Entwicklung und Erprobung eines Instruments zur Messung informatischer Modellierungskompetenz im fachdidaktischen Kontext

Thomas Rhode

Paderborn, 28.02.2013

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
im Fachbereich Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

Gutachter:
Prof. Dr. Johann S. Magenheim
Prof. Dr. Reinhard Keil
Prof. Dr. Sigrid Schubert

Zusammenfassung

Die Auswertung der Ergebnisse internationaler Vergleichsstudien haben Mängel am deutschen Bildungssystem aufgedeckt. In diesem Zusammenhang rückte die Vermittlung von Kompetenzen in das Zentrum bildungspolitischer Lösungsstrategien. Um die Qualität des deutschen Bildungssystems zu sichern, wird die Entwicklung nationaler Bildungsstandards gefordert, die verbindliche Anforderungen an das Lehren und Lernen in der Schule darstellen. Diese Bildungsstandards sollen sich nach Vorgaben des Bildungsministeriums für Bildung und Forschung auf wissenschaftlich fundierte Kompetenzmodelle stützen, die in Zusammenarbeit von Fachdidaktikern, Fachwissenschaftlern und Psychologen entwickelt werden. Ebenso wird hier explizit die Einbeziehung und Entwicklung entsprechender Verfahren zur Testentwicklung gefordert.

In Anbetracht dieser bildungspolitischen Umstände, beschreibt die vorliegende Dissertationsschrift die Entwicklung eines wissenschaftlich fundierten Kompetenzstrukturmodells und eines Kompetenzmessinstruments für einen zentralen Teilbereich der informatischen Bildung, der objektorientierten Modellierung.

Um die Bedeutung des Gegenstandsbereichs zu verdeutlichen, wird die Relevanz der Modellierung für die Informatik aus fachwissenschaftlicher und fachdidaktischer Perspektive erörtert. Im weiteren Verlauf wird dargestellt, wie die Gestaltung eines Kompetenzstrukturmodells in zwei Schritten erfolgen kann. Zunächst erfolgt eine normativ theoretische Ableitung von Kompetenzdimensionen und -komponenten anhand von einschlägiger Fachliteratur. In einem weiteren Schritt findet die empirische Verfeinerung des theoretischen Rahmenmodells durch eine Expertenbefragung (durchgeführt in 2009 und 2010), bei der Fachwissenschaftler, Fachdidaktiker und Fachleiter für das Fach Informatik befragt wurden, statt. Auf Grundlage des wissenschaftlich fundierten Kompetenzmodells wird die Entwicklung eines Testinstruments für den Kompetenzbereich Modellierung dargestellt. Hier erfolgt die Beschreibung der Entwicklung von Items, um die formulierten Kompetenzen zur objektorientierten Modellierung innerhalb der Komponenten des Kompetenzmodells überprüfen zu können.

Als theoretische Grundlage für die Erprobung des Messinstruments in einer Lerngruppe, wird die Entwicklung eines kompetenzförderlichen Lehr-/Lernarrangements erläutert.

Im Anschluss erfolgt die Darstellung der statistisch ausgewerteten Ergebnisse der Erprobung, deren Interpretation und die Formulierung von Forschungsfragen für künftige Forschungsarbeiten.

Abstract

Based on the outcomes of international studies, which show the inadequateness of the German education system, fostering learners' competencies becomes increasingly relevant. In order to further improve the quality of the German education system, educational policy requires the education system to implement national educational standards. These standards delineate how learning and teaching in schools should take place. Teaching and learning should be based on empirically-grounded competence models that should be developed by domain experts and psychologists cooperatively. For the evaluation of these competence models, educational policy demands the development of appropriate instruments to measure relevant competencies as well. In order to fulfill these demands, the development of an empirically-grounded competence model and a respective instrument will be described for the domain of computer science modeling. First, the relevance of object-oriented modeling for computer science education will be shown. Hereafter, the development of an empirically-grounded competence model and an associated test instrument will be shown. The development of the competence model requires two intermediate steps: In the first step it will be shown how the dimensions and components of the model can be derived theoretically. In a second step, the competence model will be refined empirically by referring to the results of the qualitative content analysis of experts interviews conducted in 2009 and 2010. In the interviews, three groups of experts have been interviewed: (1) experts of computer science, (2) experts of computer science education and (3) expert computer science teachers.

Based on the empirically-grounded competence model, the development of an instrument for measuring competencies will be described. Here, the creation of items to prove the respective competencies covered in the competence model will be illustrated. In order to evaluate the developed measurement instrument, a learning unit fostering informatics modeling was developed. This unit has been conducted in spring 2011 in secondary computer science education. After summarizing the statistically derived results of the evaluation, open issues for potential further research projects will be outlined.

Vorwort

Die vorliegende Dissertationsschrift wurde während meiner zweijährigen Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl „Didaktik der Informatik“ der Universität Paderborn und anschließend parallel zu meiner Beschäftigung in einem Wirtschaftsunternehmen verfasst.

Ich bedanke mich herzlich bei Herrn Prof. Dr. Johann S. Magenheim für die umfassende Unterstützung und stetige Förderung meines Promotionsvorhabens.
Frau Prof. Dr. Sigrid Schubert und Herrn Dr. Dieter Engbring gilt ebenfalls mein besonderer Dank für die Beratung, wissenschaftliche Förderung und Hilfestellung bei der thematischen Präzisierung.
Ein weiterer Dank gilt den Projektpartnern im DFG-Projekt „Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lern-Prozessen für Modellierung und Systemverständnis in der Informatik“. Hierbei danke ich insbesondere Wolfgang Nelles, Prof. Dr. Niclas Schaper und Dr. Peer Stechert für die zahlreichen konstruktiven Diskussionen und die kollegiale Zusammenarbeit.
Herrn Michael Dohmen danke ich für die tatkräftige Unterstützung bei der Durchführung der praktischen Erprobungen.
Ganz besonders bedanke ich mich bei meiner Familie für die liebevolle Unterstützung während der Promotion.

Ich widme diese Arbeit meiner Frau Christina und meinen Töchtern Elisa und Mirja.

Inhaltsverzeichnis

1. Einleitung

Im Rahmen der vorliegenden Dissertation wird der Prozess der Entwicklung und Erprobung eines Instruments zur Messung informatischer Modellierungskompetenz im Kontext der objektorientierten Softwareentwicklung dargestellt. Die Ergebnisse der Evaluation des Messinstrumentariums werden vorgestellt, interpretiert und anknüpfende Forschungsfragen formuliert. Ein wichtiger Meilenstein dieses Prozesses ist die Konzeption eines empirisch fundierten Kompetenzstrukturmodells, das als mögliches Kategoriengerüst für Modellierungskompetenz in der Informatik dienen kann. Dieses wurde im DFG-geförderten Projekt „Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-/Lern-Prozessen für Modellierung und Systemverständnis in der Informatik“ (kurz: *MoKoM*) erarbeitet. Es stellt die theoretische Basis für die Entwicklung und Evaluation des Instrumentariums dar. Ferner sollen aufbauend auf dem Strukturmodell in weiteren Forschungsvorhaben, die nicht Bestandteil dieser Dissertationsschrift sind, ein Kompetenzniveaumodell und -entwicklungsmodell erarbeitet werden. Ersteres soll Niveaustufungen enthalten, die mit unterschiedlich anspruchsvollen kognitiven Prozessen und Wissensanforderungen korrespondieren. Letzteres hat das Ziel, Annahmen über das Erreichen von Kompetenzen innerhalb eines bestimmten Zeitraums zu machen.
Um die Zielsetzung dieser Arbeit weiter zu spezifizieren, werden die zentralen Forschungsfragen dargelegt.

1.1. Forschungsfragen

1. Welche kognitiven und nicht-kognitiven Facetten umfasst informatische Modellierungskompetenz? (Entwicklung eines Kompetenzstrukturmodells)

2. Lässt sich ein Zuwachs an informatischer Modellierungskompetenz messbar machen? (Entwicklung und Erprobung eines Messinstruments)

1.2. Forschungsmethodik und Struktur der Arbeit

Als Reaktion auf internationale Vergleichsstudien, die unbestreitbare Mängel am deutschen Bildungssystem offenbart haben, soll im Kapitel 2 *Kompetenzorientierung als fachdidaktische Ausgangslage* ein Einblick in die Diskussion zur Kompetenzorientierung in Deutschland am Beispiel des Informatikunterrichts gegeben werden. In diesem Zusammenhang werden verschiedene Ansätze zur Entwicklung von Kompetenzmodellen für die gesamte informatische Bildung und für deren Teilbereiche vorgestellt. Sie können als Grundlage für die Entwicklung von Bildungsstandards und kompetenzorientiertem Unterricht angesehen werden.
Diese bildungspolitische Umorientierung motiviert maßgeblich die vorliegende Arbeit.
Das folgende Kapitel 3 *Modellierung im fachwissenschaftlichen und fachdidaktischen Kontext* beschreibt die Modellierung als Inhaltsbereich der Informatik. Hier sollen Kompetenzen gefördert und gemessen werden. Entsprechend wird die Bedeutung der Modellierung im Rahmen der objektorientierten Softwareentwicklung unter fachwissenschaftlicher und fachdidaktischer Perspektive erörtert.
Es ergeben sich folgende thematische Schwerpunkte für das Kapitel 3:

1. *Legitimation der informatischen Modellierung aus fachwissenschaftlicher und fachdidaktischer Sicht*
   Zunächst wird die besondere Bedeutung der Modellierung für das Gesamtgebiet der Informatik fachwissenschaftlich begründet. In einem Unterkapitel wird anhand unterschiedlicher didaktischer Beiträge aufgezeigt, welcher Wert der informatischen Modellierung im Unterricht an allgemeinbildenden Schulen zukommt.

2. *Erörterung und Festlegung einer normativ-theoretischen Grundlage für ein Kompetenzmodell*
   In diesem Kapitel wird die normativ-theoretische Basis für die Entwicklung des Kompetenzmodells erörtert und festgelegt. Insbesondere wird danach gefragt, welche kognitiven und nicht-kognitiven Facetten die Modellierungskompetenz beinhaltet.

3. *Erörterung und Festlegung einer normativ-theoretischen Grundlage für eine Unterrichtsreihe zur Erprobung des Kompetenzmessinstruments*
   Ziel ist es hier, das Kompetenzmessinstrument auf Tauglichkeit in der fachdidaktischen Praxis zu prüfen. Um dies zu erreichen, wird eine Unterrichtsreihe für Schüler der gymnasialen Oberstufe entwickelt. Die theoretischen Grundlagen für die Lerneinheit werden hier erarbeitet.

Zwei weitere Kapitel fokussieren die Entwicklung des empirisch gesicherten Kompetenzstrukturmodells für informatisches Modellieren. Diese erfolgt in zwei Stufen: Zunächst wird die normativ-theoretische Entwicklung des Kompetenzstrukturmodells dargestellt [Nelles et al. 2009], [Kollee et al. 2009] (siehe Kapitel 4 *Theoretische Entwicklung eines Kompetenzstrukturmodells für informatisches Modellieren*). Im nächsten Schritt wird die empirische Verfeinerung des Modells erläutert [Magenheim et al. 2010a] [Magenheim et al. 2010b] [Lehner et al. 2010] (siehe Kapitel 5 *Empirische Entwicklung eines Kompetenzstrukturmodells für informatisches Modellieren*).
Das Kapitel 6 *Entwurf eines Messinstruments für informatische Modellierungskompetenz und Entwicklung eines Lehr-/Lernarrangements zur Erprobung* befasst sich mit der Entwicklung des Messinstruments. Gemessen werden die im Kompetenzstrukturmodell kategorisierten Facetten informatischer Modellierungskompetenz. Kompetenzen werden hier in Form von Profilen definiert. Um sie messbar zu machen, werden jeweils qualitative und quantitative Items entwickelt, die zu dem Messinstrument gebündelt werden.
Eine Unterrichtsreihe auf der Grundlage des didaktischen Konzepts des Paderborner *Informatik Lernlabors (ILL)* wird für die Kompetenzmessung in der gymnasialen Oberstufe angepasst. Basis ist eine Inhaltseinheit *Kommissionierstation*. Diese hat sich in der Hochschullehre (insb. im Rahmen der Informatiklehrerausbildung) als besonders tauglich für die Entwicklung von Modellierungskompetenz erwiesen. Daher wird sie als Untersuchungssetting für die durchzuführenden Kompetenzmessungen herangezogen.
Die Erprobung des Messinstruments wird im Kapitel 7 *Erprobung des Messinstruments für informatische Modellierungskompetenz* beschrieben. Hier besteht - wie in Forschungsfrage 2 beschrieben - die Zielsetzung, Kompetenzentwicklung für Modellierungskompetenz und deren Teilbereiche zu messen. Hierzu sind zwei Messzeitpunkte notwendig, zu Beginn und zum Ende der Unterrichtseinheit.
Die Messergebnisse werden in Tabellen erfasst und grafisch dargestellt. Hierbei wird zunächst eine deskriptive statistische Analyse vorgenommen sowie ein induktives statistisches Verfahren angewandt. So wird nachgewiesen, dass das Instrument geeignet ist, einen statistisch signifikanten Kompetenzzuwachs zu ermitteln (Vergleich gepaarter Stichproben).
Abschließend werden die Ergebnisse der beiden Forschungsfragen zusammengefasst und daraus resultierende weiterführende Forschungsfragen erörtert.

1.3. Tätigkeit im DFG-geförderten Projekt *MoKoM*

In diesem Abschnitt wird die Tätigkeit des Autors als wissenschaftlicher Mitarbeiter in dem Projekt *MoKoM* vorgestellt. Ferner wird erläutert, welche Aktivitäten im Rahmen des oben beschriebenen Projekts *MoKoM* durchgeführt wurden und welche nicht mehr Bestandteil des Forschungsprojekts waren.
Zunächst wurde im Rahmen des Projekts *MoKoM* ein Kompetenzstrukturmodellmodell für die Inhaltsbereiche informatisches Modellieren und Systemverständnis entwickelt. Zielsetzung war es ein theoretisches Kategoriengerüst für die nach Dimensionen gegliederten kognitiven und nicht-kognitiven Voraussetzungen, über die ein Lernender verfügen soll, um Aufgaben und Probleme in dem genannten Aufgabenbereich zu lösen, zu entwickeln. In diesem Zusammenhang wurde eine normativ-theoretische Literaturrecherche anhand von fachdidaktischer und fachwissenschaftlicher Literatur vorgenommen, um die jeweiligen Kompetenzdimensionen und -komponenten für den Bereich der informatischen Modellierung und das Systemverständnis theoretisch herzuleiten. Hierbei wurde die in Kapitel 4 beschriebene normative Entwicklung der Dimension *K1.3*, die die Kompetenzen zur Modellierung enthält, vom Autor entwickelt. Im Rahmen dieses Forschungsschrittes wurden zwei Veröffentlichungen (Informatik Spektrum)[Nelles et al. 2009] und der (World Conference of Computer in Education 2009) [Kollee et al. 2009] als Mitautor publiziert.
Im nächsten Forschungsschritt (siehe Kapitel 5) ist die empirische Verfeinerung und Ausdifferenzierung des Kompetenzmodells durch Expertenbefragungen erfolgt. In diesem Arbeitsschritt hat der Autor die inhaltliche Gestaltung der Interviewszenarien für den Bereich der Modellierung erstellt und konzipiert. Weiterhin wurde ein Großteil der Interviews von ca. 60-90 Minuten von dem Autor zusammen mit den Kollegen der Arbeits- und Organisationspsychologie durchgeführt. Im Anschluss hat der Autor die technische und organisatorische Koordination der Transkriptionen für einen Teil der Interviews durchgeführt. Anschließend wurde die qualitative Inhaltsanalyse für die Interviewszenarien zur Modellierung und Aggregation der Ergebnisse zur informatischen Modellierung in einer Gesamtauswertung vorgenommen. In diesem zweiten Forschungsschritt wurden drei Veröffentlichungen als Mitautor publiziert, ISSEP 2010 [Magenheim et al. 2010a], IEEE EDUCON 2010 [Magenheim et al. 2010b] und WCC 2010 [Lehner et al. 2010].
Auf Grundlage der vorherigen Arbeiten wurde in einem weiteren Forschungsschritt des Projekts *MoKoM* die Entwicklung des Messinstruments auf Grundlage des empirisch gesicherten Kompetenzstrukturmodells vorgenommen (siehe Kapitel 6.1). In diesem Zusammenhang hat der Autor die Aufgaben und Items für die informatische Modellierung in Zusammenarbeit mit den Kollegen der Arbeits- und Organisationspsychologie kon-

zipiert. Hierbei hat der Autor insbesondere die Items zur informatischen Modellierung erstellt. Die Kollegen aus der Psychologie haben parallel die Items zu den nicht-kognitiven Anforderungen formuliert. Die logische Verknüpfung der Items zu Aufgaben mit lebensweltnahem Stimulusmaterial wurde gemeinschaftlich vorgenommen.
Die folgenden Arbeitsschritte haben im Anschluss an das Projekt *MoKoM* stattgefunden: Um das entwickelte Messinstrument für Kompetenzen zur informatischen Modellierung zu erproben, wurde ein Lehr-/Lernarrangement auf theoretischer Grundlage des Paderborner *Informatik Lernlabors* (siehe Kapitel 6.2) entwickelt. Dieses wurde zunächst von Studenten im Rahmen der Lehrveranstaltung *Informatik Lernlabor* entwickelt. Der theoretische Entwurf der Unterrichtsreihe sowie die technischen und softwaretechnische Eigenschaften wurden durch den Autor didaktisch reduziert und weiterentwickelt um für den schulischen Einsatz geeignet zu sein.
Die schulische Erprobung wurde gemeinsam vom Autor und einer studentischen Hilfskraft begleitet und von einem erfahrenen Informatiklehrer durchgeführt. Die beiden Kompetenzmessungen wurden vorbereitet und durchgeführt und deren Ergebnisse durch den Autor statistisch ausgewertet (siehe Kapitel 7).
Zum besseren Verständnis und zur Verdeutlichung der Kohärenz illustriert die Abbildung 1.1 den Inhalt der Arbeit und veranschaulicht den Aufbau und Zusammenhang der Kapitel. Hierbei wird außerdem die Rolle des Projekts *MoKoM* im Gesamtkontext dieser Arbeit illustriert. Im weiteren Verlauf der Arbeit soll die Grafik dafür verwendet werden, um den Fortschritt der Arbeit und den roten Faden für jedes Kapitel hervorzuheben.

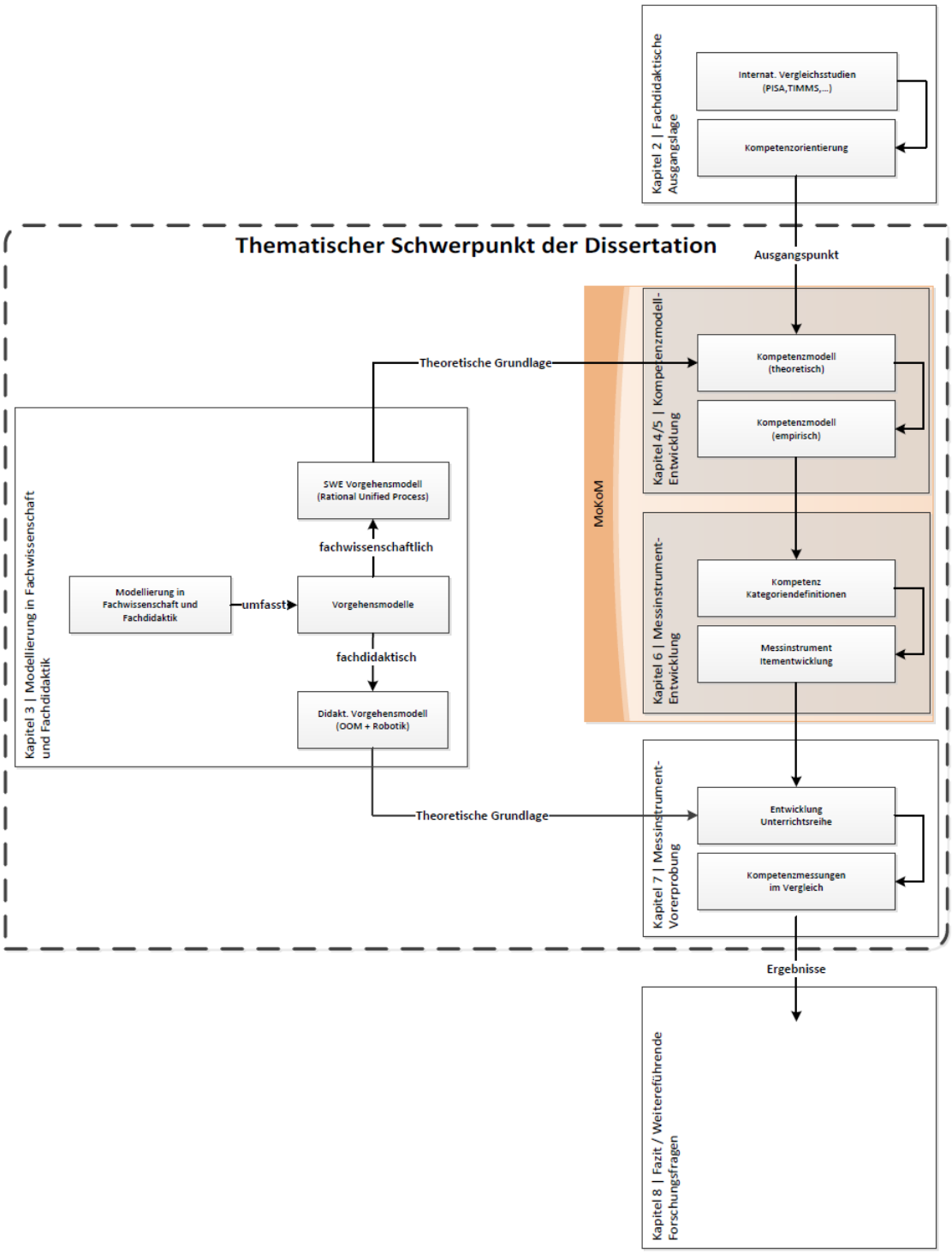


Abbildung 1.1.: Übersicht Kapitel und logischer Zusammenhang

1.4. Die Verwendung des Begriffs „Modell“

Da der Begriff „Modell“ in vielfältigem Kontext verwendet wird, sollen zum besseren Verständnis die folgenden Modellebenen vorgestellt werden: [1]

Ebene 1- Kompetenzmodellebene

- Umfasst die Erstellung des Kompetenzmodells für den Gegenstandsbereich der Informatischen Modellierung (Ebene2)

Ebene 2 - Fachwissenschaftliche Modellebene

- Stellt den Gegenstandsbereich dar für den die entsprechenden kognitiven und sozialkommunikativen Anforderungen im Kompetenzmodell (Ebene1) zusammengefasst und strukturiert werden

Ebene 3 - Vermittlungs-Modellebene

- Stellt die methodische Art und Weise dar, wie die im Kompetenzmodell (Ebene1) beinhalteten Kompetenzen zur informatischen Modellierung (Ebene2) vermittelt werden

Die einzelnen Modellebenen sind eng miteinander verflochten. Dies soll in der folgenden Abbildung illustriert werden.

