtenförderung. Ein abschließender Block fasst die vorhergehenden Lehrinhalte zusammen und rundet das Themengebiet inhaltlich ab. Die Unterrichtsblöcke arbeiten mit dem von Randy Pausch geprägten Begriff „Headfake“ – *The best way to teach somebody something is to have them think they're learning something else.* Das Konzept des „Headfakes“ findet sich in vielerlei Formen wieder. Die gelernten Fähigkeiten bzgl. Modellierung der Kaffeemaschine mögen teilweise trivial erscheinen, doch TCP/IP controller, H.264 decoder, Steuergeräte, ASICs usw. werden in der Industrie auf ähnliche Art und Weise modelliert und verifiziert.

Die Kompetenzmodelle stellen gegenwärtig eine Beispielsammlung dar, mit der versucht werden soll, Kompetenzen von HTL Absolventen zu dokumentieren. Es ist vermutungsweise noch ein langer Weg bis aus den Kompetenzmodellen standardisierte Unterrichts- und Prüfungsbeispiele abgeleitet werden können. Für den HTL Schultyp gilt es besonders die individuelle Autonomie und Ausbildungsschwerpunkte zu sichern, da diese eine der wesentlichen Schlüsselfaktoren zum Erfolg der berufsbildenden höheren Schulen darstellen.

Hinsichtlich der Kompetenzmodelle stellt die Arbeit klar, dass eine Absprache mit der Industrie für die Ausarbeitung der Referenzbeispielsammlung notwendig ist, um den hohen Ausbildungsstandard der HTLs langfristig aufrechtzuerhalten und Bildung weiterhin auf europäischen Top Niveau zu betreiben.

7.2 Betrachtung der Hypothesen

Die in Kapitel 1 aufgestellten Hypothesen werden durch die Beiträge der Arbeit wie folgt beurteilt:

7.2.1 Hypothese (H.a)

✓	*(H.a)* **Der Informatikunterricht bedarf einer Verbesserung** *und ist aktuell nicht im perfekten Einklang mit den Anforderungen der Industrie an die Absolventen, es bedarf einer Änderung der Unterrichtsschwerpunkte.*

Die qualitativen Interviews in Kapitel 3 und 4 haben eindeutig aufgezeigt, dass der gegenwärtige Informatikunterricht (zumindest in der befragten HTL) nicht im Einklang mit den Erwartungen der Industrie steht. Eine Änderung der Unterrichtschwerpunkte in Richtung Verifikation und Test von Software ist wünschenswert.

7.2.2 Hypothese (H.b) und Hypothese (H.c)

✓	*(H.b)* **Eine Anpassung des Informatikunterrichts** *mit thematischer Ausrichtung auf einen industrienahen Softwareentwicklungsprozess (Entwickeln, Testen, Verifizieren)* **ist machbar** *und stellt einen signifikanten Qualitätszuwachs der Ausbildung dar.*

✓	*(H.c)* **Das Kompetenzmodell des bm:ukk ist eine Bereicherung** *für den standardisierten Unterricht. Es ist möglich den bestehenden Kompetenzmodell Entwurf mit Beispielen rund um die formale Softwareverifikation zu erweitern.*

In dieser Arbeit wurde das Kompetenzmodell erstmals auf die Unterrichtsinhalte (formale) Verifikation von Software und Systeme angewandt. Durch eine der Altersgruppe entsprechenden Aufbereitung der Fachinhalte wird auch Schülern der HTL der Zugang zum Thema ermöglicht. Es wurde gezeigt, dass durch eine Ausrichtung des Unterrichts auf die Anforderungen der Industrie eine beidseitige Win-Win Situation erreicht werden kann.

Für das Kompetenzmodell bleibt es zu wünschen, dass die darin definierten Fähigkeiten in naher Zukunft auch in konkrete, praktisch anwendbare Unterrichtsbeispiele münden.

Literaturverzeichnis

[1] J. L. Lions, "ARIANE 5 flight 501 failure report," July 1996, visited: April 2010. [Online]. Available: http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html

[2] M. I. Board, "Mars climate orbiter - phase I report," November 1999, visited: April 2010. [Online]. Available: ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf

[3] NYISO, "Interim report on the August 14, 2003 blackout," Januar 2004, visited: April 2010. [Online]. Available: http://www.hks.harvard.edu/hepg/Papers/NYISO.blackout.report.8.Jan.04.pdf

[4] M. Kanellos, "Software glitch stalls some Toyota hybrids," October 2005, visited: April 2010. [Online]. Available: http://news.cnet.com/Software-glitches-stalls-some-Toyota-hybrids/2100-11389_3-5895574.html

[5] D. Gainer, "Microsoft Excel calculation issue update," September 2007, visited: April 2010. [Online]. Available: http://blogs.msdn.com/excel/archive/2007/09/25/calculation-issue-update.aspx

[6] APA, "A1-Netzausfall: Mobilkom gibt Entwarnung," October 2008, visited: April 2010. [Online]. Available: http://futurezone.orf.at/stories/317646/

[7] E. S. Raymond, *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly Media, 1999. [Online]. Available: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

[8] G. J. Myers, *Art of Software Testing.* New York, NY, USA: John Wiley & Sons, Inc., 1979, ISBN: 0471043281.

[9] D. Peled, *Software Reliability Methods.* Springer, 2001, ISBN 0387951067.

[10] M. Weissenböck, *From Computer Literacy to Informatics Fundamentals*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, ch. Informatics Education at Vocational Schools and Colleges in Austria, pp. 32–36, ISBN: 9783540253365.

[11] W. Arnhold, "Lieben Sie PHYTON?" *LOG IN*, vol. 21, pp. 18–24, 2001.

[12] R. Baumann, "Java - Stimulans für den Informatikunterricht," *LOG IN*, vol. 17, pp. 20–31, 1997.

[13] I. Linkweiler and L. Humbert, "Ergebnisse der Untersuchung zur Eignung einer Programmiersprache für die schnelle Softwareentwicklung – kann der Informatikunterricht davon profitieren?" September 2002. [Online]. Available: http://www.ham.nw.schule.de/pub/bscw.cgi/d23460/24_September_2002_Linkweiler.pdf

[14] A. Schwill, "Programmiersprachen im Unterricht," 1998, visited: March 2010. [Online]. Available: http://www.informatikdidaktik.de/Forschung/Schriften/PSimUnterrMatNatTag.pdf

[15] P. Balog, "Embedded Systems - Ausbildung an der Fachhochschule Technikum Wien," in *Informationstagung Mikroelektronik ME01, Themenkreis: Zukunft der Schulinformatik*, Wien,, October 2001, pp. 57 – 61, ISBN: 3851330226. [Online]. Available: http://embsys.technikum-wien.at/staff/balog/documents/conference-papers/me01_fb-embsys.pdf

[16] K. Zeillinger, "Fehler im System," *Auto Touring, Das ÖAMTC Magazin*, visited: April 2010. [Online]. Available: http://www.autotouring.at/archiv/2006/06/28677.html

[17] G. J. Holzmann, "Software safety in rocket science," *ERCM News Special: Safety-Critical Software*, vol. 75, pp. 14–15, October 2008.

[18] HTL Innovativ, "Internetplattform für Diplom- und Abschlussarbeiten an technisch-(kunst)gewerblichen Lehranstalten," visited: April 2010. [Online]. Available: http://www.htl-innovativ.at/

[19] IEC, "Standard IEC 61508-3 – Functional safety of electrical/electronic/programmable electronic safety-related systesms, int'l electrotechnical safety-related systems," 1997, Int'l Electrotechnical Commission.

[20] ESA Board for Software Standardisation and Control (BSSC), "ESA software engineering standards issue 2," 1994. [Online]. Available: ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/PSS050.pdf

[21] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008, ISBN 026202649X.