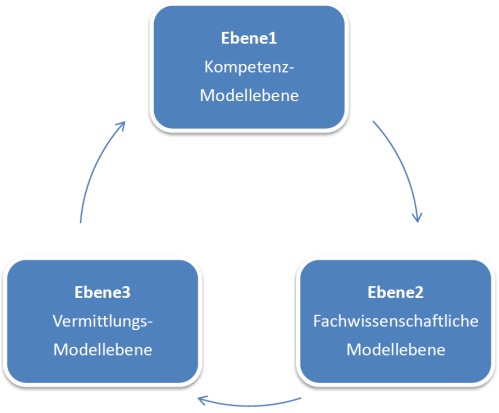


Abbildung 1.2.: Übersicht der Modellebenen

[1] Eine Definition des Begriffs „Modell“ aus fachwissenschaftlicher und fachdidaktischer Sicht erfolgt in Kapitel 3. Die Modellebenen sollen hier lediglich die unterschiedliche Verwendung der Begriffe „Modell“ oder „Modellierung“ innerhalb der vorliegenden Arbeit verdeutlichen.

2. Kompetenzorientierung als fachdidaktische Ausgangslage

Ausgelöst durch die Ergebnisse internationaler Vergleichsstudien, die Mängel am deutschen Bildungssystem aufgedeckt haben, findet die Vermittlung von Kompetenzen in der aktuellen Bildungsdiskussion als zentrales Bildungsziel besondere Beachtung. Hierbei wird über die Vermittlung von trägem Wissen [Whitehead 1939] hinaus die Entwicklung von prozeduralem und in vielfältigen Kontexten anwendbarem Handlungswissen gefordert.

> „Neu in der deutschen Bildungsdiskussion ist vor allem die Verknüpfung von Kompetenzorientierung und Standardisierung in der Qualitätsentwicklung von Schule und Unterricht, die als Reaktion auf unbefriedigende TIMSS und PISA Ergebnisse und bildungspolitische Impulse der OECD die derzeitigen Reformvorhaben im Bildungssystem maßgeblich beeinflusst [Drieschner 2009, S. 10]."

Nationale Bestrebungen bei der Entwicklung von bundesweiten Bildungsstandards verfolgen zur Lösung dieses Missstandes die folgende Zielsetzung [Drieschner 2009, S. 29]:

1. Die Festlegung von Kompetenzzielen.
2. Die Vermittlung der Kompetenzen durch geeignete didaktische und bildungsorganisatorische Maßnahmen.
3. Die Überprüfung, ob die festgelegten Kompetenzen erreicht wurden.

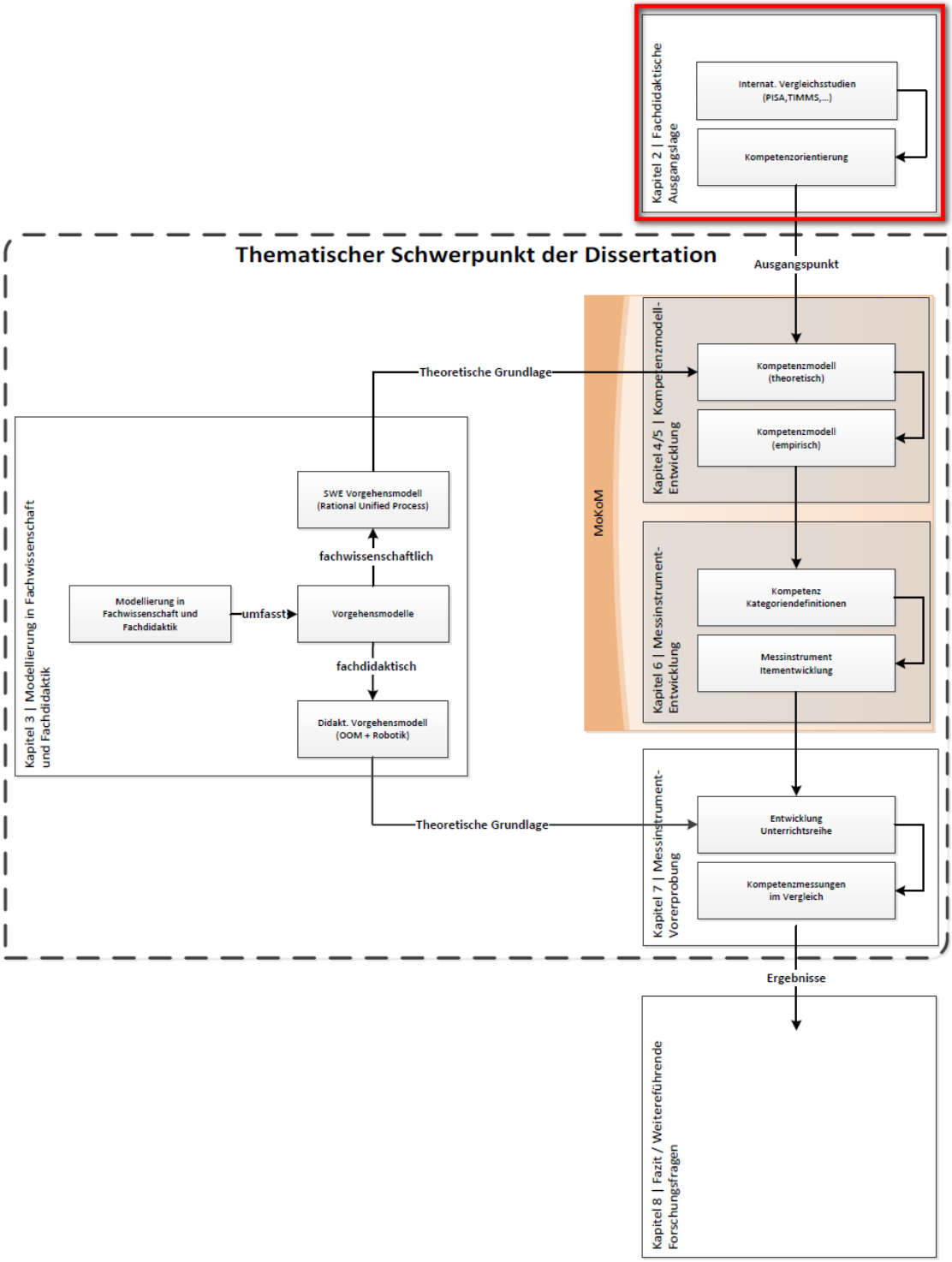


Abbildung 2.1.: Kapitel 2 im Gesamtkontext der Arbeit

2.1. Output-Orientierung als Replik auf Internationale Vergleichsstudien

Seit Veröffentlichung der Ergebnisse der TIMMS-Studie wird verstärkt über den aktuellen Zustand des deutschen Bildungssystems und dessen Entwicklungsperspektiven diskutiert. Insbesondere die empirischen Studien – wie z.B. PISA – haben unbestreitbare Mängel am deutschen Bildungssystem offenbart. Diese Schwächen haben sich insbesondere im internationalen Vergleich gezeigt und belegen eindeutig den Zusammenhang zwischen sozialen Faktoren und schulischem Erfolg im deutschen Bildungssystem [OECD 2001],[Baumert et al. 2000].
Durch die oben genannten internationalen Vergleichsstudien hat sich eine grundsätzliche Wende des deutschen Bildungssystems abgezeichnet: Wo früher Lehrpläne, Haushaltspläne, Rahmenrichtlinien und Prüfungsrichtlinien richtungweisend waren, spielen heute insbesondere die Lernergebnisse der Schülerinnen und Schüler eine Rolle. Man spricht in diesem Zusammenhang auch von einer *Outputorientierung* des Bildungssystems. Diese Umorientierung fokussiert insbesondere die Entwicklung von Kompetenzen in einer Unterrichtsdomäne [Klieme et al. 2007].
Um die Qualität des deutschen Bildungssystems zu sichern, wurden Anstrengungen unternommen, um nationale Bildungsstandards zu verfassen. Diese stellen verbindliche Anforderungen an das Lehren und Lernen in der Schule [Klieme et al. 2007, S. 9] dar. Folglich bieten sie einen Orientierungspunkt, um Lehrpersonen professionelles Handeln zu ermöglichen und die in den Bildungsstandards implizit aufgeführten Kompetenzaspekte bestmöglich zu vermitteln [Klieme et al. 2007].
Es ergibt sich dementsprechend eine hohe Erwartungshaltung an die Implementierung von Bildungsstandards im deutschen Bildungssystem. Neben der qualitativen Verbesserung schulischer Lernergebnisse steht eine bundesweite Vergleichbarkeit und Anschlussfähigkeit von Schulabschlüssen im Fokus [Riecke-Baulecke und Artelt 2004, S. 7]. Ferner verspricht man sich einen Beitrag zu mehr Bildungsgerechtigkeit [Lankes 2006, S. 9].
Neben der Diskussion zur Kompetenzorientierung und Bildungsstandards in Deutschland hat das Thema auch international einen hohen Stellenwert:
Die UNESCO-Publikation „Understanding Information Literacy: A Primer“ (UNESCO 2008) werden notwendige sog. *Literacys* aufgeführt, die mündige Bürger im 21. Jahrhundert benötigen [Horton 2007, S. 3].

1. the Basic or Core functional literacy fluencies (competencies) of reading, writing, oralcy and numeracy;

2. Computer Literacy;
3. Media Literacy;
4. Distance Education and E-Learning;
5. Cultural Literacy;
6. Information Literacy.

Ein weiterer Ansatz der OECD sieht aufgrund steigender Anforderungen im Alltag der Wissensgesellschaft die Definition von Schlüsselkompetenzen von zentraler Bedeutung. In diesem Zusammenhang hat sie in dem Bericht „Definition and Selection of Key Competencies (DeSeCo)“ drei zentrale Ausprägungen von Schlüsselkompetenzen definiert:

1. Interaktive Anwendung von Medien und Mitteln (Tools),
2. Interagieren in heterogenen Gruppen,
3. Eigenständiges Handeln (OECD 2005).

Der Zusammenhang zwischen fachbezogenen Kompetenzen im Informatikunterricht und Schlüsselkompetenzen gemäß DESECO werden im Kapitel 4.2 erläutert.
Neben diesen exemplarisch näher aufgeführten nationalen und internationalen Beiträgen zu Standards und Kompetenzen, sind beispielhaft weitere zu nennen (NCTM-Standards - siehe auch Kapitel 2.2) [NCTM - National Council of Teachers of Mathematics 2000], Curricula der Bachelor und Master Ausbildung (Bologna Declaration) [Rectors, Universities 1999] und der Europäischer Qualifikationsrahmen für lebenslanges Lernen (EQR) [Europäische Gemeinschaften 2008].
Ferner wurde ein DFG Schwerpunktprogramm zum Thema „Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen“ kurz: „Kompetenzmodelle“ etabliert. Das DFG-Schwerpunktprogramm wird in einem Zeitraum von sechs Jahren (2007-2013) gefördert. In dem interdisziplinären Forschungsprogramm arbeiten Psychologen, Erziehungswissenschaftler und Fachdidaktiker zusammen [Leutner 2006].
Explizit für die informatische Bildung sind zusätzlich das ACM K-12 Curriculum [Tucker2006 2006] und die GI Bildungsstandards Informatik Sekundarstufe I [GI 2008] (siehe auch Kapitel 2.2) zu nennen.
Die zahlreichen nationalen und internationalen Beiträge und Maßnahmen zum Thema Bildungsstandards und Kompetenzen zeigen die hohe Relevanz der Thematik innerhalb

der aktuellen bildungspolitischen Diskussion und stellen eine zentrale Motivation für die vorliegende Dissertationsschrift dar. Das folgende Kapitel erläutert Ansätze zur Entwicklung von Bildungsstandards in der Informatik.

2.2. Bildungsstandards in der Informatik

Bei der Entwicklung von Bildungsstandards sind nach Klieme-Expertise gesellschaftliche und pädagogische Zielentscheidungen, wissenschaftlich fundierte Aussagen zum Aufbau von Kompetenzen und Konzepte sowie Verfahren zur Testentwicklung mit einzubeziehen.

> „In diesem Sinne gehen in die Entwicklung von Bildungsstandards (a) gesellschaftliche und pädagogische Zielentscheidungen, (b) wissenschaftliche, insbesondere fachdidaktische und psychologische Aussagen zum Aufbau von Kompetenzen, sowie (c) Konzepte und Verfahren der Testentwicklung ein [Klieme et al. 2007, S. 19]."

Ausgehend von dieser bildungspolitischen Anforderung stellt sich im Hinblick auf die zu untersuchende Teildisziplin der Informatik die Forschungsfrage (1), welche kognitiven und nicht-kognitiven Facetten informatische Modellierungskompetenz umfasst?
Die Klieme-Expertise definiert darüber hinaus konkrete Vorgaben für die Entwicklung von Bildungsstandards. In diesem Zusammenhang wird die Eignung von Bildungsstandards als verbindliche Vorgabe für Schulqualität anhand von Gütekriterien für sog. *Performance Standards* manifestiert [Drieschner 2009, S. 28][Klieme et al. 2007]:

- Fokussierung auf Kernbereiche der Fächer oder Fächergruppen und ihre grundlegenden Begriffsvorstellungen, Grundprinzipien, Verfahren und grundlegenden Wissensbestände
- die Benennung von Kompetenzen als Resultate übergreifenden, kumulativen, systematischen und vernetzten Lernens, die zu vorgegebenen Zeitpunkten (etwa am Ende der 4. Klasse) verfügbar sein sollen
- die verbindliche Festlegung von Mindeststandards sowie die Entwicklung von Regel- und Maximalstandards auf der Basis von Kompetenzstufenmodellen zur differenzierten Bestimmung von Lernentwicklungen und Lernleistungen
- die Verständlichkeit für Lehrer, Schüler und Eltern
- die Erreichbarkeit durch geeignete didaktisch-methodische Maßnahmen

- eine präzise Beschreibung von Leistungserwartungen, die Konzeptualisierungen von Aufgaben und Testverfahren ermöglicht

Wie bereits erläutert, sollen Bildungsstandards fachdidaktische und psychologische Aussagen zum Aufbau von Kompetenzen in der jeweiligen Domäne beschreiben. Infolgedessen ist es unerlässlich, einen einheitlichen Kompetenzbegriff zu verwenden. In diesem Zusammenhang soll die in der Bildungsforschung etablierte Kompetenzdefinition nach Weinert & Klieme für diese Forschungsarbeit richtungweisend sein:

> „In Übereinstimmung mit [Weinert 2002, S. 27ff] verstehen wir unter Kompetenzen die bei Individuen verfügbaren oder von ihnen erlernbaren kognitiven Fähigkeiten und Fertigkeiten, bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können [Klieme et al. 2007, S. 70]."

Kompetenzen beziehen sich demnach auf komplexe Anforderungssituationen und bestehen nicht allein aus einzelnen Fähigkeiten und Fertigkeiten. Sie beinhalten neben kognitiven auch nicht- kognitive, motivationale, willensmäßige, personale und sozialkommunikative Aspekte, die zum selbständigen Lösen domänenrelevanter Probleme notwendig sind. Ein weiterer wichtiger Aspekt, den das Weinert'sche Kompetenzverständnis umfasst, ist die Bereitschaft ein Problem zu lösen. Weinert unterscheidet folgende, zusammenwirkende Kompetenzfacetten: Fähigkeit, Wissen, Verstehen, Können, Handeln, Erfahrung, Motivation [Klieme et al. 2007, S. 73].
Im Folgenden soll anhand eines Ansatzes zur Entwicklung von Bildungsstandards für das Unterrichtsfach Informatik in der Sekundarstufe I aufgezeigt werden, dass der Kompetenzbegriff nicht immer einheitlich verwendet wird. Dies soll den Leser dahingehend sensibilisieren, dass eine einheitliche Verwendung des Kompetenzbegriffs wichtig ist und künftige Forschungsarbeiten stets ein gemeinsames Kompetenzverständnis zugrunde legen sollten.

Bildungsstandards Informatik für die Sekundarstufe I

Die Bildungsstandards für den Informatikunterricht in der Sekundarstufe I orientieren sich vorwiegend an den „Principles and Standards for School Mathematics" der „National Council of Teachers of Mathematics (NCTM)" [NCTM - National Council of Teachers of Mathematics 2000] und diversen fachdidaktischen Diskussionen. Es erfolgt keine Zugrundelegung empirisch fundierter Aussagen zum Aufbau von Kompetenzen im Informatikunterricht.

„Diese vorliegende Strukturierung ist schrittweise in vielen Workshops mit zahlreichen Lehrerinnen und Lehrern sowie Fachdidaktikerinnen und Fachdidaktikern entstanden und somit das Ergebnis eines mehrjährigen Diskussionsprozesses mit vielen Beteiligten. Die grundsätzliche Unterteilung in Inhalts und Prozessbereiche wurde von den NCTM-Standards übernommen, weil sie sich dort bereits als sehr erfolgreich erwies. Damit wird allerdings nicht behauptet, dass dies die einzig mögliche sinnvolle Strukturierung sei [GI 2008, S. 12]."

Die in der Klieme-Expertise geforderte Einbeziehung von Psychologen und Erziehungswissenschaftlern bei der Formulierung von Kompetenzen ist somit nicht geleistet.
Die Bildungsstandards basieren auf einem impliziten Kompetenzmodell, welches in Inhalts- und Prozessbereiche getrennt ist. Hier werden nach Aussage der Autoren jahrgangsstufenübergreifend Mindestanforderungen in Form von Kompetenzen formuliert.
Die Inhaltsbereiche umfassen [GI 2008, S. 12ff]:

- *Information und Daten*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - verstehen den Zusammenhang von Information und Daten sowie verschiedene Darstellungsformen für Daten,
  - verstehen Operationen auf Daten und interpretieren diese in Bezug auf die dargestellte Information,
  - führen Operationen auf Daten sachgerecht durch.
- *Algorithmen*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - kennen Algorithmen zum Lösen von Aufgaben und Problemen aus verschiedenen Anwendungsgebieten und lesen und interpretieren gegebene Algorithmen,
  - entwerfen und realisieren Algorithmen mit den algorithmischen Grundbausteinen und stellen diese geeignet dar.
- *Sprachen und Automaten*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - nutzen formale Sprachen zur Interaktion mit Informatiksystemen und zum Problemlösen,
  - analysieren und modellieren Automaten.
- *Informatiksysteme*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - verstehen die Grundlagen des Aufbaus von Informatiksystemen und deren Funktionsweise,

  - wenden Informatiksysteme zielgerichtet an,
  - erschließen sich weitere Informatiksysteme.

- *Informatik, Mensch und Gesellschaft*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - benennen Wechselwirkungen zwischen Informatiksystemen und ihrer gesellschaftlichen Einbettung,
  - nehmen Entscheidungsfreiheiten im Umgang mit Informatiksystemen wahr und handeln in Übereinstimmung mit gesellschaftlichen Normen,
  - reagieren angemessen auf Risiken bei der Nutzung von Informatiksystemen.

Die Prozessbereiche sind:

- *Modellieren und Implementieren*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - erstellen informatische Modelle zu gegebenen Sachverhalten,
  - implementieren Modelle mit geeigneten Werkzeugen,
  - reflektieren Modelle und deren Implementierung.

- *Begründen und Bewerten*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - stellen Fragen und äußern Vermutungen über informatische Sachverhalte,
  - begründen Entscheidungen bei der Nutzung von Informatiksystemen,
  - wenden Kriterien zur Bewertung informatischer Sachverhalte an.

- *Strukturieren und Vernetzen*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - strukturieren Sachverhalte durch zweckdienliches Zerlegen und Anordnen,
  - erkennen und nutzen Verbindungen innerhalb und außerhalb der Informatik.

- *Kommunizieren und Kooperieren*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - kommunizieren fachgerecht über informatische Sachverhalte,
  - kooperieren bei der Lösung informatischer Probleme,
  - nutzen geeignete Werkzeuge zur Kommunikation und Kooperation.

- *Darstellen und Interpretieren*
  Schülerinnen und Schüler aller Jahrgangsstufen
  - interpretieren unterschiedliche Darstellungen von Sachverhalten,
  - veranschaulichen informatische Sachverhalte,
  - wählen geeignete Darstellungsformen aus.

In einem weiteren Schritt werden unter Berücksichtigung der unterschiedlichen Ausprägungen informatischer Bildung der Bundesländer sog. Nahtstellen definiert und jene zu erreichende Kompetenzen aus den unterschiedlichen Inhalts- und Prozessbereichen des Informatikunterrichts zugeordnet.