[22] A. Prein, "Didaktische Entwürfe zum Pflichtgegenstand "Angewandte Programmierung" des ersten Jahrganges einer HTL für Informationstechnologie," Master's thesis, Technical University of Vienna (Institut für Rechnergestützte Automation), 2008.

[23] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 1999, ISBN 0262032708.

[24] T. Reinbacher, "Model checking and static analysis of MCS-51 assembly code," Master's thesis, Department of Embedded Systems, University of Applied Sciences Technikum Wien, June 2009.

[25] J. Börstler, "Objektorientiertes Programmieren - Machen wir irgendwas falsch?" in *12. GI-Fachtagung Informatik und Schule (INFOS '07)*, Siegen, September 2007, pp. 9–20.

[26] U. Fritz, "Bildungsstandards in der Berufsbildung – Projekthandbuch," Herausgeber: Bundesministerium für Unterricht, Kunst und Kultur, Sektion II: Berufsbildung, Visited: April 2010. [Online]. Available: http://www.bildungsstandards.berufsbildendeschulen.at/fileadmin/content/bbs/Handbuch_BIST_Februar2010.pdf

[27] H. Binder, "Bildungsstandard Elektrotechnik – Höhere Technische Lehranstalt," Version November 2008, Visited: March 2010. [Online]. Available: http://www.bildungsstandards.berufsbildendeschulen.at/

[28] S. Niemeyer and B. Wess, "Bildungsstandard Elektrotechnik – Höhere Technische Lehranstalt," Version 1.0, Visited: January 2010. [Online]. Available: http://www.bildungsstandards.berufsbildendeschulen.at/

[29] A. Reiter, "20 Jahre Schulinformatik in Österreich und IKT-Einsatz im Unterricht," Februar 2005. [Online]. Available: http://www.gym1.at/schulinformatik/buecher/schulinformatik2005.pdf

[30] S. Schubert, "Stand der Fachdidaktik Informatik," in *Fachtagung Informatik*, Hamburg, 2001, pp. 1 – 7. [Online]. Available: http://www.gym1.at/informatik/arge/03102002/schubert-gedankenzuminfounterricht.pdf

[31] Österreichische Computer Gesellschaft, "Website des Europäischen Computer Führerscheins," visited: April 2010. [Online]. Available: http://www.ecdl.at/

[32] G. Futschek, "Das ECDL-Zertifikat im Bildungswesen," in *Wissen und Lernen - Was trägt die Informatik zum Unterricht bei?* Wien: Österreichische Computer Gesellschaft, 2001, pp. 51–56, eingeladen.

[33] G. Hager, "Das Cisco Networking Academy Program und dessen Umsetzung in der Schule," in *Schulinformatik in Österreich*. Wien: Ueberreuter, 2003, pp. 363–368.

[34] G. Futschek, "Neue Impulse für den Informatikunterricht durch das universitäre Lehramtstudium Unterrichtsfach Informatik und Informatikmanagement," in *Schulinformatik in Österreich*. Wien: Ueberreuter, 2003, pp. 377–382, eingeladen.

[35] "Website des bm:ukk," visited: March 2010. [Online]. Available: http://www.bmukk.gv.at/

[36] M. Weissenböck, "Informatik an berufsbildenden Schulen," in *Schulinformatik in Österreich*. Wien: Ueberreuter, 2003, pp. 347–350.

[37] R. Radinger and M. Schwabe, "Entwicklung der Schülerzahlen 2004/2005," *Statistische Nachrichten der Statistik Austria*, Februar 2006. [Online]. Available: http://www.statistik.at/web_de/static/entwicklung_der_schuelerzahlen_200405_035590.pdf

[38] Europäisches Parlament und der Rat der Europäischen Union, "Richtlinie 2005/36/EG des Europäischen Parlaments und des Rates über die Anerkennung von Berufsqualifikationen," September 2005, Amtsblatt der Europäischen Union. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2005:255:0022:0142:DE:PDF

[39] "Website der HTL Hollabrunn," visited: March 2010. [Online]. Available: http://www.htl-hl.ac.at/

[40] "Website der FH Technikum Wien," visited: March 2010. [Online]. Available: http://www.technikum-wien.at/

[41] P. Hubwieser, *Didaktik der Informatik: Grundlagen, Konzepte, Beispiele*. Springer, Berlin, 2007, vol. 3, ISBN: 354072477X.

[42] W. Klafki, *Das pädagogische Problem des Elementaren und die Theorie der kategorialen Bildung*. Weinheim: Beltz Verlag, 1964, ISBN: 3407106068.

[43] P. Heimann, *Didaktik als Unterrichtswissenschaft*. Stuttgart: Klett, 1976, ISBN: 3129233806.

[44] J. Kühn, *Das "Berliner Modell" von Paul Heimann*. Grin Verlag, 2007, ISBN: 3638754197.

[45] W. Hartmann, M. Näf, and R. Reichert, *Informatikunterricht planen und durchführen*. Springer-Verlag Berlin Heidelberg, 2006, ISBN: 978-3-540-34484-1.

[46] W. Schulz, *Didaktische Einblicke: "Das Gesicht der Schule gestalten"*. Weinheim, 1995, ISBN: 3407251637.

[47] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948. [Online]. Available: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

[48] H. G. Frank and B. S. Meder, *Einführung in die kybernetische Pädagogik*, 1971, ISBN: 3423041080.

[49] F. v. Cube, *Kybernetische Grundlagen des Lernens und Lehrens*. Klett-Cotta (Stuttgart), 1982, ISBN: 3129217703.

[50] K. H. Schäfer and K. Schaller, *Kritische Erziehungswissenschaft und Kommunikative Didaktik*. Quelle und Meyer (Heidelberg), 1971, ISBN: 3494020019.

[51] S. Schubert and A. Schwill, *Didaktik der Informatik*. Spektrum Akademischer Verlag, 2004, vol. 1, ISBN: 3827413826.

[52] S. Jähnichen, M. Fothe, and S. Friedrich, "Produktwissen oder Konzeptwissen?" *DLGI Magazin*, vol. Oktober, pp. 14 – 15, 2008.

[53] "The fist computer bug," 1945, visited: March 2010. [Online]. Available: http://www.history.navy.mil/photos/images/h96000/h96566k.jpg

[54] F. S. Foundation, "Debugging with gdb," visited: April 2010. [Online]. Available: http://sourceware.org/gdb/current/onlinedocs/

[55] D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young, "Computing as a discipline," *Communications of the ACM*, vol. 32, no. 1, pp. 9–23, 1989.

[56] P. J. Denning, "Great principles of computing," *Communications of the ACM*, vol. 46, pp. 15–20, November 2003.

[57] P. J. Denning and C. Martell, "Great principles of computing web site," 02 2010.

[58] P. J. Denning, "Computing is a natural science," *Communications of the ACM*, vol. 50, no. 7, pp. 13–18, 2007.

[59] W. Einsiedler, *Lehrmethoden*. Urban & Schwarzenberg, 1981, ISBN: 3541414111.

[60] K. Lewin, G. W. Lewin, and H. A. Frenzel, *Die Lösung sozialer Konflikte*, 1953, ASIN: B0000BKYTQ.

[61] C. Pauli and K. Reusser, "Zur Rolle der Lehrperson beim kooperativen Lernen," *Schweizerische Zeitschrift für Bildungswissenschaften*, vol. 22(3), pp. 421–442, 2000. [Online]. Available: http://www.didac.uzh.ch/public/Publikationen/2000/Pauli_Reusser_Kooperatives_Lernen.pdf

[62] R. T. Johnson and D. W. Johnson, "An overview of cooperative learning," *Creativity and Collaborative Learning: A Practical Guide to Empowering Students and Teacher*, pp. 31—44, 1994.