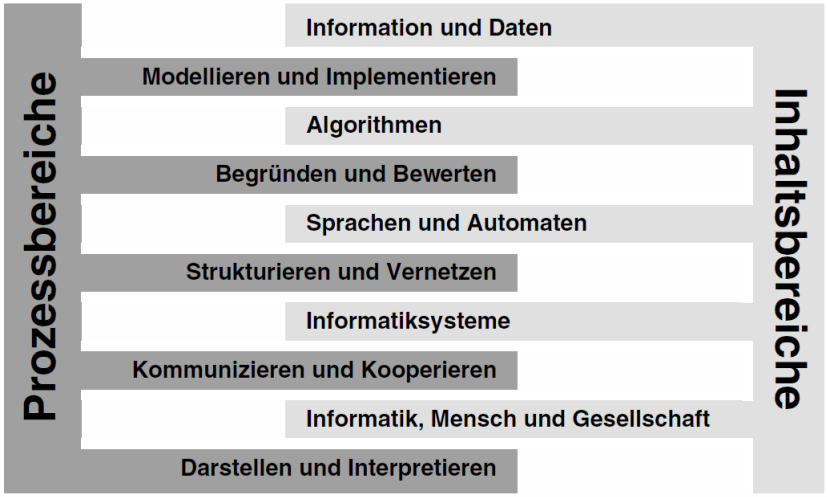


Abbildung 2.2.: Inhalts und Prozessbereiche [GI 2008, S. 11]

Konkret werden innerhalb der einzelnen Inhalts- und Prozessbereiche im Sinne von Mindeststandards jeweils Kompetenzen differenziert nach zwei Jahrgangsstufen, nämlich *Schülerinnen und Schüler der Jahrgangsstufen 5 bis 7* und *Schülerinnen und Schüler der Jahrgangsstufen 8 bis 10*, aufgeführt. Ein empirisch gesichertes Kompetenzmodell und ein Instrument zur Messung der vorgestellten Kompetenzen liegen nicht vor.
In der Abbildung 2.3 wird diese Zuordnung am Beispiel des Prozessbereichs *Modellieren & Implementieren* aufgezeigt:
Neben den methodologischen Abweichungen von der in der Klieme-Expertise vorgeschlagenen Vorgehensweise zur Entwicklung von Bildungsstandards, ergeben sich weitere Diskussionspunkte bei der Herleitung relevanter Kompetenzen und bei der Verwendung des Kompetenzbegriffs.
Die Ableitung der Kompetenzen erfolgt lediglich anhand der fachlichen Systematik des Informatikunterrichts und vorrangig anhand der Komponenten, die innerhalb der NCTM-Standards für die Schulmathematik definiert wurden. Nach Klieme-Expertise ist eine derartige rein normative Vorgehensweise zur Ableitung von Kompetenzen nicht hinreichend. Deskriptive Modelle bedürfen nach Klieme der Absicherung durch empirische fachdidaktische und lernpsychologische Forschung.

Prozessbereiche

Modellieren und Implementieren

Schülerinnen und Schüler aller Jahrgangsstufen
erstellen informatische Modelle zu gegebenen Sachverhalten

Schülerinnen und Schüler der Jahrgangsstufen 5 bis 7	**Schülerinnen und Schüler der Jahrgangsstufen 8 bis 10**
► betrachten Informatiksysteme und Anwendungen unter dem Aspekt der zugrunde liegenden Modellierung	► analysieren Sachverhalte und erarbeiten angemessene Modelle
► identifizieren Objekte in Informatiksystemen und erkennen Attribute und deren Werte	► entwickeln für einfache Sachverhalte objektorientierte Modelle und stellen diese mit Klassendiagrammen dar
	► modellieren die Verwaltung und Speicherung großer Datenmengen mithilfe eines Datenmodells
	► modellieren reale Automaten mithilfe von Zustandsdiagrammen

Abbildung 2.3.: Prozessbereich Modellieren & Implementieren [GI 2008, S. 19]

Eine Ableitung von Fähigkeiten und Fertigkeiten aus der Fachsystematik der Informatik, anhand von Bildungsstandards der Mathematik und anhand zahlreicher Diskussionen von erfahrenen Informatiklehrern ist sicherlich wertvoll aber im Sinne der Klieme-Expertise nicht geeignet, um Kompetenzen zu formulieren.
Die Kompetenzbeschreibungen innerhalb der vorgestellten Inhalts- und Prozessbereiche entsprechen darüber hinaus nicht dem Weinert'schen Kompetenzverständnis.
Die Bildungsstandards sehen eine explizite Trennung von Inhalts und Prozessbereichen vor. Hier werden einzelne Fähigkeiten und Fertigkeiten beschrieben, die die Schülerinnen und Schüler einer Jahrgangsstufe in einem bestimmten Teilbereich des Informatikunterrichts erreichen sollen. Dies widerspricht dem Weinert'schen Kompetenzverständnis insofern, als dass Kompetenzen aus komplexen Anforderungssituationen, die es zu bewältigen gilt, bestehen. Hier sollte vielmehr eine ganzheitliche Betrachtung von kognitiven und nicht-kognitiven Kompetenzaspekten berücksichtigt werden.
Die vorliegende Arbeit hat die Zielsetzung ein Kompetenzstrukturmodell zu entwickeln. Entsprechend der zuvor dargestellten Forderungen und dargestellter Mängel in bestehenden Ansätzen soll bei der Entwicklung des Kompetenzmodells explizit das Weinert'sche Kompetenzverständnis zugrunde gelegt werden. Das Kompetenzmodell wird im Rahmen des *MoKoM-Projekts* von Fachdidaktikern und Psychologen entwickelt und durchläuft Prozesse zur empirischen Differenzierung und Absicherung.

Als Grundlage für das Kapitel 4 soll im Folgenden der Begriff des Kompetenzmodells erläutert werden. Wie in der Einleitung erwähnt, sieht die vorliegende Arbeit die Entwicklung eines empirisch gesicherten Kompetenzstrukturmodells für die informatische Modellierung vor.

2.3. Kompetenzmodelle in der Informatik

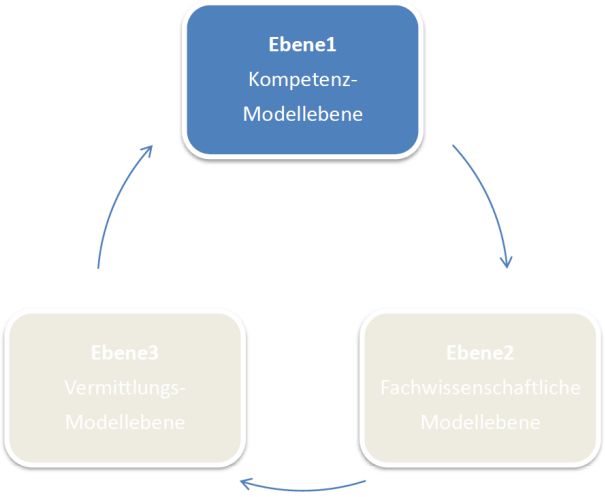


Abbildung 2.4.: Modellebene 1 - Kompetenzmodell-Ebene

Bildungsstandards sollen sich nach Klieme auf Kompetenzmodelle stützen, die in Zusammenarbeit von Fachdidaktik und Fachwissenschaft mit Psychologen oder Erziehungswissenschaftlern entwickelt werden. Diese spezifizieren die Dimensionen und Kategorien von Kompetenz, die mit entsprechend angepassten Aufgabensammlungen empirisch überprüft werden können.
Um das Erreichen der Standards empirisch zu überprüfen, ist es nach Klieme-Expertise unabdingbar, entsprechende Testverfahren und Aufgaben zu entwickeln. Insbesondere im Hinblick auf einen aktiven Schulbezug, sind Ergebniskontrollen unverzichtbar [Klieme et al. 2007] und stellt eine zentrale Motivation zur Konzeption eines Kompetenzmessinstruments für die Domäne der Modellierung im Rahmen des Software-Engineerings dar (siehe Forschungsfrage 2:). Darüber hinaus statuieren Schecker & Parchmann den Bedarf an empirisch fachdidaktischer und lernpsychologischer Forschung bei der Entwicklung von Kompetenzmodellen, die eine wesentliche theoretische Grundlage für die Entwicklung von Bildungsstandards darstellen.

„Normative Modelle dürfen nicht allein aus fachlichen Bildungszielen abgeleitet werden. Sie sollen vielmehr eine theoretische Fundierung aus der Lernpsychologie aufweisen. Ein deskriptives Modell bedarf der Absicherung durch empirische fachdidaktische und lernpsychologische Forschung [Schecker und Parchmann 2006, S. 47]."

Kompetenzmodelle

Aufbauend auf dem Weinert'schen Kompetenzverständnis charakterisieren Schecker & Parchmann Kompetenzmodelle folgendermaßen:

„Die Beschreibung von anforderungs- bzw. domänenbezogenen Kompetenzen, deren Erlangung durch Lernsituationen unterstützt werden soll, erfordert eine Systematik oder mit anderen Worten ein tragfähiges und für Messung und Lernen umsetzbares Kompetenzmodell [Schecker und Parchmann 2006, S. 46]."

Es lassen sich darüber hinaus drei wesentliche Typen von Kompetenzmodellen unterscheiden [Klieme et al. 2007][Schecker und Parchmann 2006]:

1. *Kompetenzstrukturmodell*
   Deskriptive Kompetenzstrukturmodelle dienen als mögliches Kategoriengerüst, um zu erreichende Kompetenzen ordnen und darstellen zu können. Sie konstituieren das Gefüge einer nach Dimensionen gegliederten Beschreibung kognitiver Voraussetzungen, über die ein Lernender verfügen soll, um Aufgaben und Probleme in einem bestimmten Anforderungsbereich lösen zu können. Kompetenzstrukturmodelle sind explizit ausformuliert oder liegen implizit vor.

2. *Kompetenzstufenmodell*
   Modelle dieses Typs umfassen Kompetenzstufungen, die mit unterschiedlich anspruchsvollen kognitiven Prozessen und Wissensanforderungen korrespondieren. Dementsprechend sind jene Stufen mit dem Erreichen eines bestimmten Niveaus einer Kompetenzdimension verknüpft.

3. *Kompetenzentwicklungsmodell*
   Kompetenzentwicklungsmodelle machen Annahmen darüber, in welcher Weise sich Kompetenzstrukturen herausbilden. Häufig werden in zeitlicher Perspektive Erwartungen an das Erreichen bestimmter Kompetenzen nach Altersstufen/Schulstufen gemacht. Sie basieren auf einer gestuften Kompetenz (siehe Kompetenzstufenmodell) und legen fest, unter welchen Bedingungen der Übergang in eine höhere Stufe

möglich wird. Zur tatsächlichen Beschreibung, gezielten Planung und Förderung von Entwicklungen muss fachliche Perspektive mit Voraussetzungen der Lernenden und weiteren Einflussfaktoren verknüpft werden.

Ansätze für Kompetenzmodelle in der Schulinformatik

Im Rahmen der vorliegenden Arbeit soll die theoretische und empirische Entwicklung eines Kompetenzstrukturmodells für den Teilbereich der informatischen Modellierung beschrieben werden. In diesem Zusammenhang werden zunächst bestehende Ansätze zur Entwicklung von Kompetenzmodellen in der Informatik aufgeführt.
Kohl untergliedert die bestehenden Ansätze für Kompetenzmodelle in der Schulinformatik in zwei Kategorien (*(1) Modelle für die gesamte informatische Bildung* und *(2) Modelle für Teilbereiche der informatischen Bildung*) und erläutert diese im Rahmen seiner Dissertation. Die folgende Liste gibt einen Überblick über die bestehenden Ansätze zur Kompetenzmodellierung [Kohl 2009, S. 47].

1. Modelle für die gesamte informatische Bildung
   - Drei Dimensionen des Model of ICT-Competence-Classes (Magenheim 2005)
   - Fünf Stufen von Kompetenz auf Grundlage des PISA-Mathematik-Kompetenzmodells (Friedrich 2003)
   - Das implizite Kompetenzmodell als Grundlage der KMK-Bildungsstandards (Puhlmann et. al. 2008)
2. Modelle für Teilbereiche der informatischen Bildung
   - Ansätze für ein Kompetenzmodell für informatisches Modellieren (Brinda / Schulte, 2005)
   - Ansatz zur Entwicklung eines Kompetenzmodells für die theoretische Informatik anhand von Kategorisierungen von Aufgaben eines Schülerwettbewerbs (Schlüter und Brinda 2007)
   - Kompetenzmodell für die angewandte Informatik in zwei Dimensionen (Dorninger 2007)

Kohl stellt ein Kompetenzmodell für die *Algorithmik in der Sekundarstufe I* vor. Dieses basiert auf dem Inhaltsbereich *Algorithmen* und den dort aufgeführten jahrgangsstufenübergreifenden Kompetenzen und denjenigen Kompetenzen für Schülerinnen und Schüler der Jahrgangsstufe 8 bis 10 (siehe Abbildung 2.5).

Algorithmen

Schülerinnen und Schüler aller Jahrgangsstufen kennen Algorithmen zum Lösen von Aufgaben und Problemen aus verschiedenen Anwendungsgebieten und lesen und interpretieren gegebene Algorithmen

Schülerinnen und Schüler der Jahrgangsstufen 5 bis 7	**Schülerinnen und Schüler der Jahrgangsstufen 8 bis 10**
▶ benennen und formulieren Handlungsvorschriften aus dem Alltag	▶ überprüfen die wesentlichen Eigenschaften von Algorithmen
▶ lesen und verstehen Handlungsvorschriften für das Arbeiten mit Informatiksystemen	▶ lesen formale Darstellungen von Algorithmen und setzen sie in Programme um
▶ interpretieren Handlungsvorschriften korrekt und führen sie schrittweise aus	

Abbildung 2.5.: Inhaltsbereich Algorithmen 1/2 [GI 2008, S. 15]

Schülerinnen und Schüler aller Jahrgangsstufen entwerfen und realisieren Algorithmen mit den algorithmischen Grundbausteinen und stellen diese geeignet dar

Schülerinnen und Schüler der Jahrgangsstufen 5 bis 7	**Schülerinnen und Schüler der Jahrgangsstufen 8 bis 10**
▶ benutzen die algorithmischen Grundbausteine zur Darstellung von Handlungsvorschriften	▶ stellen die algorithmischen Grundbausteine formal dar
▶ entwerfen Handlungsvorschriften als Text oder mit formalen Darstellungsformen	▶ verwenden Variablen und Wertzuweisungen
▶ entwerfen und testen einfache Algorithmen	▶ entwerfen, implementieren und beurteilen Algorithmen
	▶ modifizieren und ergänzen Quelltexte von Programmen nach Vorgaben

Abbildung 2.6.: Inhaltsbereich Algorithmen 2/2 [GI 2008, S. 16]

Ausgehend von den oben aufgeführten Kompetenzen entwickelt Kohl ein Kompetenzmodell basierend auf vier Dimensionen (in der Dissertation Kohl als Komponenten bezeichnet):

Komponente A - Eigenschaften von Algorithmen

Komponente B - algorithmische Grundbausteine und Datentypen

Komponente C - Arbeit mit Algorithmen

Komponente D - Programmentwicklung

Folgende Abbildung 2.7 zeigt das Kompetenzstrukturmodell, welches Kohl für den Inhaltsbereich *Algorithmen* vorschlägt.
Darüber hinaus wird den oben strukturell dargestellten Kompetenzen mit folgender Niveaustufung graduiert. Diese wurde auf Grundlage der sog. SOLO-Taxonomie entwickelt [Chan et al. 2010]: [1]

1. *Stufe 1*
   Diese Stufe umfasst *grundlegende Kompetenzen*, die eine einfache Verknüpfung von algorithmischen Grundbausteinen vorsieht.

2. *Stufe 2*
   Diese Stufe enthält *vertiefte Kompetenzen*, die das Analysieren, Implementieren, Modifizieren und Prüfen von Algorithmen mit mehreren ineinander Verschachtelten Verzweigungen und Wiederholungen erfordert.

3. *Stufe 3*
   Diese Stufe umfasst den Umgang mit *komplexen Algorithmen*, die das Erklären und Anwenden von Unterprogrammen mit Parametern erfordert.

Auf Grundlage des zuvor dargestellten Kompetenzstrukturmodells und dieser Niveaustufung wird das folgende Kompetenz-Stufenmodell vorgeschlagen.

[1] Structure of the Observed Learning Outcome (SOLO)

Kompetenzen der GI-Empfehlungen der Jahrgangsstufen 8 bis 10 *Schülerinnen und Schüler*	Komponente des Kompetenzmodells „Algorithmen“	Kompetenzen des Kompetenzmodells „Algorithmen“ der Jahrgangsstufen 8 bis 10 Die Schülerinnen und Schüler
- überprüfen die wesentlichen Eigenschaften von Algorithmen	A Eigenschaften von Algorithmen	- erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen - überprüfen die wesentlichen Eigenschaften von Algorithmen - nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind
- stellen die algorithmischen Grundbausteine formal dar *- verwenden Variablen und Wertzuweisungen*	B algorithmische Grundbausteine und Datentypen	- erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen, Wiederholungen und wenden diese Erklärungen an - stellen die algorithmischen Grundbausteine dar - verwenden Datentypen
- lesen formale Darstellungen von Algorithmen und setzen sie in Programme um *- modifizieren und ergänzen Quelltext eines Programms nach Vorgaben*	C Arbeit mit Algorithmen	- analysieren gegebene Algorithmen - prüfen Algorithmen - setzen gegebene Algorithmen in Programme um - modifizieren und ergänzen Algorithmen bzw. Programme
- entwerfen, implementieren und beurteilen Algorithmen	D Programmentwicklung	- entwerfen Programme - implementieren Programme mit einem Programmiersystem - testen Programme auf ihre Funktionalität

Abbildung 2.7.: Kompetenzstrukturmodell Algorithmen [Kohl 2009, S. 90]

Entgegen der Empfehlung der Klieme-Expertise, Bildungsstandards auf Grundlage eines theoretisch und empirisch fundierten Kompetenzmodells zu entwickeln, wird in der Jenaer Forschungsarbeit ein konträrer Weg eingeschlagen, nämlich die Ableitung eines Kompetenzmodells anhand der GI-Empfehlungen für Bildungsstandards. Hierbei ist aus Sicht des Autors zu bemängeln, dass die Kompetenzmodellierung als Grundlagenarbeit vor der Formulierung von Empfehlungen für Bildungsstandards hätte stattfinden müssen und nicht umgekehrt.
Dieser Umstand legitimiert die Forschungen im Rahmen des *MoKoM-Projekts* und der vorliegenden Dissertation. Dementsprechend gilt es, in einem ersten Schritt ein Kompetenzmodell zu entwickeln, das zunächst normativ-theoretisch abgeleitet und in weiteren Entwicklungsschritten empirisch ergänzt wird. Hierbei soll eine interdisziplinäre Zusammenarbeit zwischen Psychologen und Informatikern stattfinden.
Neben der interdisziplinären Entwicklung von Kompetenzmodellen wird in der Klieme-Expertise eine Entwicklung entsprechender Testverfahren und Aufgaben gefordert. Bevor im weiteren Verlauf (siehe Kapitel 6.1) die Aufgabenentwicklung im Rahmen des *MoKoM-Projekts* und der vorliegenden Dissertation vorgestellt werden, soll nach einer allgemeinen Beschreibung der Kompetenzmessung Kohls Ansatz zur Aufgabenentwicklung dargestellt werden. Im Anschluss wird erläutert, wie Kohls Erkenntnisse bei der Aufgabenentwicklung die Forschungsarbeit dieser Dissertation beeinflusst haben, und wie sich die Aufgabenentwicklung im *MoKoM-Projekt* und der vorliegenden Arbeit von Kohls Ansatz unterscheiden.

2.4. Kompetenzmessung im Informatikunterricht

Mit Hilfe von Kompetenzmodellen werden diejenigen Kompetenzen spezifiziert, die die Lernenden im jeweiligen Teilbereich der Informatik erwerben sollen. Sie umfassen deren Kompetenzstruktur (Kompetenzstrukturmodell) sowie die unterschiedlichen Niveaustufen jener Kompetenzen (Kompetenzstufenmodell). Darüber hinaus legen Bildungsstandards ein Minimalniveau fest, das von allen Lernern erreicht werden soll [Klieme et al. 2007, S. 81].
Um diese Standards im Unterricht praktisch einsetzen zu können, bedürfen die im Kompetenzmodell formulierten Anforderungen einer weiteren Konkretisierung durch Aufgabenstellungen und Testverfahren. Damit die Standards als Orientierung für Lehrpersonen dienen können; gilt es somit Aufgaben zu entwickeln mit denen fachliche Bildungsziele konkretisiert werden können und ein Orientierungspunkt für die Leistungsbewertung geschaffen wird.

Stufen / Komponenten	Stufe I *Die Schülerinnen und Schüler haben grundlegende Kompetenzen zu Algorithmen.* *Die Schülerinnen und Schüler …*	Stufe II *Die Schülerinnen und Schüler haben vertiefte Kompetenzen zu Algorithmen.* *Die Schülerinnen und Schüler …*	Stufe III *Die Schülerinnen und Schüler haben umfassendere Kompetenzen zu Algorithmen.* *Die Schülerinnen und Schüler …*
A Eigenschaften von Algorithmen	▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen ▷überprüfen die wesentlichen Eigenschaften von Algorithmen in einfachen Fällen ▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind	▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen an bekannten Beispielen ▷begründen anhand dieser Eigenschaften, ob gegebene Handlungsabläufe Algorithmen sind ▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind	▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen an selbst konstruierten Beispielen ▷begründen anhand dieser Eigenschaften, ob gegebene Handlungsabläufe Algorithmen sind ▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind
B Algorithmische Grundbausteine und Datentypen	▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen und Wiederholungen und wenden diese Erklärungen an ▷stellen die algorithmischen Grundbausteine als Pseudocode dar ▷verwenden einen numerischen Datentyp	▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen und Wiederholungen und wenden diese Erklärungen an ▷stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar ▷verwenden verschiedene Datentypen	▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen, Wiederholungen und Unterprogramme mit Parametern und wenden diese Erklärungen an ▷stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar und wechseln zwischen Darstellungsformen ▷verwenden verschiedene Datentypen
C Arbeit mit Algorithmen	▷lesen in Pseudocode gegebene einfache Algorithmen ▷prüfen schrittweise einfache Algorithmen mit gegebenen Beispielen ▷setzen gegebene einfache Algorithmen in Programme um ▷modifizieren und ergänzen einfache Algorithmen bzw. Programme nach Vorgaben	▷analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen ▷prüfen Algorithmen mit gegebenen Beispielen mithilfe von Durchlauftabellen (Schreibtischtest) ▷setzen gegebene Algorithmen in Programme um ▷modifizieren und ergänzen Algorithmen bzw. Programme nach Vorgaben	▷analysieren die Funktionsweise und den Leistungsumfang gegebener komplexer Algorithmen ▷prüfen Algorithmen mithilfe von Durchlauftabellen (Schreibtischtest) und wählen dazu typische und untypische Beispiele selbst aus ▷setzen gegebene komplexe Algorithmen in Programme um ▷modifizieren und ergänzen komplexe Algorithmen bzw. Programme nach Vorgaben und nach selbst gesetzten Zielen ▷korrigieren gegebene fehlerhafte Algorithmen bzw. Programme
D Programmentwicklung	▷entwerfen einfache Programme skizzenhaft ▷implementieren einfache Programme mit einem Programmiersystem ▷testen einfache Programme anhand gegebener Eingaben auf ihre Grundfunktionalität	▷fertigen einen schriftlichen Entwurf für Programme an ▷implementieren Programme mit einem Programmiersystem benutzungsfreundlich ▷testen Programme anhand gegebener Eingaben auf ihre Funktionalität ▷reflektieren über den Lösungsweg	▷fertigen einen schriftlichen Entwurf für komplexe Programme an ▷implementieren komplexe Programme mit einem Programmiersystem benutzungsfreundlich ▷testen komplexe Programme anhand selbst gewählter Eingaben auf ihre Funktionalität ▷reflektieren über den Lösungsweg sowie über Vor- und Nachteile der Lösung ▷verbessern Programme eigenständig

Abbildung 2.8.: Kompetenzstufenmodell Algorithmen [Kohl 2009, S. 93]

Neben dem praktischen Einsatz für einen kompetenzorientierten Informatikunterricht dienen jene Aufgabensammlungen aber auch zur schulübergreifenden Qualitätssicherung und -entwicklung. Dementsprechend sollte die Testentwicklung einem aus Perspektive der Fachwissenschaft, Fachdidaktik und pädagogisch-psychologischer Forschung professionellem Anspruch genügen. Ansonsten können schulübergreifende Vergleichstests „Gefahr laufen, mehr Fehlinformation und Schaden als Aufklärung und Orientierung zu erzeugen [Klieme et al. 2007, S. 82]."
Bei der Entwicklung von Testverfahren haben die Verwendungsziele einen Einfluss auf die Entwicklung von Aufgaben, die Testdurchführung und der Testauswertung. In diesem Zusammenhang lassen sich die folgenden Ziele unterscheiden [Klieme et al. 2007, S. 82ff]:

1. *Überprüfung von Kompetenzmodellen*
   Empirische Überprüfung, ob das jeweilige Kompetenzmodell *tatsächlich die Aspekte der Kompetenzen von Lernenden, ihre Niveaustufung und ggf. ihre Entwicklung angemessen widerspiegeln.*

2. *Systemmonitoring*
   Die Messverfahren dienen dazu, *Aussagen über das Kompetenzniveau von Lernenden zu machen und Zusammenhänge mit schulischen und außerschulischen Bedingungen aufzudecken.*

3. *Schulevaluation*
   Die Testverfahren werden zur Selbstevaluation eingesetzt, um zu prüfen, inwieweit Lehrkräfte oder Schulen ihre pädagogischen Ziele erreicht haben.

4. *Individualdiagnostik und Förderung einzelner Lernender*
   Der Einsatz der Testverfahren verfolgt die Zielsetzung, *Aussagen über spezifische Stärken und Schwächen und damit dem Förderbedarf einzelner* Lernender *zu machen.*

Das im Rahmen dieser Arbeit entwickelte Kompetenzmodell verfolgt die oben erläuterte erste Zielsetzung, nämlich die Evaluation ob das Kompetenzmodell für informatisches Modellieren tatsächlich jene Kompetenzen angemessen widerspiegelt.

Test- und Aufgabenentwicklung im Informatikunterricht

Kohl beschreibt die Entwicklung von Aufgaben, um jene Kompetenzen, die im Kompetenzmodell abgebildet sind zu überprüfen. Hierzu legt er zunächst Kriterien zur Konkretisierung der Aufgaben fest, nach denen Algorithmen den einzelnen Komponenten

und Stufen des Kompetenzmodells zugeordnet werden können. Davon ausgehend wurden konkrete Beispiel- und Testaufgaben entwickelt, die im Unterricht unter Verwendung der visuellen Programmiersprache PUCK [Kohl 2009] bearbeitet werden können. Hierbei ist anzumerken, dass die Aufgaben auch mit der Zielsetzung entwickelt wurden, um bei der gewählten Unterrichtsmethode handhabbar zu sein.
In den Abbildungen 2.9 und 2.10 wird eine exemplarische Aufgabe zur Überprüfung der Stufe I zur Komponente C des Kompetenzmodells und deren Lösung in der visuellen Programmiersprache PUCK dargestellt.

Unterrichtsaufgabe **Stufe I** **Komponente C**

Aufgabe (Gehaltserhöhung)

Die Softwarefirma IT-Triple konnte ihre Effizienz durch den Umstieg auf eine visuelle Programmiersprache steigern. Der Chef möchte deshalb seine Mitarbeiter belohnen. Die Gehälter aller Mitarbeiter sollen um 4 %, mindestens aber um 80 € im Monat erhöht werden.

Der Algorithmus zur Berechnung dieser Gehaltserhöhung ist gegeben:

ALGORITHMUS Gehaltserhoehung

Lege die Variable *altesgehalt* vom Typ Integer an.
Fordere den Benutzer mit "Geben Sie bitte ihr altes Gehalt ein." auf, eine Zahl einzugeben.
Speichere die eingegebene Zahl in die Variable *altesgehalt*.

Wenn die Bedingung *((altesgehalt*104 DIV 100)-altesgehalt > 80)* wahr ist, mache Folgendes:
  Gib "Neues Gehalt:" und danach den Wert von (*altesgehalt*104 DIV 100*) aus.
Wenn die Bedingung *((altesgehalt*104 DIV 100)-altesgehalt > 80)* falsch ist, mache Folgendes:
  Gib "Neues Gehalt:" und danach den Wert von *altesgehalt+80* aus.

a) Welche Ausgaben liefert der Algorithmus für das Gehalt 1000 € und für das Gehalt 10000 €?

Ausgabe:

b) Lesen Sie den Algorithmus, übertragen Sie ihn in ein Programm und testen Sie es.

c) Finden Sie durch mehrfaches Testen heraus, ab welcher Gehaltsgrenze mehr als 80 € Gehaltserhöhung fällig ist.

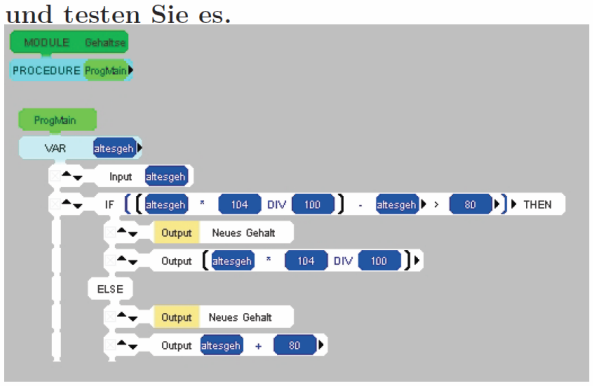
d) Ergänzen Sie das Programm so, dass jeweils die Gehaltserhöhung mit ausgegeben wird.

Abbildung 2.9.: Beispielaufgabe Kohl [Kohl 2009, S. 120]

Die Auswertung der einzelnen Aufgaben wurde anhand von Bewertungsmatrizen vorgenommen. Als Resümee nach Fertigstellung der Gesamtauswertung der Hauptuntersuchung an Thüringer Schulen formuliert Kohl seine Erfahrungen in Form von sechs Anforderungen an die Entwicklung zu Testaufgaben der Informatik [Kohl 2009, S. 183]:

1. Zwischen Beispiel-, Unterrichts- und Testaufgaben unterscheiden
2. Beispiel- und Testaufgaben anhand der im Kompetenzmodell geforderten Kompetenzen konstruieren und in klare, überschaubare Teilaufgaben untergliedern
3. Vielfältige, abwechslungsreiche Unterrichtsaufgaben zusammenstellen
4. Die Unterrichtsaufgaben in einem digitalen, einfach modifizierbaren Format bereitstellen

- **Welche Ausgaben liefert der Algorithmus für das Gehalt 1000 € und für das Gehalt 10000 €?**
  Gehalt 1000 → Neues Gehalt: 1080
  Gehalt 10000 → Neues Gehalt:10400
- **Lesen Sie den Algorithmus, übertragen Sie ihn in ein Programm und testen Sie es.**



- **Finden Sie durch mehrfaches Testen heraus, ab welcher Gehaltsgrenze mehr als 80 € Gehaltserhöhung fällig ist.**
  Ab der Eingabe 2025 sind mehr als 80€ Gehaltserhöhung fällig.
- **Ergänzen Sie das Programm so, dass jeweils die Gehaltserhöhung mit ausgegeben wird.**

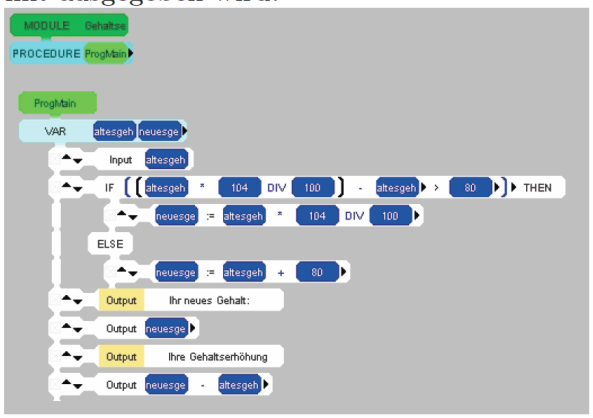


Abbildung 2.10.: Beispiellösung Kohl [Kohl 2009, S. 245]

5. In den Aufgaben nicht auf spezielle Software, Programmiersprachen oder Entwicklungsumgebungen eingehen

6. Aufgaben vor einem größeren Einsatz erproben

Kohls Erkenntnisse bei der Aufgabenentwicklung haben in bestimmten Bereichen die Aufgabenentwicklung im Rahmen des Projekts *MoKoM* und in dieser Dissertation beeinflusst. Hierdurch wurden wir beispielsweise in unseren Planungen bestärkt, die entwickelten Aufgaben vor dem breiten schulischen Einsatz mit Studierendengruppen und kleineren Schülergruppen zu erproben (siehe Kapitel 7). Ferner bestand bei der Entwicklung der Aufgaben und Items die Prämisse, diese möglichst unabhängig von bestimmten Technologien oder Programmiersprachen zu gestalten.
Ein wichtiger Unterschied bei der Gestaltung der Aufgaben und Items innerhalb des *MoKoM-Projekts* gegenüber Kohls Ansatz ist die Gestaltung der Aufgaben im Hinblick auf die zu überprüfenden Kompetenzen. Kohl schlägt vor, dass die Aufgaben entsprechend der im Kompetenzmodell modellierten Kompetenzen zu konstruieren und in klare Teilaufgaben zu zerlegen sind.

Im *MoKoM-Projekt* und im Rahmen dieser Dissertation wurde keine explizite Trennung der einzelnen Aufgaben entsprechend der einzelnen Kompetenzbereiche angestrebt, sondern die Zielsetzung verfolgt, Aufgaben und Items zu gestalten, die ein möglichst breites Spektrum an kognitiven und nicht kognitiven Fähigkeiten und Fertigkeiten abfragen. Eine im Hinblick auf den abzufragenden Kompetenzbereich zu eingeschränkt formulierte Aufgabenstellung, lässt sich aus Sicht des Autors nicht mit dem Weinert'schen Kompetenzverständnis vereinbaren. Wie erwähnt sieht jenes Kompetenzverständnis ein Hauptmerkmal von Kompetenzen in der Bewältigung komplexer Problemstellungen.

2.5. Zusammenfassung

Das Kapitel *Kompetenzorientierung als fachdidaktische Ausgangslage* beschreibt die bildungspolitische Ausgangslage als zentrale Motivation für den Forschungsgegenstand der vorliegenden Arbeit, nämlich der Entwicklung eines Kompetenzmodells für die informatische Modellierung und eines dazugehörigen Messinstruments als mögliche Grundlage für die Entwicklung von Bildungsstandards in der Sekundarstufe II.
Zu Beginn des Kapitels wird verdeutlicht, dass die Diskussion um Bildungsstandards und Kompetenzen sowohl national als auch international sehr relevant ist und Gegenstand zahlreicher Forschungsbeiträge, Veröffentlichungen und bildungsorganisatorischer Maßnahmen ist. Dies ist die zentrale Motivation und Bestärkung des Forschungsvorhabens, entsprechende Grundlagenarbeit für den Bereich der informatischen Modellierung im Projekt *MoKoM* und im Rahmen des Promotionsvorhabens zu leisten.
Darüber hinaus werden die Güte-Kriterien für die Formulierung nationaler Bildungsstandards gemäß Klieme-Expertise aufgeführt und die damit verbundenen Anforderungen an Kompetenzformulierungen. Dementsprechend wird hier festgelegt, unter welchem theoretischen Verständnis des Kompetenzbegriffs [Weinert 2002] die vorliegende Arbeit basiert und welche Vorgehensweise bei der Entwicklung von Kompetenzmodellen und entsprechenden Instrumenten richtungsweisend sind [Klieme 2004].
Weiterhin soll das Kapitel aufzeigen, dass der Gesamtprozess zur Entwicklung von Bildungsstandards nicht immer gemäß nationaler Vorgaben korrekt eingehalten wird. Nicht alle Ansätze, die sich Bildungsstandards nennen sind auch im Sinne der Klieme-Expertise. Dies motiviert das interdisziplinäre Vorgehen im Projekt *MoKoM* und in der vorliegenden Forschungsarbeit.
Weiterhin soll dieses Kapitel den Leser dahingehend sensibilisieren, dass die Verwendung des Kompetenzbegriffs nicht immer eindeutig ist. Es werden im zitierten Ansatz einzelne Fähigkeiten und Fertigkeiten beschrieben, die nicht dem Anspruch des Weinert'schen

Kompetenzverständnisses genügen. Dies zeigt wiederum, dass eine einheitliche Verwendung des Begriffs „Kompetenz“ unabdingbar ist.
Im weiteren Verlauf gibt das Kapitel Überblick über die verschiedenen Ansätze für Kompetenzmodelle in der Informatik und stellt exemplarisch Kohls Ansatz als Kompetenzmodell für Algorithmen in der Sekundarstufe I vor. In diesem Zusammenhang soll deutlich gemacht werden, dass dieser Ansatz teilweise den Vorgaben der Klieme-Expertise widerspricht und den *MoKoM-Ansatz* bestärkt. Dieser sieht wie in der Einleitung der vorliegenden Arbeit beschrieben, zunächst die Entwicklung eines interdisziplinären und empirisch abgesicherten Kompetenzmodells vor und zeigt die Notwendigkeit, Grundlagenarbeit für den informatischen Themenbereich Modellierung zu betreiben.
Eine weitere wichtige Begründung für dieses Kapitel war die Darstellung der Forderung, dass Kompetenzmodelle durch entsprechende Messverfahren in Form von Aufgaben und Items unterstützt werden sollen. Hier war es hilfreich, einen exemplarischen Ansatz zur Aufgabenentwicklung auf Grundlage eines bestehenden Kompetenzmodells für einen Aufgabenbereich der Informatik zu bewerten. Hierbei konnten wertvolle Hinweise für die eigene Forschungsarbeit gewonnen werden. Ferner wurde wiederum eine Uneinigkeit hinsichtlich des Kompetenzverständnisses festgestellt. In diesem Zusammenhang wurde aus Sicht des Autors eine zu isolierte scharfe Trennung und Zuordnung von Aufgaben nach zu überprüfenden Kompetenzen vorgenommen. Diese ist nicht im Sinne des Weinert'schen Kompetenzverständnis und hat uns in unserer Forschung dazu bewogen, bei der Aufgabenentwicklung darauf zu achten, ein möglichst breites Spektrum von kognitiven und nicht kognitiven Kompetenzbereichen anhand eines Beispiels aus der Lebenswelt der Probanden zu adressieren.

3. Modellierung im fachwissenschaftlichen und fachdidaktischen Kontext

Nach dem im vorherigen Kapitel die Relevanz der Diskussion um Kompetenz als Grundlage für Bildungsstandards deutlich gemacht wurde, soll in diesem Kapitel die Wahl des Gegenstandsbereichs der *objektorientierten Modellierung* begründet werden.
Dementsprechend verfolgt das Kapitel die Zielsetzung, die objektorientierte informatische Modellierung aus fachwissenschaftlicher und fachdidaktischer Perspektive zu erörtern und dessen Relevanz als wichtigen informatischen Inhaltsbereich aufzuzeigen. Neben dieser Legitimation besteht die Absicht, eine theoretische Grundlage für die normativ-theoretische Entwicklung eines Kompetenzmodells in Kapitel 4 zu schaffen. Es gilt ebenso, die theoretische Basis für die Entwicklung einer entsprechenden Unterrichtsreihe zur Erprobung des Messinstruments aufzuzeigen und festzulegen.
Im Verlauf des Kapitels werden nach einer Definition des informatischen Modellbegriffs und der Fokussierung auf die objektorientierte Modellierung, verschiedene Ansätze zur Modell-Kategorisierung aufgezeigt. Hierbei besteht die Zielsetzung, eine strukturgebende theoretische Basis für die Entwicklung von Kompetenzkomponenten zur objektorientierten informatischen Modellierung zu recherchieren und als theoretische Grundlage für die Kompetenzmodellierung und Instrumentenentwicklung festzulegen.
Hierbei soll gezeigt werden, inwieweit sich Vorgehensmodelle in der Softwaretechnik (insb. der *Rational Unified Process (RUP)*) hilfreich sein können, um auf deren normativer Grundlage Kompetenzaspekte abzuleiten.
Im Sinne des Weinert'schen Kompetenzverständnisses und den Gütekriterien der Klieme-Expertise sind fachdidaktische Aspekte bei der Formulierung von Kompetenzen mit zu berücksichtigen. Demgemäß sollen in einem weiteren Unterkapitel didaktisch motivierte Vorgehens- und Vermittlungsmodelle zur objektorientierten Modellierung (insb. im Bereich der Robotik) als theoretischer Ausgangspunkt zur Konzeption einer Evaluations-Unterrichtsreihe für die Kompetenzmessinstrumente vorgeschlagen und begründet werden.

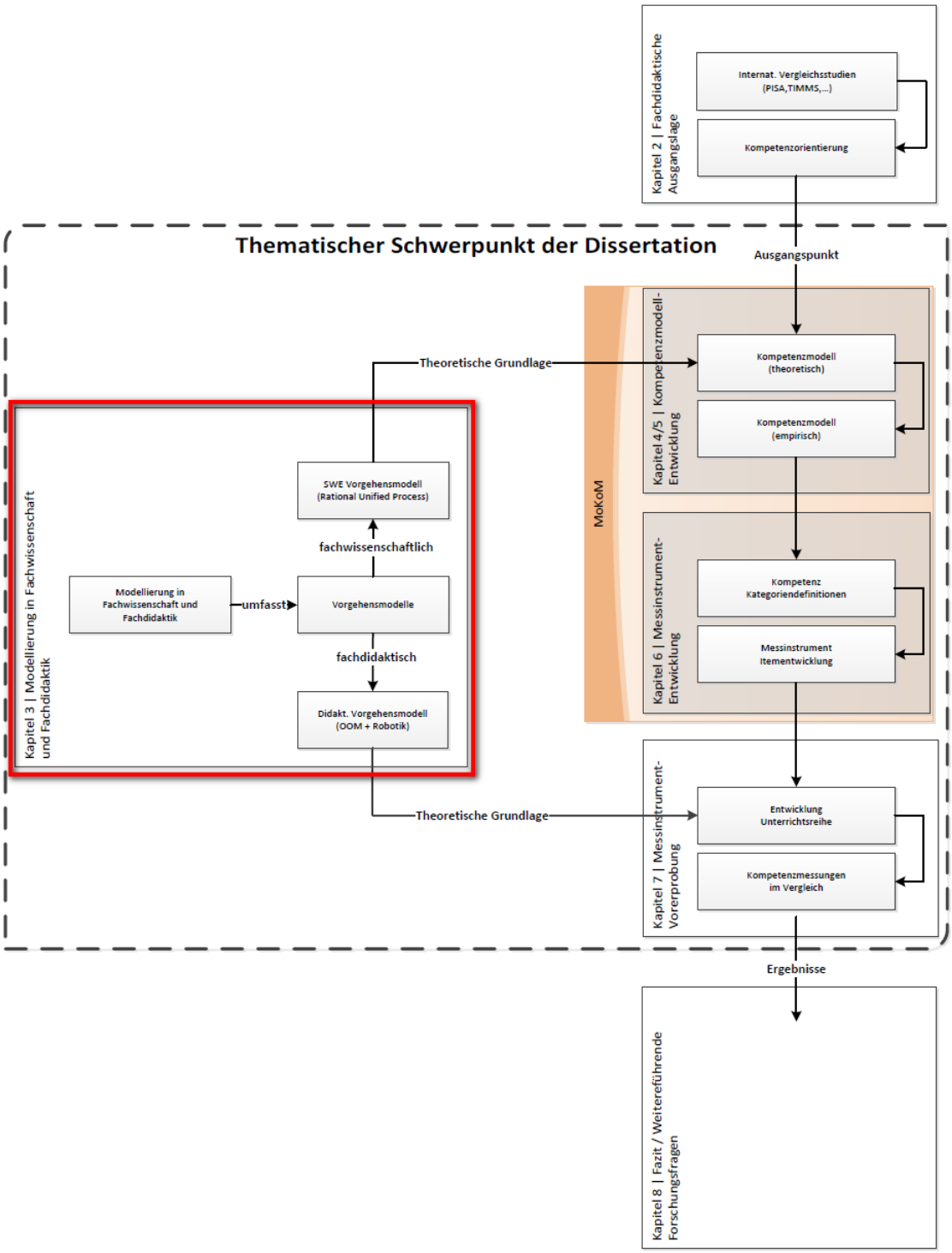


Abbildung 3.1.: Kapitel 3 im Gesamtkontext der Arbeit

3.1. Begriffsdefinition und Fokussierung

Nach Glinz (fachwissenschaftliche Perspektive) und Thomas (fachdidaktische Perspektive) sind informatische Modelle in Anlehnung an die *Allgemeine Modelltheorie* nach Stachowiak durch die folgenden Merkmale gekennzeichnet [Stachowiak 1973, S. 131ff], [Thomas 2002, S. 27]

- *Abbildungsmerkmal:*
  Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.