[63] D. W. Johnson and R. T. Johnson, *Learning Together and Alone: Cooperative, Competitive, and Individualistic Learning*. Allyn & Bacon, 1998, ISBN: 0205287719.

[64] "Website Kooperatives Lernen," visited: March 2010. [Online]. Available: http://www.kooperatives-lernen.de

[65] M. Schmidt-Klingenberg, "Unter Druck nach oben," *Spiegel Special*, no. 3, pp. 106–111, 2002.

[66] W. Stangl, "Der Begriff der sozialen Kompetenz in der psychologischen Literatur," June 2002, Version 2.2, Visited: March 2010. [Online]. Available: http://paedpsych.jk.uni-linz.ac.at/PAEDPSYCH/SOZIALEKOMPETENZ/

[67] J. Ripplinger, "Lernziel Sozialkompetenz – Wie Schulen soziales Lernen systematisch fördern können," 2009, Argentur Mehrwert. [Online]. Available: http://www.agentur-mehrwert.de/pdf/Fachartikel_Lernziel_Sozialkompetenz.pdf

[68] E. G. Cohen, "Restructuring the classroom: Conditions for productive small groups," *Review of Educational Research*, vol. 64, no. 1, pp. 1–35, 1994.

[69] W. Sensink, *Fachdidaktik Wirtschaftswissenschaft*, 1st ed. Oldenbourg, 1994, ISBN: 3486226053.

[70] R. Loska, *Lehren ohne Belehrung. Leonard Nelsons neosokratische Methode der Gesprächsführung*. Bad Heilbrunn: Klinkhardt, 1995, ISBN: 3781507904.

[71] E. Meyer, *Gruppenunterricht: Grundlegung und Beispiel*. Schneider Verlag Hohengehren, 1996, vol. 9. A, ISBN: 387116853X.

[72] M. Csongrady, E. Geretschläger, C. Kimbacher, S. Nöbauer, and K. Reiter, "Die ARGE Didaktik und Methodik der HTL-Steyr erprobt und evaluiert den Puzzleunterricht als Alternative Unterrichtsmethode," pp. 1–3, July 2006, visited: April 2010. [Online]. Available: http://imst3plus.uni-klu.ac.at/materialien/2006/244_Langfassung_Geretschlaeger.pdf

[73] E. Aronson and S. Patnoe, *The Jigsaw Classroom: Building Cooperation in the Classroom*. Longman, 1997, ISBN: 0673993833.

[74] W. Emer and K.-D. Lenzen, "Methodenlernen als Zugang zum Projektunterricht. Projekteigene und projektnahe Methoden im Überblick," *Pädagogik*, vol. 1, pp. 16–19, 2007.

[75] K. Frey, *Die Projektmethode: Der Weg zum bildenden Tun*. Beltz, Weinheim, 2005, ISBN: 3407254679.

[76] R. Fabianski, "Möglichkeiten der Projektteambildung," 2003. [Online]. Available: http://www.fabianski.de/infoprojekt/projektteam.php

[77] V. Klenowski, *Developing Portfolios for Learning and Assessment: Processes and Principles*. Routledge, 2002, ISBN: 075070988X.

[78] K. Reich, "Portfolio im Methodenpool," visited: March 2010. [Online]. Available: http://methodenpool.uni-koeln.de/download/portfolio.pdf

[79] ——, "Website: Unterrichtsmethoden im konstruktiven und systemischen Methodenpool," visited: March 2010. [Online]. Available: http://methodenpool.uni-koeln.de/

[80] "Lehrplan Höhere Lehranstalt für Elektrotechnik – Informationstechnik, v. 2004," visited: March 2010. [Online]. Available: http://www.htl.at/fileadmin/content/Lehrplan/HTL_Elektrotechnik_Informationstechnik.pdf

[81] K. J. Fuchs, *Fachdidaktische Studien.* Aachen: Shaker Verlag, 2007, ISBN: 9783832265939.