- *Verkürzungsmerkmal:*
  Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant erscheinen.

- *Pragmatisches Merkmal:*
  Modelle sind in ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion für bestimmte - erkennende und/oder handelnde, modellbenutzende - Subjekte, innerhalb bestimmter Zeitintervalle und unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen.

Insbesondere die letzte Eigenschaft zeigt, dass Modelle kontextualisiert für einen bestimmten Verwendungszweck entwickelt werden [Glinz 2008, S. 425-426].
Über die allgemeine Modelldefinition hinaus sieht Glinz spezifische Eigenschaften informatischer Modelle. Entgegen dem Modellverständnis nach Stachowiak, (definiert durch das Verkürzungsmerkmal und das pragmatische Merkmal) versteht Glinz die Abstraktion nicht als alleiniges konstituierendes Merkmal informatischer Modelle.

> "Ein Modell als Abstraktion eines Originals zu definieren, greift jedoch zu kurz: Zeichen beispielsweise sind Abstraktionen, aber keine Modelle [Glinz 2008, S. 426]."

Ferner sieht er eine Abgrenzung gegenüber dem Modellbegriff der mathematischen Logik. Hier wird eine Menge von Axiomen als Modell jenes Axiomsystems bezeichnet. Derartige Modelle werden von dem hier betrachteten Modellbegriff ausgeschlossen und finden auch in der vorliegenden Dissertationsschrift keinerlei Berücksichtigung. Vielmehr sollen Modelle als Informatikartefakte oder als Mittel zum Verstehen von *Informatikartefakten* betrachtet werden [Glinz 2008, S. 1].

Thomas sieht informatische Modelle als kulturell-tradiertes Bildungsgut mit dem sich Schülerinnen und Schüler auseinander setzen sollten. Er begründet diese These unter anderem damit, dass sich die Allgemeinbildungsbegriffe von Klafki [Klafki 2007] und Bussmann u. Heymann [Bussmann und Heymann 1987] auf informatische Modelle anwenden lassen. Ferner untermauert er die Forderung damit, dass dem Schüler anhand der Systematiken zu den informatischen Modellen und deren Entstehungsprozess ein vollständiges Bild der gesamten Informatik vermittelt werden kann [Thomas 2002, S. 81].
Die vorliegende Arbeit fokussiert bei der Entwicklung eines Kompetenzstrukturmodells und eines Kompetenzmessinstruments die objektorientierten Modellierung als Gegenstandsbereich.
Im Folgenden wird die Relevanz der objektorientierten informatischen Modellbildung aus fachwissenschaftlicher und fachdidaktischer Perspektive erörtert. Ferner soll deutlich gemacht werden, warum gerade dieser Bereich der informatischen Bildung als Gegenstandsbereich für die Kompetenzmodellierung in der vorliegenden Dissertationsschrift thematisiert wird.

3.2. Relevanz informatischer Modelle

Relevanz aus fachwissenschaftlicher Perspektive

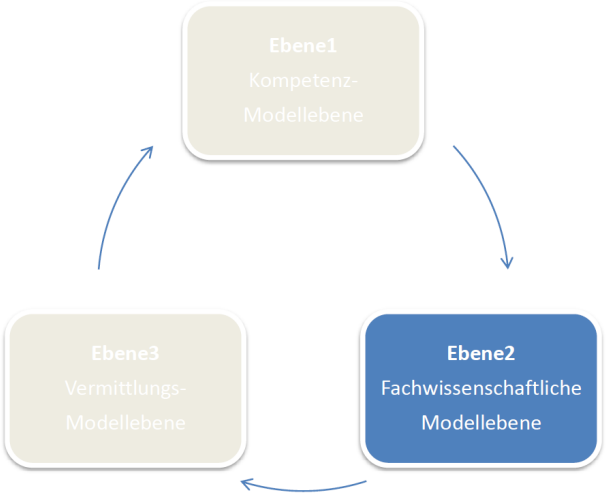


Abbildung 3.2.: Modellebene 2 - Fachwissenschaftliche Modellebene

Das Verstehen und Lösen komplexer Probleme macht einen wesentlichen Teil informatischer Kompetenz aus. Die Modellbildung ist eine unentbehrliche kognitive Fähigkeit

zur Bewältigung von Komplexität (Hesse, 2008) und hat innerhalb der Informatikstudiengänge einen hohen Stellenwert. Dies wird auch durch einschlägige Empfehlungen für Informatik-Curricula bestätigt [Joint Task Force on Computing 2001], [Gesellschaft für Informatik e.V. (GI) (Hrsg) 2004], [Fakultätentag Informatik (Hrsg) 2004].
Insbesondere im Rahmen der Softwaretechnik kommt der Modellierung seit den 1990er Jahren durch Aufkommen der objektorientierten Analyse- und Designtechniken und der *Unified Modeling Language (UML)* ein hoher Stellenwert zu.

> „Modelle sind die Artefakte der Softwareentwicklung [Ebert 2005]."

Spätestens aus Perspektive des *Model Driven Development* und der *Model Driven Architecture (MDD/MDA)* hat sich die zentrale Bedeutung der Modellierung im Rahmen der Softwaretechnik weiter verfestigt [Hesse und Mayr 2008].
Vor allem diese Aussage bekräftigt die enorme Wichtigkeit der objektorientierten Modellierung für die Informatik und motiviert die Entwicklung eines Kompetenzstrukturmodells für diesen Bereich der Informatik voranzutreiben.
Fieber, Huhn und Rumpe sehen die Qualität informatischer Modelle sogar als Indikator für Softwarequalität. Sie ermöglichen eine

- Struktur- und Schnittstellenbeschreibung,
- konstruktive Verhaltensbeschreibung, typischerweise von Zustand und Funktion,
- deskriptive Kommunikationsprotokolle und -mechanismen,
- Darstellung der logischen sowie physischen Verteilung,
- (meist informelle) Organisation und Strukturierung der Anforderungsbeschreibung,
- Modellierung von Aufgaben- und Prozessabläufen und
- Datenmodellierung.

Einen besonderen Stellenwert messen sie in diesem Zusammenhang der *Unified Modeling Language (UML)* bei. Diese unterstützt den SWE-Prozess mit ihren 13 Modellarten am umfassendsten [Fieber et al. 2008, S. 408]. In ihrem Beitrag geben sie einen Überblick, was Modellqualität ausmacht und leisten den Transfer von selbiger zu Softwarequalität. Sie postulieren, dass die Qualität der Modellbildung einen Einfluss auf die Qualität der Software und die Planbarkeit des SWE-Prozess hat.

„Allgemein anerkannt ist, dass adäquate Modellbildung und Analyse der Fähigkeiten des Modells sowie die Extrapolation dieser Eigenschaften auf das zu bildende Softwaresystem einen deutlichen Qualitätsvorteil und Planungssicherheit liefern können [Fieber et al. 2008, S. 422].“

Relevanz aus fachdidaktischer Perspektive

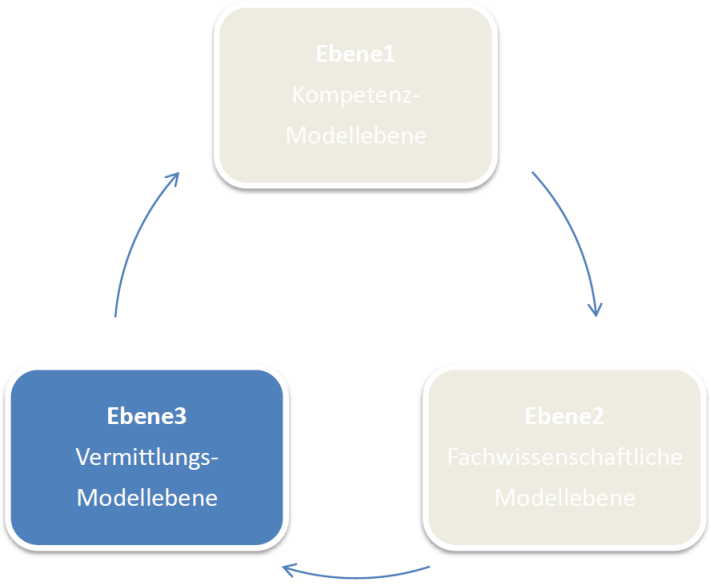


Abbildung 3.3.: Modellebene 3 - Vermittlungs-Modellebene

Im Folgenden sollen mögliche Ansätze zur Legitimation der Modellbildung im Informatikunterricht der Sekundarstufe aufgeführt werden:
Thomas zeigt unter Verwendung der allgemeinen Modelltheorie nach Stachowiak und weiteren Klassifikationsansätzen die inhaltliche Relevanz von Modellen in der Informatik [Thomas 2002, S. 3]. Er verdeutlicht, dass informatische Modelle ein Bildungsgut mit Bildungswert darstellen, indem er sie hinsichtlich der Allgemeinbildungskriterien von Klafki und Bussmann/Heymann untersucht. Er sieht seine Arbeit als Ausgangspunkt für weitere Forschungsfragen und fordert, dass weitere Untersuchungen unter Einbeziehung weiterer Merkmale zur Allgemeinbildung durchgeführt werden müssen [Thomas 2002, S. 80]. Er bewertet seinen Ansatz der Zuordnung von informatischen Modellen als eine Möglichkeit, den Informatikunterricht im allgemeinbildenden Schulchanon zu legitimieren.
Brinda legitimiert die Einbeziehung von objektorientierter Modellierung in den Informatikunterricht anhand der vier Fundamentalitätskriterien von Schwill [Brinda 2004,

S. 40-41]. Er sieht das Horizontalkriterium als erfüllt, da objektorientierte Sichtweisen in verschiedenen Teilgebieten der Informatik etabliert sind. Weiterhin sieht er das Vertikalkriterium als erfüllt, da sich Inhalte aus dem Bereich der Objektorientierung auf verschiedenen Bildungsniveaus vermitteln lassen. Das Zeitkriterium ist erfüllt, da erste Ansätze zur Objektorientierung bereits in den 1960er Jahren entwickelt wurden. Da sich objektorientierte Sichtweisen gut als Erklärungsmodell für informatische Erscheinungen eignen, ist das Sinnkriterium ebenfalls erfüllt.
Darüber hinaus statuiert Magenheim, dass die Thematisierung der objektorientierte Modellierung sogar das Potential hat, den Informatikunterricht an sich an allgemeinbildenden Schulen zu legitimieren: Nach Magenheim verlangen unterschiedliche Diskussionsbeiträge zur Didaktik der Informatik, dass eine Konzentration auf elementare Prinzipien der Bezugswissenschaft Informatik zu leisten sei. Insbesondere bei der Inhaltsbestimmung und Zielsetzung des Informatikunterrichts sollten keine kurzlebigen Modetrends verfolgt werden. Hierbei nennt er explizit die Prinzipien der Systemgestaltung sowie Modellierung. Hierbei gibt er gleichzeitig zu bedenken, dass auch Themenbereiche, wie z.B. die Kommunikation in vernetzten Umgebungen und die Bewertung des Einsatzes von Informatiksystem im sozialen Kontext im Sinne eines allgemeinbildenden Anspruchs des Informatikunterrichts mit einzubeziehen sind [Hampel et al. 1999].
Fachwissenschaftliche Argumente für eine inhaltlich dauerhafte Etablierung von Objektorientierung sind die Sicherheit und Stabilität von Software, leichte Wartbarkeit sowie sich daraus ergebende Vorteile bei Wiederverwendbarkeit und Sicherheit. Insbesondere in komplexen Softwareprojekten zeigen objektorientierte Maximen wie Kapselung, Abstraktion sowie Vererbung und Polymorphie ihre Vorteile gegenüber imperativen Konzepten. Der Ansatz der systemorientierten Didaktik verlangt somit eine Thematisierung sowohl im wissenschaftspropädeutischen als auch im allgemein bildenden Kontext [Hampel et al. 1999, S. 17].

Zwischenfazit

In Anbetracht dieser fachdidaktischen Beiträge zur informatischen Modellierung lässt sich feststellen, dass die objektorientierte Modellierung sowohl für die Informatik als Wissenschaft als auch für den Informatikunterricht an allgemeinbildenden Schulen von hoher Relevanz ist.

3.3. Ansätze zur Klassifikation von informatischen Modellen

Dieses Kapitel gibt einen exemplarischen Einblick in verschiedene Ansätze zur Klassifikation von informatischen Modellen. Hierbei werden ein fachwissenschaftlicher Ansatz und ein fachdidaktisch motivierter Ansatz dargestellt. Zielsetzung des Kapitels ist es, eine mögliche Vorstrukturierung für die Dimensionen und Komponenten des Strukturmodells aufzufinden. Auf dieser Grundlage sollen inhaltliche Themenbereiche ermittelt werden, die ggf. Hinweise auf typische Kompetenzen für die objektorientierte informatische Modellierung geben können.

Modelle in der Softwaretechnik nach Hesse/Mayr

Hesse und Mayr erörtern zunächst den Modellbegriff in der Softwaretechnik (ausgehend von Stachowiak) und geben einige Klassifizierungen und Charakterisierungen von Modellen und Modellierungen innerhalb der Softwaretechnik vor. Diese orientieren sich vorwiegend an der Art der modellierten Gegenstände und ob diese statischer oder dynamischer Natur sind. Hierbei liegt ein systemtheoretischer Ansatz zugrunde, der auch als eine theoretische Basis bei der Entwicklung von Vorgehensmodellen innerhalb der Softwaretechnik zu verstehen ist [Hesse und Mayr 2008].

> "Den meisten genannten Modellkategorien liegt ein systemtheoretischer Ansatz zugrunde, der bei der Behandlung diskreter dynamischer Strukturen – wie in der Softwaretechnik üblich – nahe liegt. Er wurde daher schon relativ frühzeitig in der Softwaretechnik aufgegriffen [...]. Der systemtheoretische Ansatz wird z.B. bei den o.g. Prozessmodellen [Wasserfallmodell, Rational Unified Process] besonders deutlich, wo auch Prozesse als Systeme – bestehend aus Komponenten und verbunden durch Beziehungen – aufgefasst werden [Hesse und Mayr 2008, S. 383]."

Im Folgenden werden die oben erwähnten Modell-Klassifikationen und -Charakteristika aufgeführt.

- *Statikmodelle:* Modellelemente beschreiben zu einem bestimmten Zeitpunkt beobachtbare oder beobachtbar gedachte Konstellationen von Gegenständen, Beziehungen und sonstigen beschreibenden Elementen.
  - *Gegenstands-, Struktur-, Entitäts- und Klassenmodelle:* Trotz teilweise dynamischer Elemente (z.B. Methoden in Klassendiagrammen) ist die statische Betrachtungsweise bestimmend.

- *Dynamikmodelle:* Modellelemente repräsentieren einen Vorgang, eine Aktion oder einen Prozess. Diese sind zeitlichen Veränderungen unterworfen.
  - *Vorgehens-, Aktions- und Prozessmodelle:* Hierbei geht es vorrangig um die Betrachtung mehrerer - nicht an einen Zeitpunkt gekoppelter - Konstellationen und Verläufe.
  - *Zustandsmodelle:* Bei diesem Modelltyp werden Objekte und Klassen gleichartiger Objekte betrachtet. Zustände beschreiben hierbei die Konstellation von Objekten, wo hingegen Zustandsübergänge deren dynamische Zusammenhänge darstellen.

Kategorisierung aus fachdidaktischer Perspektive nach Thomas

Thomas untersuchte Skripte aus der Hochschullehre innerhalb verschiedener Teildisziplinen der Informatik. Hierbei hat sich ergeben, dass das Wort „Modell“ in 83% der untersuchten Skripte in irgendeiner Flexionsform verwendet wurde [Thomas 2002, S. 48]. Auch innerhalb dieser fachdidaktischen These wird die inhaltliche Relevanz der informatischen Modellierung offensichtlich.

> „Schließt man von der Auftrittshäufigkeit des Wortes Modell auf das Verwenden von Modellen in der Informatik, so ist dem Modell offensichtlich ein zentraler Stellenwert innerhalb der Fachwissenschaft zuzuordnen [Thomas 2002, S. 48].“

Die Begriffe „modelliert“ oder „modellieren“ tauchten mit einer Häufigkeit von 61% auf. Anhand sog. Modifizierer, die dem Begriff “Modell“ hinzugefügt werden, illustriert Thomas die Vielfältigkeit des Modellbegriffs in der Informatik anhand unterschiedlichster Modellbezeichnungen. Hierbei erfolgt die Bezeichnung des Modells anhand einer Metapher (z.B. Call-Back-Modell, Schichten-Modell oder Master-Slave-Modell), anhand eines Autors (z.B. Markov-Modell, oder Hufmann-Modell) oder dem jeweiligen Einsatzgebiet des Modells (z.B. Speicher-Modell, Farb-Modell, Simulations-Modell etc.).
Er versteht jene Modifizierer allerdings auch als Mittel zur Präzisierung des Modelltyps [Thomas 2002, S. 49]:

- nach Art der Zustandsübergänge
  statisch, dynamisch (diskret, kontinuierlich, deterministisch, stochastisch, nichtdeterministisch), analog, ereignisorientiert.

- nach der Sichtweise der Modellierung: abstrakt, objektorientiert, strukturbasiert, zustandsorientiert, verhaltensbasiert, eigenschaftsbasiert, kompositional, statistisch, perzeptuell, kognitiv, handlungspsychologisch, minimal.
- nach der inneren Modellstruktur: analytisch, applikativ/funktional, (nicht-)logisch, assoziativ, hierarchisch, taxonomisch, relational, geschlossen, parallel, ökonomisch, konzeptuell.
- nach der Natur des Modells: (nicht)materiell, mechanisch, formal, operational, mathematisch, semantisch, implementiert, mental, physiologisch, visuell, parametrisiert, physikalisch, kinematisch.