[82] W. Schwendenwein, *Theorie des Unterrichtens und Prüfens.* facultas wuv universitätsverlag, 2000, vol. 7, ISBN: 3851145046.

[83] E. Kommission, "Der europäische Qualifikationsrahmen für lebenslanges Lernen (EQR)," 2008, ISBN: 9789279084720. [Online]. Available: http://ec.europa.eu/education/policies/educ/eqf/eqf08_de.pdf

[84] M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, *Taxonomie von Lernzielen im kognitiven Bereich*, 16th ed., B. S. Bloom, Ed. Beltz Verlag, 1972, no. 35, ISBN: 3407182961.

[85] F. E. Weinert, *Concept of competence: A conceptual clarification.* Göttingen: Hogrefe & Huber, 2001, ch. Defining and selecting key competencies, pp. 45–65.

[86] "Website der Bildungsstandards des bm:ukk," visited: March 2010. [Online]. Available: http://www.bildungsstandards.berufsbildendeschulen.at/

[87] S. Friedrich and H. Puhlmann, "Bildungsstandards Informatik - von Wünschen zu Maßstäben für eine informatische Bildung," in *12. GI-Fachtagung Informatik und Schule (INFOS '07)*, September 2007, pp. 21–32.

[88] F. Nielson, H. Nielson, and C. Hankin, *Principles of Program Analysis.* Springer, 2004, ISBN 3540654100.

[89] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems.* New York, NY, USA: Springer-Verlag New York, Inc., 1992.

[90] A. Pnueli, "The temporal logic of programs," *Foundations of Computer Science, Annual IEEE Symposium on Foundations of Computer Science (FOCS '77)*, pp. 46–57, 1977.

[91] L. Lamport, "Sometime is sometimes not never: on the temporal logic of programs," in *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '80)*, Las Vegas, USA, 1980, pp. 174–185.

[92] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, ser. LNCS, vol. 131, 1981, pp. 52–71.

[93] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, 1986.

[94] R. Golecki and J. Jungmann, "Einführung in die Aussagenlogik," online, August 1990, visited: March 2010. [Online]. Available: http://lbs.hh.schule.de/oberstufe/ki-al.pdf

[95] A. Pnueli and Z. Manna, *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag Gmbh, 1991, ISBN 0387976647.

[96] E. M. Clarke, E. A. Emerson, and J. Sifakis, "Model checking: algorithmic verification and debugging," *Commun. ACM*, vol. 52, no. 11, pp. 74–84, 2009, Turing Award lecture.

[97] D. Gries, *The Science of Programming.* Springer-Verlag New York, Inc., 1983, ISBN: 0-387-90641-X.

[98] T. Reinbacher, M. Horauer, B. Schlich, J. Brauer, and F. Scheuer, "Model checking assembly code of an industrial knitting machine," in *Proceedings of the 4th IEEE International Conference on Embedded and Multimedia Computing (EM-Com '09)*, Jeju, Korea, December 10-12 2009, pp. 97–104.

[99] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: a new Symbolic Model Verifier," in *Proceedings of the Eleventh Conference on Computer-Aided Verification (CAV '99)*, ser. Lecture Notes in Computer Science, N. Halbwachs and D. Peled, Eds., no. 1633. Trento, Italy: Springer, July 1999, pp. 495–499.

[100] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking," in *Proceedings of the International Conference on Computer-Aided Verification (CAV '02)*, ser. LNCS, vol. 2404. Copenhagen, Denmark: Springer, July 2002.

[101] G. Fraser, F. Wotawa, and P. E. Ammann, "Testing with model checkers: a survey," *Software Testing, Verification & Reliability*, vol. 19, no. 3, pp. 215–261, 2009.

Abbildungsverzeichnis

Tabellenverzeichnis