Anhand der aus den Skripten aufgefundenen Modifizierer und Modelltypen leitet Thomas eine Kategorisierung in Form von fünf Hauptmodelltypen ab [Thomas 2002, S. 49ff]

1. Architekturmodelle
   - (theoretisches) Maschinenmodell
     - hardware-orientierte Konzepte von Rechnerarchitekturen
     - abstrakte, theoretische Automaten (Turingmschine, Automatenmodelle)
   - Rechenmodell
     - grundlegende Konzepte, die Programmiersprachen zugrunde gelegt werden (imperativ-prozedural, funktional applikativ, logisch deklarativ, objektorientiert, zustandsorientiert)
   - Programmiermodell
     - im Sinne eines Programmierparadigmus
     - als abstraktes Maschinenmodell
     - als grundlegendes Konzept für die Interaktion von parallelen Prozessen (teilweise als Kommunikationsmodell bezeichnet)
   - Referenzmodell
     - beschreiben Vereinbarungen zu technischen Konzepten und Prinzipien
     - OSI-Referenzmodell
     - Client-Server Modell
     - Farbmodell
2. Vorgehensmodelle
   - beschreiben Aktivitäten, die auszuführen sind, um ein bestimmtes Ziel zu erreichen
   - enthalten Hinweise zu erforderlichen oder zu erstellenden Dokumenten
   - enthalten Ziele zu einzelnen Phasen/Arbeitsschritten
   - geben Hinweise auf einsetzbare Verfahren und Hilfsmittel

- beruhen i.d.R. auf bestimmten Sicht- oder Denkweisen (z.B. für den Gesamtprozess der SW-Entwicklung oder für einzelne Phasen)

3. Entwurfsmodelle
   - stellen die Dokumentation von Ergebnissen der Aktivitäten bei der Erstellung eines konkreten technischen Problems dar
   - können in natürlicher oder grafisch-symbolischer Art und Weise formuliert werden
   - Systemmodell
     - Verwendung in der Automatentheorie in Form einer Mengenstruktur
     - Verwendung als Modell beliebigen Typs zu einem entwerfenden System
   - Modellierungssprache
     - allgemeine Beschreibungsformen, die in verschiedenen Phasen der Softwareentwicklung zum Einsatz kommen
     - Modelle für die Entwurfsphase der Softwareentwicklung
     - *Unified Modeling Language (UML)*: Sprache zur Veranschaulichung von Ergebnissen im objektorientierten SWE-Prozess; besteht aus verschiedenen Diagrammtypen zu verschiedenen Sichtweisen auf das System
     - zu jeder Modellierungssprache existiert i.d.R. ein Metamodell; jenes beschreibt Syntax, Semantik und Pragmatik der Sprache oder fasst gemeinsame Komponenten von Modellen zusammen
   - Aufgabenmodell
     - wird basierend auf dem Ergebnis der Analyse einer Aufgabe erstellt
     - enthält Angaben zur Reihenfolge von Arbeitsabläufen
     - relevant innerhalb der Problemanalyse für der Entwurfsphase im SWE-Prozess
   - Daten(bank)modell
     - Beschreibung von Daten und ihrer strukturellen/funktionalen Beziehungen untereinander
     - Relationenmodell/Relationales Datenmodell: alle Daten werden in Form von mathematischen Relationen repräsentiert, z.B. Entity Relationship Modell (ER-Modell), Normalform-Modell
     - Hierarchisches Modell und Netzwerkmodell: sind im Gegensatz zum relationalen Datenmodell nicht mengen- sondern satzorientiert, d.h. um einen Datensatz zu erreichen muss innerhalb der zugrundeliegenden Struktur von Datensatz zu Datensatz navigiert werden.
     - Logisches Datenmodell oder Deduktives Modell: Erweiterung des relationalen Datenmodells um Deduktionskomponente (auf dem Prädikatenkalkül basierend)
     - Objektorientiertes Modell: strukturelle Repräsentationen werden mit der verhaltensmäßigen (operativen) Komponenten in einem Objekt verknüpft

- Objektmodell
  - im Softwareentwurf werden die identifizierten Komponenten eines betrachteten Originals in meinst grafisch-symbolischer Form beschrieben
- Komponentenmodell
  - Komponente als Teil von (wiederverwendbarer) Software, das eine zusammenhängende Funktionalität hat
  - Definition der einzelnen Komponenten eines Systems
  - Konfiguration eins Systems aus bereits gegebenen Komponenten
- Funktionales Modell
  - dient als Ausgangspunkt für die Entwicklungsphase
  - es werden Objekte ausgewählt und deren Methoden bestimmt
- Prozessmodell
  - Beschreibung der Prozessverwaltung und Implementierung bei Betriebssystemen
  - Beschreibung des zeitlichen Verhaltens eines Produktionsprozesses
  - kognitionspsychologisches Modell zur Beschreibung des Sprechens im Teilgebiet der künstlichen Intelligenz
- Zustandsmodell
  - Beschreibung der möglichen Zustände, die Automaten oder Prozesse einnehmen können
- Ereignismodell
  - Beschreibung wie bei Benutzungsschnittstellen Ereignisse bearbeitet werden
  - Steuerung der Ereignisbehandlung

4. Untersuchungsmodelle
   - dienen zur Erstellung von Prognosen für Informatiksysteme, Bewertung von Systemen zu unterschiedlichen Kriterien (Leistung, Kosten, Auslastung)
   - zumeist in formaler mathematischer Notation beschrieben
   - sind von starker Abstraktion, um auch quantitative Aussagen erzielen zu können
   - Mathematisches Modell: Beschreibt mit abstrakten Symbolen und Notationen Objekte und deren Zusammenhänge
     - Minimales Modell: Beschreibung im Sinne der mathematischen Modelltheorie
     - Abstraktes und formales Modell: umfassen mathematische Modelle und Graphen; keine abstrakten Datentypen
     - Model Checking: Beweisverfahren zur Verifikation, dass ein bestimmtes Modell eine Spezifikation erfüllt

- Analytisches Modell: gehen oft von Voraussetzungen aus, die ein System nicht erfüllt; das Systemverhalten wird durch mathematische Größen und Beziehungen untereinander beschrieben; im Gegensatz zu numerischen Simulationsmodellen ergeben sich Zusammenhänge direkt aus dem Modell
  - Stochastisches Modell: berücksichtigen zufällig auftretende Ereignisse
  - Kostenmodell: dienen zur Aufwandsabschätzung (z.B. bei Datenbankzugriffen)
- Simulationsmodell (simulatives Modell): beschreiben in statischer Weise vorrangig Systemverhalten mittels mathematischer Größen; enthalten aber auch Variablen, die sich in Abhängigkeit von der Zeit dynamisch ändern
  - Diskretes Modell: Modelle, die ihren Zustand nur zu bestimmten Zeiten ändern
  - Kontinuierliches Modell: Darstellung der Zeit im Modell erfolgt in Form realer Werte; Zustandsgrößen sind meist stetige Funktionen der Zeit; können typischerweise in Form von Differentialgleichungssystemen beschrieben werden.

5. Mentale Modelle
   - interne semantische Modelle, die externen semantischen Modellen (z.B. Vorgehensmodellen oder Entwurfsmodellen) zugrunde liegen; viele psychologische Effekte sind für die Softwaretechnik von Bedeutung
     - Konzeptuales Modell: Menschen bilden sich konzeptuale (begriffliche) Modelle von Objekten, die zu bedienen sind und innerhalb derer (sichtbare und unsichtbare) Operationen ablaufen; wird verwendet, um Operationen auf dem Objekt mental zu simulieren
     - Modellwelt: Verwendung im Sinne einer Sichtweise in der Modellierung; im Bewusstsein der Abgrenzung zur Realität und zu anderen Modellen
     - Modellvorstellung: stehe i.d.R. für Ziele, die das Subjekt mit dem Modell gedanklich verbindet
     - Modellklasse: Modelle werden aufgrund von Gemeinsamkeiten zu Modellklassen vereint; Klassen werden in der objektorientierten Modellierung zu Modellklassen zusammengeführt

Nach Betrachtung der verschiedenen Modelltypen eignen sich aus Sicht des Autors vor allem Vorgehensmodelle als strukturgebende theoretisch-normative Grundlage zur Beschreibung von Kompetenzen für die objektorientierte informatische Modellierung.
Nach Thomas beschreiben Vorgehensmodelle Aktivitäten, die auszuführen sind, um ein bestimmtes Ziel zu erreichen. Ferner beschreiben sie Zielsetzungen für einzelne Phasen und Arbeitsschritte und geben gleichzeitig Hinweise auf einsetzbare Verfahren und Hilfsmittel [Thomas 2002, S. 52].
Diese Modellcharakteristik ist unter Berücksichtigung des Weinert'schen Kompetenzverständnis [Klieme 2004] stimmig um als strukturgebende Grundlage für die Formulierung

von Kompetenzen zu fungieren. Vorgehensmodelle beschreiben erforderliche kognitive und nicht-kognitive Fähigkeiten und Fertigkeiten (in Form von Aktivitäten), die ein Individuum zur Problemlösung in variablen Anforderungssituationen einsetzen kann. Die zu bewältigenden variablen Anforderungssituationen werden innerhalb der Vorgehensmodelle durch die unterschiedlichen Phasen der Softwareentwicklung ausgedrückt.

3.4. Informatische Vorgehensmodelle als strukturgebende theoretische Grundlage

3.4.1. Vorgehensmodelle in der Softwaretechnik

Nach Sichtung der unterschiedlichen Klassifikations- und Kategorisierungsansätze für informatische Modelle, scheinen informatische Vorgehensmodelle in der Softwaretechnik als geeignete theoretische Grundlage zur Strukturierung informatischer Kompetenz fungieren zu können.
Bevor eine Vorstellung verschiedener Vorgehensmodelle innerhalb der Softwareentwicklung erfolgt, ist es sinnvoll den Begriff „Softwaretechnik“ zu definieren.

> „Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen [Balzert 2000, S. 36].“

Balzert betont in seinem Lehrbuch, dass die Softwaretechnik stets die Zielsetzung verfolgt, umfangreiche Software, die arbeitsteilig und ingenieurmäßig entwickelt wird, um die Ziele des Kunden bzw. Auftraggebers zu erreichen. Hierbei werden Prinzipien (z.B. Abstraktion, Strukturierung, Hierarchisierung, Modularisierung) und Methoden (d.h. planmäßig angewandte, begründete Vorgehensweisen) zur Erreichung von festgelegten Zielen verwendet [Balzert 2000, S. 36].
Im Folgenden werden die wesentlichen Phasen der Softwareentwicklung, die in jedem Softwareprojekt (Neu- oder Weiterentwicklung) und unabhängig vom gewählten Vorgehensmodell relevant sind, aufgeführt [Kleuker 2011, S. 24ff].

- *Anforderungsanalyse*
  Die Zielsetzung dieser Phase ist es zu verstehen, welche Ziele und Ergebnisse der Kunde wünscht. Hierbei kommt dem Dialog zwischen Kunden und Softwareentwicklern eine hohe Bedeutung zu. Die Qualität dieser Phase (und dieser Kommunikation) beeinflusst maßgeblich das Gelingen eines Softwareentwicklungs-Projekts.

- *Grobdesign*
  Die Phase des Grobdesigns fokussiert die Verwandlung der zuvor aufgenommenen Anforderungen in ein unmittelbar für die Softwareentwicklung einsetzbares Modell zu verwandeln. Hierbei gilt es, informelle Anforderungen zu präzisieren und eine grundsätzliche Software-Architektur festzulegen.

- *Feindesign*
  Im Rahmen dieser Phase erfolgt die Verfeinerung und Optimierung der Modelle des Grobdesigns. In diesem Kontext werden genaue Schnittstellen zwischen den verschiedenen Software-Komponenten definiert und das Design, welches hierbei als innere Struktur der Software zu verstehen ist, entwickelt.

- *Implementierung*
  In der Implementierungsphase erfolgen die Programmierung der Software und die Umsetzung der Modelle aus der Feindesign-Phase. Ein wesentlicher Meilenstein dieser Phase ist die Vorlage einer lauffähigen Software.

- *Test und Integration*
  Hierbei erfolgt die Zusammensetzung der einzelnen Programmkomponenten zu einem Softwaresystem, dem eigentlichen Software-Produkt. In dieser Phase muss zudem sichergestellt werden, dass die Software-Komponenten korrekt miteinander agieren und die Anforderungen des Kunden an das Produkt umgesetzt wurden.

- *Qualitätssicherung*
  Die Qualitätssicherung ist eng mit den oben dargestellten Phasen verknüpft. Hierbei muss für jedes Teilprodukt innerhalb des SWE-Prozess sichergestellte werden, dass die vor dem Projekt definierten Qualitätskriterien erfüllt sind. Folgephasen dürfen erst dann beginnen, sobald die jeweiligen Qualitätskriterien der Vorphase erreicht sind.

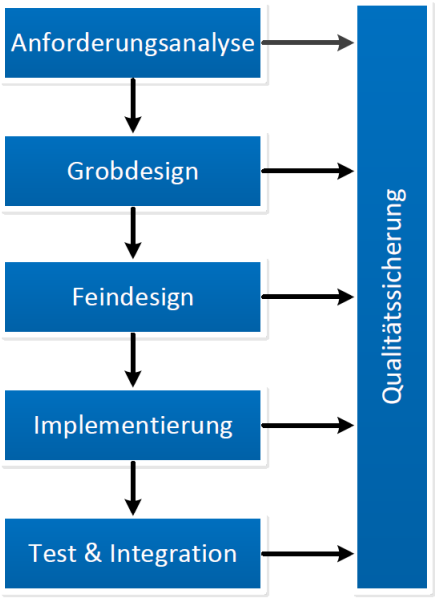


Abbildung 3.4.: Allgemeine Phasen des Software Engineerings

Wasserfallmodell

Die Bezeichnung *Wasserfall* begründet sich in dessen Eigenschaft, dass die oben beschriebenen Phasen der Softwareentwicklung nacheinander durchlaufen werden. Die einzelnen Phasen *Anforderungsanalyse*, *Grobdesign*, *Feindesign*, *Implementierung* sowie *Test* und *Integration* haben die jeweiligen Ergebnisse der Vorphase als Ausgangspunkt [Royce 1970]. Davon ausgehend ist eine grobe Projektplanung anhand der beschriebenen Phasen einfach machbar, da jede der Phasen mit einem Meilenstein abschließt. Bei jedem Meilenstein müssen die Ergebnisse der abgeschlossenen Phase kritisch geprüft werden. Davon ausgehend ist die Entscheidung zu treffen, ob und in welcher Form das Projekt weiterläuft. Da im Rahmen der Entwicklung von hochkomplexen Software-Systemen nicht sichergestellt werden kann, dass alle Phasen auf Anhieb erfolgreich abgeschlossen werden, wurde das Wasserfallmodell um die Möglichkeit ergänzt, dass man bei offenen Problemen in eine vorherige Phase zurückspringen kann, um mögliche Probleme innerhalb dieser vorherigen Phase zu lösen.
Trotz dieser Optimierung verlangt das Wasserfallmodell, dass sämtliche funktionale Anforderungen des künftigen Benutzers nach dem ersten Durchlauf der Phase vollständig vorliegen. Dies widerspricht sich allerdings mit Erfahrungen aus der Praxis bei der Entwicklung komplexer Software-Systeme. Hierbei kommt es immer wieder vor, dass sich die Anforderungen des Kunden ändern.

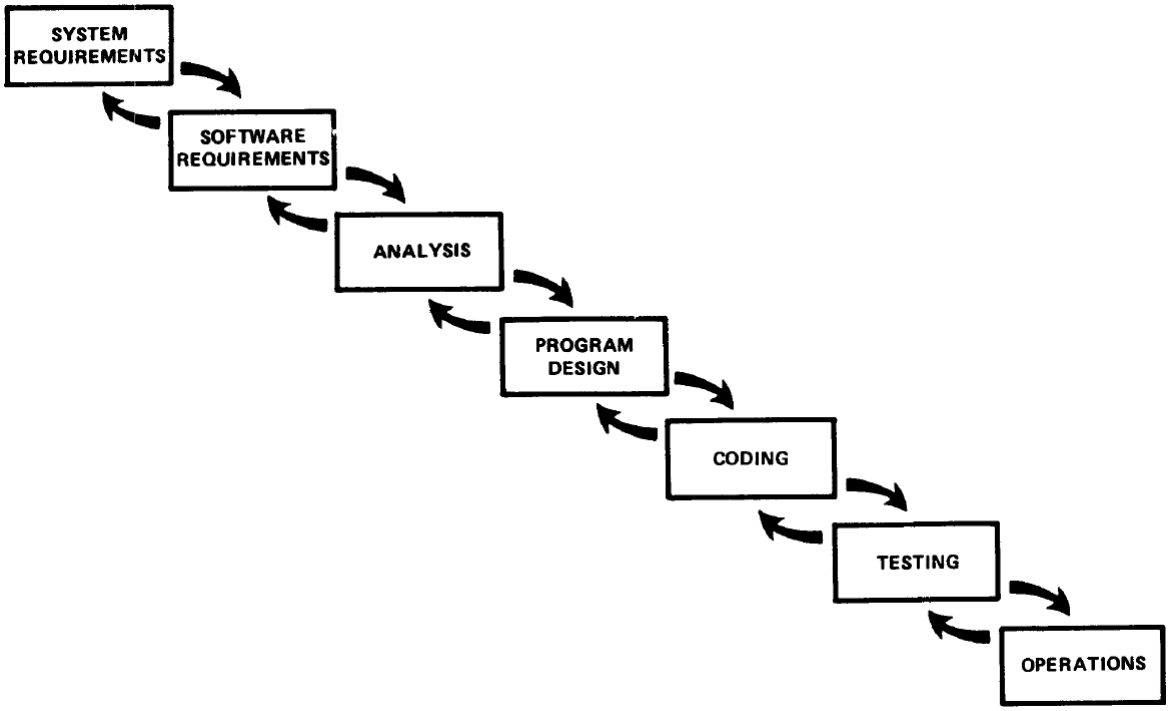


Abbildung 3.5.: Wasserfallmodell [Royce 1970, S. 330]

Der Projekterfolg verlagert sich dann auf eine Umsetzung dieser geänderten Anforderungen zur Zufriedenheit des Kunden.

> „Aber alle größeren Software-Projekte haben gezeigt, dass diese Annahme falsch ist. Es ist praktisch unmöglich, eine vollständige Menge von Anforderungen zu einem frühen Projektzeitpunkt zu formulieren [Kleuker 2011, S. 26]."

Ein weiteres Problem bei der Verwendung des Wasserfallmodells als Grundlage zur Planung von SWE-Projekten ist das Phänomen, dass Projekte bis kurz vor Projektabschluss wie ein Erfolg aussehen und innerhalb der letzten beiden Phasen (Test und Integration) zu erheblichen Zeitverzögerungen kommen. Die Ursache liegt hierbei nach Kleuker in der Tatsache, dass der Übergang von einer unvollständigen Anforderungsanalyse (z.B. aufgrund von Termindruck) in die Grobdesign Phase augenscheinlich wenige Probleme bereitet. Ähnlich verhält es sich beim Übergang vom Grobdesign in das Feindesign, da jene Phasen fast ausschließlich auf Papier mit Texten und Modellen dokumentiert werden. Erst im Rahmen der Implementierung werden Probleme durch fehlende oder unzureichende Spezifikationen deutlich. Durch den dadurch implizierten Nachholbedarf von versäumten Tätigkeiten aus den vorherigen Phasen gerät das Projekt zeitlich und ressourcenmäßig aus den Fugen [Kleuker 2011].

Prototypische Entwicklung

Die prototypische Softwareentwicklung stellt eine Verbesserung des Wasserfallmodells dar: Vor dem eigentlichen Projekt wird hier ein Prototyp der Software (bzw. von Teilen der Software) mit der Zielsetzung entwickelt, möglichst viele potentielle Probleme im Vorfeld des eigentlichen Projekts zu finden. Hierbei ist beispielsweise ein Prototyp einer Benutzer-Oberfläche des zu entwickelnden Softwaresystems besonders typisch. In diesem Zusammenhang wird die spätere Funktionalität der beauftragten Software sichtbar, wobei die dahinter liegende Funktionalität der einzelnen GUI-Elemente noch nicht „ausprogrammiert" ist. Der Ansatz eignet sich folglich dazu, mit dem Kunden zu entscheiden, ob die Bestrebungen der Softwareentwicklung in die richtige Richtung gehen und mit der Zielsetzung des Software-Systems zu vereinbaren sind. Ein weiteres Einsatzgebiet von Prototypen ist der sog. *technische Durchstich*. Hierbei besteht die Zielsetzung im Sinne einer technischen Machbarkeitsstudie die einzelnen Technologien auf Realisierbarkeit zu untersuchen und diese sicherzustellen [Bischofberger und Pomberger 1992].
Dem entscheidenden Problem im Ansatz des *Wasserfallmodells* kann aber auch durch den Einsatz von Prototypen nicht beigekommen werden: Die Entwicklung von Prototypen im

Vorfeld des SWE-Projekts sieht auch keine Änderungen der Kundenanforderungen im späteren Verlauf des Projekts vor.

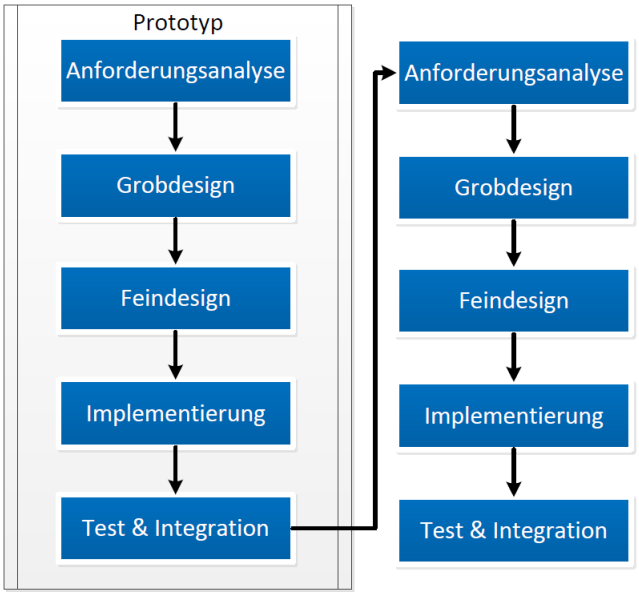


Abbildung 3.6.: Prototypisches Vorgehensmodell

Iterativ/Inkrementelle Vorgehensmodelle

Die Begriffe *iterativ* und *inkrementell* haben im Kontext des unten dargestellten Vorgehensmodells folgende Bedeutung:

> „Wenn man den Begriff Iteratives Vorgehen formal anwendet, bedeutet er, dass ein vorgegebenes Problem durch wiederholte Bearbeitung gelöst wird. Der Zusatz *inkrementell* bedeutet, dass bei jedem Durchlauf nicht nur das existierende Ergebnis verfeinert wird, sondern [...] neue Funktionalität hinzukommt [Kleuker 2011, S. 30]."

Iterative Vorgehensmodelle sehen die von der prototypischen Entwicklung bekannte Wiederholung der Phasen mehrfach vor. Die jeweiligen Phasen werden in einer Schleife durchlaufen und enden mit Fertigstellung des Produkts. Hierbei besteht die Zielsetzung, mit jedem Schleifendurchlauf die Ergebnisse des vorherigen Durchlaufs zu verfeinern und zu optimieren.

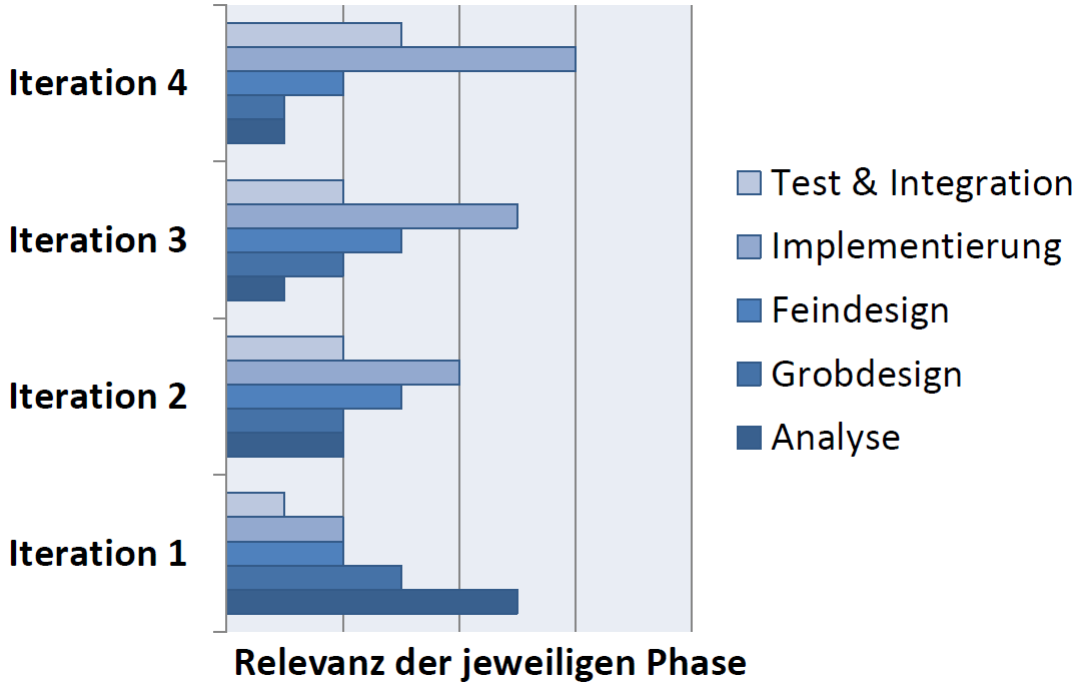


Abbildung 3.7.: Iterativ-/Inkrementelles Vorgehensmodell

Die Abbildung 3.7 verdeutlicht beispielhaft, dass sich der Schwerpunkt der einzelnen Phasen innerhalb der verschiedenen Iterationen verschiebt. Wo zu Projektbeginn möglichst viele Anforderungen aufgenommen werden ist zu einem späteren Zeitpunkt im Verlauf des Projekts die Implementierung von deutlich höherer Relevanz. Ein kennzeichnendes Merkmal dieses Vorgehensmodells ist das mehrmalige Durchlaufen aller Phasen. Demgemäß umfasst auch die erste Iteration eine Implementierungs- und Testphase. Hierdurch ermöglicht der iterative Ansatz Probleme frühzeitig zu erkennen und diese bei der Planung der folgenden Iteration mit zu berücksichtigen. Wesentlicher Unterschied zum *Wasserfallmodell* und der *prototypischen Entwicklung* ist die Möglichkeit, auf Änderungen von Anforderungen (Requirements) zu reagieren.
Die Tatsache, dass stets zeitliche Puffer für die Reaktion auf Probleme und Risiken für die jeweilige Iteration mit eingeplant werden müssen, macht in Form von schlechter Planbarkeit einen wesentlichen Nachteil dieses Ansatzes aus.
Der Zusatz *inkrementell* stellt – wie im obigen Zitat erläutert – eine Erweiterung der Software in jedem Durchlauf der Phasen dar. Dies kann bedeuten, dass ein erstes Inkrement die technischen Herausforderungen der Software umfasst und in einem weiteren Inkrement insbesondere die Hauptanforderungen des Kunden umgesetzt werden. Die Planung

von Folgeinkrementen könnten dann ggf. an neu entdeckten Risiken oder dringlichen Kundenwünschen orientiert werden.
Der inkrementelle Bestandteil des *iterativ-inkrementellen* Ansatzes erfordert für jedes Inkrement eine neue Planung. Diese ganzheitliche Neuplanung von der *Anforderungsanalyse* bis hin zur *Test- und Integrationsphase* ermöglicht allerdings im Gegenzug den flexiblen Umgang mit sich ändernden Anforderungen.

V-Modell

Das *V-Modell* legt neben den Phasen des Systementwurfs einen besonderen Wert auf die Schritte der Qualitätssicherung [Droeschel 1998].

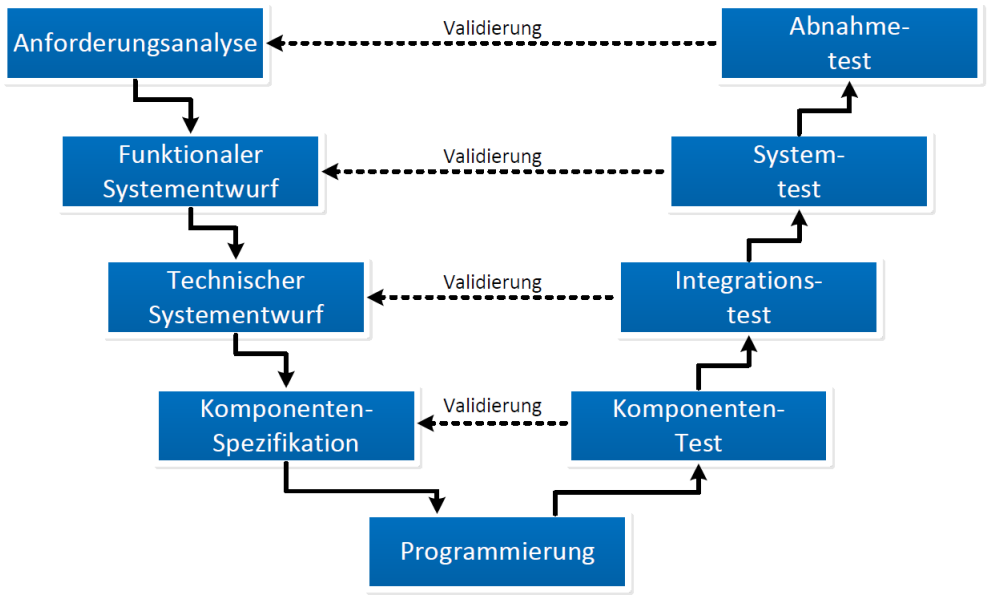


Abbildung 3.8.: V-Modell

Im linken Bereich der Abbildung 3.8 werden die eigentlichen SW-Entwicklungsschritte (die Konstruktion) beschrieben. Diese umfassen die *Anforderungsanalyse*, den *funktionalen Systementwurf* (enthält die Schritte des Grobdesigns), den *technischen Systementwurf* (auf welchem System soll die Software laufen) und die *Komponenten-Spezifikation* (enthält die Schritte des Feindesigns). Die gestrichelten Linien zeigen, dass nach jeder Entwicklungsphase geprüft wird, ob die Ergebnisse der vorherigen Phase akzeptabel sind oder ggf. verbessert werden müssen.
Im Anschluss an die Phasen der Konstruktion erfolgt die Programmierung des Software-Systems.

Der rechte Bereich des *V-Modells* (innerhalb der Integration) unterscheidet verschiedene Arten des Softwaretests und der Qualitätssicherung. Die einzelnen Testphasen sind mit den zugehörigen Phasen der Konstruktionsphase verknüpft und überprüfen diese [Droeschel 1998].

- Im *Komponenten-Test* wird geprüft, ob die entsprechenden Aspekte, die in der Komponenten-Spezifikation definiert wurden, adäquat umgesetzt worden sind. Nach dem Test der einzelnen Softwarekomponenten werden diese zum Gesamtsystem zusammengesetzt und integriert.

- Bei dem *Integrationstest* zeigt sich, ob die Phase des technischen Systementwurfs erfolgreich verlaufen ist. Der Systemtest verfolgt die Zielsetzung, die Lauffähigkeit des gesamten Systems nachzuweisen und in erster Instanz (hier noch ohne Beteiligung des Kunden/Auftraggebers) zu prüfen, ob die Anforderungen des Kunden umgesetzt worden sind.

- Im *Abnahmetest* erfolgt ähnlich wie im Systemtest eine Abnahme der Kundenanforderungen. Im Gegensatz zum Systemtest erfolgt dieser Test mit dem Kunden.

Das *V-Modell* sollte insbesondere für staatliche IT-Projekte der Bundesrepublik Deutschland zum Einsatz kommen. Hierbei bestand die Zielsetzung, Gefahren für IT-Projekte aus Sicht des Auftraggebers zu minimieren. Dementsprechend enthält das *V-Modell* Vorgaben zu den jeweiligen Prozessen, die innerhalb der Projekte geplant werden müssen und welche Artefakte zu welchem Zeitpunkt vorliegen müssen [Kleuker 2011, S. 32]. Das *V-Modell* der Bundesrepublik Deutschland wurde in mehreren Iterationen entworfen.

- *V-Modell 92*
  Das *V-Modell 92* orientiert sich vorwiegend am Wasserfallmodell und war im Hinblick auf aufkommende iterativ-inkrementelle Vorgehensweisen innerhalb des Software-Engineerings nicht mehr zeitgemäß.

- *V-Modell 97*
  Charakterisierend für das *V-Modell 97* ist insbesondere eine erste Öffnung gegenüber weiteren Vorgehensmodellen des Software-Engineerings.

- *V-Modell XT*
  Das *V-Modell XT* (Extreme Tailoring) wurde im Gegensatz zum *V-Modell 97* wesentlich weiterentwickelt und für weitere Anwendungsszenarien geöffnet. Es wird in kürzeren Zyklen aktualisiert und weiterentwickelt.

Rational Unified Process

Der *Rational Unified Process (RUP)* wurde von *Booch*, *Rumbough* und *Jacobsen* entwickelt. Jene waren auch federführend an der Entwicklung der *Unified Modeling Lanugage (UML)* beteiligt. Ausgehend von der Erkenntnis, dass unter Zuhilfenahme der *UML* unterschiedliche erfolgreiche Vorgehensstrategien zur Entwicklung von Software-Systemen möglich sind, und diese Vorgehensmodelle Gemeinsamkeiten aufwiesen, wurde der *RUP* auf Grundlage von *best practices* des SW-Engineering entwickelt. Der *RUP* sieht die Benutzung von SW-Werkzeugen der Firma *IBM-Rational* vor, kann aber unabhängig davon als Prozessmodell im SW-Engineering verstanden werden [Rational Software Corporation IBM. 1998].

Zielsetzung und "Best Practices"

Der *RUP* versteht sich als Ansatz um Aufgaben und Verantwortlichkeiten im Rahmen von Softwareentwicklungsprozessen innerhalb einer Organisation zu verteilen. Hierbei besteht die Zielsetzung, einerseits ein Prozess-Framework zur Entwicklung qualitativ hochwertiger Software zur Verfügung zu stellen, die den Anforderungen der Benutzer gerecht werden und andererseits den Softwareentwicklungsprozess in einem vertretbaren zeit- und kosteneffizienten Rahmen zu absolvieren. Die stetige Weiterentwicklung des *RUP* wird durch *IBM - Rational* Software in Zusammenarbeit mit Kunden und Partnern vorangetrieben. Hierbei liegt der Fokus auf der ständigen Weiterentwicklung des Prozess-Frameworks, sodass jenes stets die aktuellen Erkenntnisse und in der Praxis bewährte Vorgehensweisen widerspiegelt und umfasst [Rational Software Corporation IBM. 1998]. *RUP* hat die Zielsetzung, die Produktivität in Teams, in dem jedes Teammitglied der Zugang zu einer gemeinsamen Wissensbasis (Zielbeschreibungen, Vorlagen, etc.) gewährt wird, zu erhöhen. Unabhängig von den jeweiligen Aufgaben der Teammitglieder soll sichergestellt werden, dass alle Teammitglieder

- eine gemeinsame Sprache und Terminologie verwenden.
- von einem gemeinsamen Prozess ausgehen.
- die gleiche Sicht auf die Softwareentwicklung haben.

Der *RUP* unterstützt die Erzeugung und Pflege von Modellen. In diesem Zusammenhang soll keine sinnlose Anhäufung von Modellen unterstützt werden, sondern eine semantisch möglichst umfangreiche Repräsentation einer Software im Entwicklungszustand. In diesem Zusammenhang gibt *RUP* Anleitung und Hinweise für die effektive Verwendung der

UML. Als technisches Hilfsmittel umfasst das *RUP-Prozessframework* Werkzeuge zur Unterstützung des SWE-Prozess. Hierbei kommen Werkzeuge zur visuellen Modellierung, Programmierwerkzeuge und Werkzeuge zum Softwaretest zum Einsatz.
Eine weitere wichtige Zielsetzung, die aus Sicht des Autors auch die Verwendung des *RUP* als theoretische Grundlage zur Entwicklung von Komponenten und Dimensionen eines Kompetenzstrukturmodells legitimiert, ist die dynamische Konfigurierbarkeit, die dieses Vorgehensmodell charakterisiert. Laut *RUP-Zielsetzung* ist selbiger sowohl für kleine als auch für große Entwicklungsteams und Organisationen geeignet und entsprechend anpassbar. *RUP* zeichnet sich darüber hinaus durch eine einfache und klar strukturierte Prozess-Architektur aus. Diese kann auch mittels eines Prozess-Entwicklungs-Werkzeugs auf die Anforderungen einer Organisation angepasst werden [Rational Software Corporation IBM. 1998, S. 1].
Das wohl markanteste Charakteristikum des *RUP* ist der effektive Einsatz von etablierten Vorgehensweisen bzw. -mustern aus der Softwareentwicklungspraxis. Hierbei handelt es sich um die folgenden sechs *best practices* [Rational Software Corporation IBM. 1998, S. 2]

1. *Develop Software Iteratively*
   Wie zuvor beschrieben ist es bei komplexer Software fast aussichtslos folgenden sequentiellen Ablauf bei der Softwareentwicklung zu verfolgen:
   - Das Problem definieren
   - Das Design der Lösung entwerfen
   - Die Software zu implementieren
   - Die Software zu testen

   Dementsprechend verfolgt der *RUP* einen iterativen Ansatz, der es ermöglicht,
   - ein Problem sukzessive zu verstehen und
   - durch sukzessive Verfeinerung der zu entwickelnden Software zu einer effektiven Lösung zu kommen.

   Der *RUP* offeriert einen interaktiven Lösungsansatz, der die Benutzer mit in den SW-Entwicklungsprozess einbezieht, um mögliche Risiken im Projekt frühzeitig zu erkennen und zu minimieren. Dies wird dadurch gewährleistet, dass ständig ausführbare Prototypen der Software (auch dem Kunden) zur Verfügung stehen und somit eine kundenseitige Weiterentwicklung durch Tests und Feedback stets möglich ist. Dieser interaktive Ansatz erleichtert zudem den Umgang mit Änderungen in Anforderungen, gewünschten Produkteigenschaften und Zeitplänen.

2. *Manage Requirements*
   Der *RUP* umfasst Techniken um benötigte Funktionalitäten und Grenzen in der Umsetzbarkeit eines Software-Systems aufzufinden, zu organisieren und zu dokumentieren. Er beinhaltet zudem das *Change Management* also den Umgang mit *Change Requests* und deren Dokumentation. Zur Ableitung von funktionalen Anforderungen setzt der *RUP* auf die Entwicklung Use Cases und Geschäftsprozessen in Kommunikation mit dem Kunden. Diese können das Software-Design, die Implementierung und das Testen von Software so beeinflussen, dass das Endprodukt den Anforderungen des Kunden genügt. Zudem können diese als Leitfaden für das fertige Produkt und deren Entwicklung dienen.

3. *Use Component-based Architectures*
   Das im *RUP* empfohlene Vorgehensmodell akzentuiert die frühe Entwicklung eines robusten, lauffähigen Basis-Systems. Hierzu gibt das Framework Hilfestellung um die folgenden Zielsetzungen an eine Software zu erreichen.

   - flexibel veränderbar
   - intuitiv verständlich
   - hoher Wiederverwendungswert

   Die vorgeschlagene komponentenbasierte Systemarchitektur sieht Software-Komponenten als nicht triviale Module oder Subsysteme, die einen bestimmten Zweck erfüllen. Hierzu offeriert der *RUP* einen systematischen Ansatz, um eine Architektur zu definieren, die sich sowohl aus neuen als auch aus bereits bestehenden Komponenten zusammensetzt (adhoc/Komponenteninfrastruktur: CORBA, COM, etc.).

4. *Visually Model Software*
   Der *RUP* zeigt auf, wie Software visuell modelliert werden kann, um deren Struktur und Verhalten zu illustrieren. Dies umfasst das Ausblenden von Details und die Verwendung von Syntaxbausteinen, die zu Quellcode konsistent sind.

5. *Verify Software Quality*
   Schlechte Performance und die Zuverlässigkeit sind Faktoren, die die Akzeptanz von Software entscheidend hemmen. Folglich sollte die Qualität von Software unter Maßgabe der folgenden Kriterien überprüft werden:

   - Umsetzung der Anforderungen,
   - Zuverlässigkeit,
   - Funktionalität sowie

- Anwendungs- und Systemperformance.

Der *RUP* unterstützt die Entwickler dahingehend bei der Planung, Implementierung sowie der Ausführung und Evaluation der spezifischen Tests. Darüber hinaus sieht der *RUP* die Integration des Qualitätsmanagements in jeder Entwicklungsphase vor sowie die Einbindung sämtlicher beteiligter Entwickler.

6. *Control Changes to Software*
   Hierbei geht es um die Fähigkeit des Frameworks, Changes bzgl. Software zu verwalten und sicherzustellen, dass diese für Antragsteller des Change Requests akzeptabel sind. Insbesondere in einer System-Umgebung, in der Veränderungen unabwendbar sind, ist es von essentieller Wichtigkeit, Änderungen (=Change Requests) kontrolliert in die Wege leiten zu können. Insbesondere um eine erfolgreiche iterative Entwicklung zu ermöglichen, umfasst der Prozess der Aufnahme von Change Requests, deren Kontrolle und Überwachung.

   Weiterhin bietet das *RUP-Framework* die Möglichkeit, Entwickler von Änderungen anderer Entwickler zu isolieren, die Veränderungen von *Software-Artefakten* zu kontrollieren (z.B. durch Modelle, Dokumente, etc.) und verhilft Teams als Einheit zu arbeiten, indem eine automatisierte Integration und Kompilierung von Software-Komponenten erfolgen kann.

Dimensionen des *RUP*

Der *Rational Unified Process* stellt sich als zweidimensionales Prozessmodell dar.

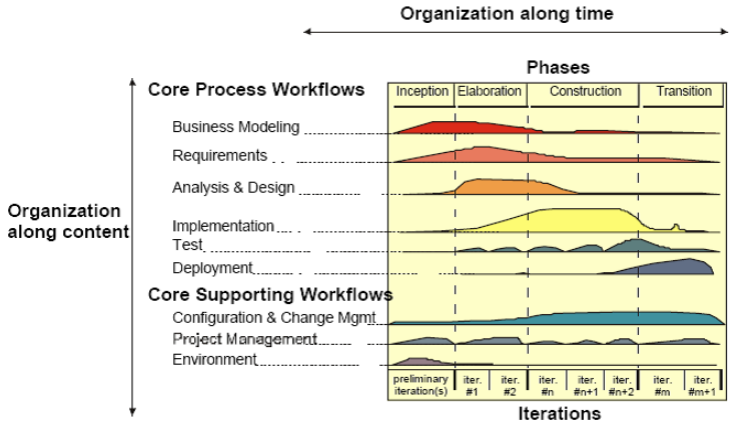


Abbildung 3.9.: RUP-Dimensionen [Rational Software Corporation IBM. 1998, S. 3]

Horizontale Dimension

Die sog. Zeitdimension repräsentiert den Software-Lebenszyklus in Form von sog. *cycles*. Jeder Zyklus steht für die Entwicklung einer neuen Ausbaustufe des Produkts. Jeder Durchlauf eines Zyklus umfasst die Phasen *Inception Phase*, *Elaboration Phase*, *Construction Phase* und *Transition Phase*. Jede dieser Phasen dient einem bestimmten Zweck und schließt mit einem explizit definierten Meilenstein. An diesem Punkt müssen wesentliche und kritische Entscheidungen getroffen werden [Rational Software Corporation IBM. 1998, S. 3ff].

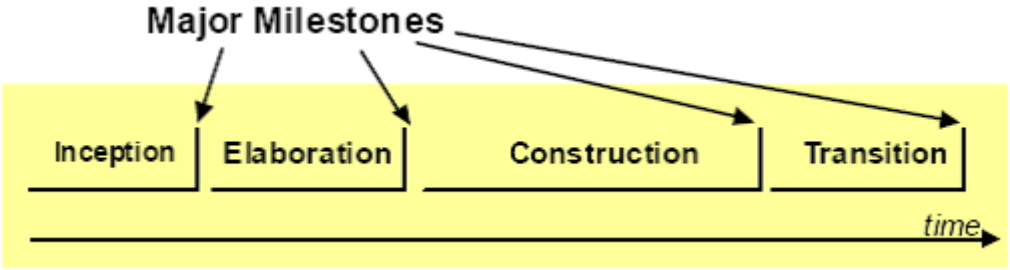


Abbildung 3.10.: RUP-Meilensteine & Software-Lebenszyklen [Rational Software Corporation IBM. 1998, S. 3]

Inception Phase

Während dieser Phase wird ein sog. Geschäftsfall (*business case*) für das zu entwickelnde System erstellt und der Projektscope abgesteckt. Hierzu müssen alle externen Akteure, mit denen das System interagiert, herausgefunden werden. Dies umfasst insbesondere das Ableiten sämtlicher Use Cases sowie die Beschreibung der wichtigsten Use Cases. Der Geschäftsfall beinhaltet zudem Erfolgskriterien, Risikoabschätzungen, Abschätzungen zu den benötigten Ressourcen sowie ein Vorgehensmodell in Form eines Phasenplans mit Meilensteinen.
Ergebnisse und Ziele der Phase:

- Visions-Dokument mit Kernanforderungen und Grenzen des Projektscopes
- initiales Use Case Modell (10-20% vollständig)
- initialer Projekt-Glossar
- ein initialer Geschäftsfall (beinhaltet Geschäftszusammenhang, Erfolgskriterien und finanzielle Prognosen)

- Risikoeinschätzung
- Projektplan, der Phasen und Iterationen beinhaltet
- Geschäftsmodell, sofern erforderlich
- ein oder mehrere Prototypen

Erfolgskriterien zum Erreichen des Meilensteins **Lifecycle Objectives**:

- Kenntnis/Vertrautheit relevanter Personen mit Projektgegenstand sowie Kosten- und Zeitabschätzung
- Verständnis der Anforderungen, wie in den Use Cases vermittelt
- Plausibilität der Kosten-/Zeitabschätzungen, Prioritäten und Risiken des Entwicklungsprozesses
- Bisherige getätigte und geplante Ausgaben

An dieser Stelle kann das Projekt abgebrochen werden oder eine Neukonzeption angestoßen werden.

Elaboration Phase

Innerhalb dieser Phase wird die Domäne des Problembereichs analysiert, eine geeignete Systemarchitektur festgelegt, ein Projektplan erstellt und Aspekte von hohem Risiko aus dem Projektumfang entfernt. Um diese Ziele umzusetzen, muss man einen umfassenden und detaillierten Blick auf das zu erstellende Software-System haben. Entscheidungen hinsichtlich einer geeigneten Systemarchitektur müssen ausgehend vom Gegenstandsbereich und dessen Abgrenzung, den Hauptfunktionalitäten und nicht-funktionalen Anforderungen (z.B. Performanz) an das zu entwickelnde System getroffen werden. Die Phase wird laut *RUP-Whitepaper* als kritischste der vier Phasen eingeschätzt mit deren Abschluss ein Großteil der Entwicklungsarbeiten abgeschlossen ist. Charakterisierend für diese Phase ist die Entscheidung, ob man die Folgephasen *construction & transition* wie geplant durchführt oder nicht. Hier erfolgt der Übergang von einer leichten, wenig kostenden und kaum riskanten Phase zu einer teuren, risikoreichen Phase. Die Aktivitäten der *inception phase* stellen trotz etwaiger Veränderungen sicher, dass die Architektur-Planungen, Anforderungen und Pläne zur Vorgehensweise so stabil sind, dass Risiken gemindert werden und verlässliche Kosten- und Zeitprognosen gemacht werden können.

Ein derartiger Grad an Verlässlichkeit ist erforderlich, um einer preislich fixierten Konstruktionsphase zuzustimmen und diese durchzuführen.
In der Ausarbeitungsphase entsteht innerhalb von einer oder mehrerer Iterationen ein ausführbarer Architektur-Prototyp. Dieser umfasst die wesentlichen Use Cases der *inception phase*. Weitere *Wegwerf-Prototypen* können dabei behilflich sein, Risiken hinsichtlich Anforderungen und Design zu minimieren. Sie können als technische Machbarkeitsstudie für bestimmte Elemente des SW-Systems fungieren und zur Demonstration bei Investoren, Endkunden und Benutzern eingesetzt werden.
Ergebnisse und Ziele der Phase:

- Use Case Modell (mehr als 80% fertiggestellt), das sämtliche relevante Fälle und Akteure identifiziert und fast alle modellierten Use Cases enthält
- Nicht funktionale Anforderungen
- Software-Architektur-Beschreibung
- gesicherte Risikoliste
- Entwicklungsplan für das gesamte Projekt inkl. eines groben Projektplans, der die Iterationen und die jeweilig dazugehörigen Evaluationskriterien umfasst

Erfolgskriterien zum Erreichen des Meilensteins **Lifecycle Architecture**:
Zu diesem Zeitpunkt werden die detaillierten Systemzielsetzungen, Sichtweisen, Architekturentscheidungen und Hauptrisiken untersucht. Hierbei ergeben sie die folgenden Evaluationskriterien:

- Ist die Vision für das Projekt gefestigt?
- Besteht eine stabile Architektur?
- Zeigen demonstrative Ausführungen von Prototypen, dass problematische/gefährdende Aspekte erkannt und glaubwürdig gelöst wurden?
- Sind die Planungen für die Konstruktionsphase akkurat und ausreichend detailliert?
- Sind die für das System relevanten Personen mit der Vision, dem Plan zur Entwicklung des Systems in Bezug auf die beschlossene Architektur einverstanden?
- Ist das Verhältnis der gemachten Ausgaben in Bezug auf die geplanten Ausgaben akzeptabel?

Falls dieser Meilenstein nicht erreicht wird, kann das Projekt abgebrochen oder neu konzipiert werden.

Construction Phase

Innerhalb der Konstruktionsphase werden alle noch ausstehenden Komponenten und Anwendungen entwickelt und in das Produkt integriert. Es werden darüber hinaus alle Features gründlich getestet. Innerhalb der Phase vollzieht sich ein Wandel des Entwicklungsgegenstands von intellektuellem Eigentum während der *Inception- bzw. Elaborationphase* zu einsetzbarem praxistauglichen Produkten während der Construction- bzw. Transfer-Phase. Viele Projekte sind von solch großem Umfang, dass eine parallele Weiterentwicklung des Systems möglich ist. Eine derartige Entwicklung kann die Verfügbarkeit von praxistauglichen Releases deutlich erhöhen. Im Umkehrschluss erhöht sich allerdings auch die Komplexität im Hinblick auf das Ressourcenmanagement und die Synchronisation der Entwicklungsprozesse.
Ergebnisse und Ziele der Phase:

- ein fertiges Software-Produkt, das für die Benutzung von Endbenutzern freigegeben ist
- Software-Produkt integriert in eine adäquate Plattform
- Bedienungsanleitungen / Release Notes

Erfolgskriterien zum Erreichen des Meilensteins **Initial Operational Capability**:
Zu diesem Zeitpunkt wird entschieden, ob es möglich ist, die Software zur Benutzung an die Endbenutzer freizugeben, ohne das Projekt einem hohen Risiko auszusetzen. Diese Ausbaustufe wird häufig als beta-release bezeichnet. Hierbei ergeben sich die folgenden Evaluationskriterien.

- Ist das Produkt ausgereift genug, um in den Benutzer-Szenarien verwendet zu werden?
- Sind die für das System relevanten Personen in der Lage, Nutzer der Software zu sein?
- Ist das Verhältnis der gemachten Ausgaben in Bezug auf die geplanten Ausgaben immer noch akzeptabel?

Wenn das Projekt diesen Meilenstein nicht erreicht, muss die Überführung (*Transition*) um ein Release verschoben werden.

Transition Phase

Zweck der *Transition-Phase* ist die Überführung der Software in die Benutzer Community. Sobald die Verwendung der Software begonnen hat, werden die Entwickler dazu veranlasst, neue Releases herauszubringen, Bugfixes zu implementieren und Funktionen zu vervollständigen. Die Phase sollte beginnen, sobald eine Basis der Software derart ausgereift ist, um in der Benutzerdomäne eingesetzt werden zu können. Hierzu ist es erforderlich, dass ein Bereich der Software finalisiert wird und eine akzeptable Qualität aufweist; des Weiteren sollte dieser Bereich fertig dokumentiert sein, damit die Transition-Phase für alle Seiten erfolgreich verläuft. Bedingungen hierfür sind:

- Beta Testing um das System in Bezug auf Benutzererwartungen zu prüfen
- Paralleler Einsatz eines bestehenden Systems, welches durch das Neue ersetzt werden soll

Ergebnisse und Ziele der Phase:

- Erreichen, dass sich die Nutzer selbst unterstützen und Support geben können
- Enduser mit dem Produkt konfrontieren, um sicherzustellen, dass die Grundanforderungen zur Veröffentlichung des Produkts vollständig sind und den Anforderungen der Vision entsprechen
- Bedienungsanleitungen / Release Notes
- Möglichst Ressourceneffizientes Finalisieren des Produkts

Die Phase kann – in Abhängigkeit von dem jeweiligen Produkt – einfach aber auch sehr komplex sein.
Erfolgskriterien zum Erreichen des Meilensteins **Product Release**:
An diesem Punkt wird entschieden, ob die Anforderungen/Ziele erreicht wurden und ob weitere Entwicklungs-Zyklen durchgeführt werden müssen. Es ergeben sich die folgenden Evaluationskriterien:

- Sind die Endnutzer zufrieden?
- Ist das Verhältnis der gemachten Ausgaben in Bezug auf die geplanten Ausgaben immer noch akzeptabel?

Vertikale Dimension

Die vertikale Dimension des *RUP* kann als statische Struktur des Prozesses verstanden werden. Ein Prozess beschreibt *wer*, *was* und *wann* tut. Hierzu verwendet der *RUP* vier sog. *Primär-Modellierungselemente* [Rational Software Corporation IBM. 1998, S. 7].

1. *workers ("wer")*
   Worker sind durch das Verhalten und die Verantwortlichkeiten eines Individuums definiert. Sie können über eine Rolle charakterisiert werden, die von jedem Individuum bekleidet werden kann. Individuen können auch mehrere Rollen bekleiden. Die Verantwortlichkeit eines *workers* können sowohl die Ausführung einer bestimmten Menge von Aktivitäten als auch das Besitzen einer Menge von *Artefakten* sein.

2. *activities ("wie")*
   Eine Aktivität eines spezifischen *workers* ist eine Arbeitseinheit, die ein Individuum, welches diese Rolle bekleidet, aufgefordert werden kann, auszuführen. Diese hat einen klar definierten Zweck, wie z.B. die Aktualisierung und Erstellung von Modellen (*Artefakten*). Jede *activity* ist einem bestimmten *worker* zugeordnet und deren zeitliche Einteilung beträgt einige Stunden bis hin zu einigen Tagen. Eine *activity* ist zur Bearbeitung von einem oder einigen wenigen *Artefakten* [Rational Software Corporation IBM. 1998, S. 8ff].

   Beispiele für *activities*:

   - *activity:* Eine Iteration planen; *worker:* Project Manager
   - *activity:* Use Cases und Akteure herausfinden; *worker:* System Analyst
   - *activity:* Review des Design; *worker:* Design Reviewer
   - *activity:* Ausführen eines Performance-Test; *worker:* Performance Tester

3. *artifacts ("was")*
   Artifacts sind ein Teil einer Information, welche von einem Prozess erstellt, verändert oder benutzt wird. Sie sind als materielle Ergebnisse des Projekts zu verstehen und können Input für *worker* zur Bearbeitung einer Aktivität sein oder können Ergebnisse jener sein.

   Beispiele für *artifacts*:

   - Ein Modell (Use Case-/Design-Modell)
   - Ein Element eines Modells (eine Klasse, ein Subsystem)
   - Ein Dokument (Geschäftsfall, Softwarearchitekturdokument)

- Quellcode
- Ausführbare Dateien

4. *Workflows ("wann")*
   Ein Prozess ist nicht eine Konstitution einer Anzahl von *workern*, *activities* und *artifacts* sondern beschreibt vielmehr einen Ablauf von Aktivitäten und die Interaktion zwischen *workern*. Ein *workflow* ist demnach ein Ablauf von Aktivitäten, der ein definiertes Ergebnis erzielt. Workflows innerhalb des *RUP* werden durch folgende *UML-Diagramme* beschrieben: Sequenzdiagramm, Kollaborations-Diagramm und Aktivitätendiagramm.

Die beschriebene vertikale Dimension des *RUP* umfasst neun zentrale Phasen, sog. *Core Workflows*. Jene repräsentieren eine Aufteilung von *workers* und *activities* in logische Gruppen. Hierbei lässt sich bei den neun Phasen eine Unterscheidung in sechs sog. *engineering workflows* und sog. *supporting workflows* machen.
Im Folgenden sollen die einzelnen *Core Workflows* beschrieben werden. Hierbei liegt der Fokus auf den engineering workflows, da diese auch als Grundlage für die theoretische Ableitung von Kompetenzdimensionen und -kategorien fungieren sollen [Rational Software Corporation IBM. 1998, S. 10ff].

Business Modeling (Geschäftsmodellierung)

Eins der größten Probleme im SWE-Prozess ist, dass die Softwareentwicklung und die Entwicklung im Geschäftsfeld häufig unter Kommunikationsproblemen leiden. Dies führt dazu, dass der output vom *business engineering* häufig nicht für das SW-Engineering verwendet wird und umgekehrt. Der *RUP* versucht die Problematik mit einer gemeinsamen Sprache und einem gemeinsamen Prozess entgegenzuwirken, indem er zeigt, wie Zusammenhänge zwischen business- und software-Modellen hergestellt und gepflegt werden können. Im Rahmen der Geschäftsmodellierung werden sog. Business Use Cases modelliert. Diese sichern ein allseitiges Verständnis hinsichtlich der Faktoren, von denen der Geschäftsprozess abhängt und inwiefern dieser unterstützt werden muss. Hierbei besteht die Zielsetzung, wie das Geschäftsfeld von der zu entwickelnden Software unterstützt werden soll. Es entsteht als Ergebnis dieser Phase ein sog. *Business Object Model*.

Requirements (Anforderungsanalyse)

Ziel dieser Phase ist die Beschreibung, wie sich das System verhalten soll. Sowohl die Entwickler als auch die Kunden müssen dieser Beschreibung zustimmen. Hierbei werden

benötigte Funktionalitäten, Grenzen des Systems und Stakeholder ermittelt und in einem Visionsdokument beschrieben und organisiert. Zentraler Inhalt sind die Anforderungen aller Stakeholder sowie sämtliche Akteure in Form von Benutzern und anderer Systeme, die mit der zu entwickelnden Software interagieren. Dementsprechend werden Use Cases identifiziert, die das Systemverhalten repräsentieren. Dies schafft für die relevanten Akteure einen direkten Bezug zum System, da die Use Cases ausgehend von Anforderungen konzipiert werden. Die Use Cases beschreiben im Detail, wie das System Schritt für Schritt mit den Akteuren interagiert und welches Verhalten das System mit sich bringt. Nicht funktionale Anforderungen werden in einer separaten Dokumentation gepflegt und organisiert.

Analysis & Design (Analyse & Design)

Ziel dieser Phase ist es zu modellieren, wie das System in der Implementierungsphase realisiert werden soll. Infolgedessen müssen die Aufgaben und Funktionen ausgehend von Use Case Beschreibungen umgesetzt werden, sodass diese den Anforderungen genügen. Auch die Auswahl einer geeigneten und einfach zu modifizierende Architektur sollte hier vorgesehen werden. So ist sichergestellt, dass eine flexible Berücksichtigung von sich ändernden Anforderungen gewährleistet ist.
Die Phase resultiert in einem Design-Modell (und optional in einem Analyse-Modell).

- fungiert als Abstraktion für den Quellcode
- beschreibt und strukturiert den Quellcode
- besteht aus Design Klassen – strukturiert in design-packages und design-Subsystemen
- besteht aus Beschreibungen, wie die jeweiligen Klassen kollaborieren um Use Cases umzusetzen.

Implementation (Implementierung)

Ziel der Implementierungsphase ist die Realisierung des Systems durch Implementierung seiner konstituierenden Software-Komponenten. Hierbei muss zunächst die Organisation und Strukturierung von Quellcode erfolgen, sodass die verschiedenen Implementierungs-Subsysteme feststehen. Ferner soll die Implementierung von Klassen als Komponenten sowie der Test von Komponenten als Einheiten (unit tests) durchgeführt werden. Die Erzeugnisse unterschiedlicher individueller Programmierer und Teams gilt es zu einem ausführbaren System zusammenzusetzen.

Zur Unterstützung der Implementierungsphase beschreibt der *RUP*, wie bestehende Komponenten weiterverwendet werden können und wie neue Komponenten zur Erfüllung eines wohl-definierten Zwecks entwickelt werden können. Weiterhin wird durch das Framework beschrieben, wie man ein System so konzipiert, dass es leicht wartbar ist und einen hohen Grad an Wiederverwendbarkeit hat.

Test

Zielsetzung der Testphase im Sinne des *RUP* ist die Verifikation der Interaktion zwischen Objekten, der Integration aller Softwarekomponenten und dass alle Anforderungen korrekt implementiert wurden. Zudem muss sichergestellt sein, dass Fehler behoben wurden. Neben der Zielsetzung schlägt der *RUP* die Integration der Testphase während der gesamten Phasen der Software Entwicklung vor. Somit ist gewährleistet, dass Bugs so früh wie möglich aufgefunden werden und somit auch die Kosten bei der Fehlerbehebung drastisch gesenkt werden können.
Die eigentlichen Tests erfolgten im *RUP* in drei Qualitätsdimensionen

1. Reliability
2. Functionality
3. Application- and System-Performance

Für jede dieser Dimensionen beschreibt der *RUP*, wie die Test-Phase zu durchlaufen ist und gibt Hinweise im Hinblick auf Planung, Design, Implementierung, Ausführung und Evaluation des Tests. Er beschreibt Strategien, wie Tests automatisiert werden können. Dies spielt insbesondere für den iterativen Ansatz eine große Rolle, da somit Regressionstests nach jeder Iteration (nach jeder neuen Version der Software) durchgeführt werden können.

Deployment (Auslieferung/Verteilung)

Ziel dieser Phase ist im engeren Sinne das erfolgreiche Verteilen von Produkt-Releases an die Benutzer. Darüber hinaus umfasst diese Phase weitreichende Aufgaben:

- Externe Releases von Software
- Verpacken der Software
- Verteilung der Software

- Installation der Software
- Support für die Software
- Planung und Durchführung von beta-Tests
- Migration von existierender Software und Daten
- Formale Akzeptanz

Die zuvor beschriebenen Phasen repräsentieren die Engineering Workflows des *RUP*. Im Folgenden soll der Vollständigkeit halber ein kurzer Einblick in die sog. *supporting workflows* gegeben werden:

Project Management

Die *Project Management Phase* erfordert das Vermögen, gegensätzliche Ziele und Aufgaben zu vereinen. Dies umfasst ein Risiko Management und spiegelt u.a. das Können wieder ein Produkt zu erzeugen, das sowohl den Anforderungen von Kunden und Auftraggebern als auch jenen der Enduser gerecht wird. Dieser *workflow* konzentriert sich auf den iterativen Entwicklungsprozess. Zum Erreichen der Zielsetzung kann der *RUP* Hilfestellung geben, in dem jener ein Framework zum Managen von Softwareprojekten umfasst, praktische Richtlinien für die Planung, Personalfragen sowie die Ausführung und Überwachung von Projekten gibt und ein Framework zur Risikohandhabung bereitstellt.

Configuration & Change Management

In dieser Phase wird beschrieben, wie man mit den zahlreichen *Artefakten* (entwickelt von verschiedenen Personen) im Projekt umgeht. Eine Kontrolle dahingehend hilft Unklarheiten bzgl. Kosten zu vermeiden und stellt sicher, dass die daraus resultierenden Artefakte nicht im Konflikt stehen. Im Folgenden sind mögliche Szenarien aufgeführt, bei denen eine Kontrolle sinnvoll erscheint:

- Gleichzeitiges Update: Zwei Personen arbeiten am selben Artefakt, der Letzte, der die Änderungen comitted, zerstört die Arbeit seines Vorgängers.
- Eingeschränkte Benachrichtigungen: Wenn ein Problem in *Artefakten* behoben wird, und andere Entwickler davon nicht in Kenntnis gesetzt werden.
- Verschiedene Versionen: Meist haben große Softwareprojekte mehrere Versionen; wenn z.B. Fehler in einer Version auftauchen, muss sichergestellt werden, dass Verbesserungen in allen Versionen vorgenommen werden.

Environment (Umgebung)

Zweck dieses *Workflows* ist die Unterstützung der Softwareentwicklungs-Organisationen mit geeigneten Software-Tools. Hierbei stellt der *RUP* Prozesse und Werkzeuge zur Verfügung um das Entwickler-Team zu unterstützen.

Potential und Grenzen des *RUP*

Im Gegensatz zum *V-Modell*, das auf einer generischen Auswahl von Methoden und Werkzeugen basiert, gibt der *RUP* konkrete Hilfestellung unter expliziter Zuhilfenahme der Unified Modeling Language. Als Kritikpunkt merkt Kleuker jedoch an, dass nach Ablauf der *elaboration phase* innerhalb des *RUP* von einer statischen Anforderungsmenge ausgegangen wird und das Änderungen am System nur schwer und bei entsprechender Interpratation verschiedener Aktivitäten zu leisten ist [Kleuker 2011, S. 20].

Agile Vorgehensmodelle

Umfassende Vorgehensmodelle wie das *V-Modell XT* oder der *RUP* sind sehr komplex. Sie erfordern eine große Anzahl zu pflegender Dokumente und es kann vorkommen, dass mögliche Zusammenhänge nicht auf Anhieb verstanden werden [Kleuker 2011, S. 42ff]. Um dieser Tendenz entgegenzuwirken haben sich renommierte SWE-Experten in der *Agile Allianz* verbündet. Das folgende sog. *agile Manifest* beschreibt die Leitlinien der Organisation [Agile Alliance 2013]:

> We are uncovering better ways of developing software
> by doing it and helping others do it. Through this
> work we have come to value:
>
> Individuals and interactions over processes and tools
> Working software over comprehensive documentation Customer
> collaboration over contract negotiation Responding to change
> over following a plan
>
> That is, while there is value in the items on the right, we
> value the items on the left more.
>
> www.agileAlliance.org

1. Als erste Grunderkenntnis wird die Bedeutsamkeit des Entwicklerteams akzentuiert, dessen Zusammenhalt, Kommunikation und Motivation maßgeblich den Projekterfolg beeinflussen. Hierbei erfolgt eine Integration von Individuen (mit möglichst kreativer Freiheit) in Abhängigkeit der jeweiligen Stärken und Schwächen.

2. Die lauffähige Software wird als zentraler Mittelpunkt von höchster Wichtigkeit eingeschätzt. Die Dokumentation soll nur im minimalen Umfang und bei Bedarf (möglichst innerhalb des Quellcodes der zu entwickelnden Software) erfolgen.

3. Der Kunde muss möglichst intensiv mit in den Entwicklungsprozess einbezogen werden.

4. Software-Projekte müssen so geplant werden, dass sie auf Änderungen in den Kundenanforderungen reagieren können. Eine Suche nach neuen Lösungen und das Verwerfen bestehender Lösungen muss stets in Betrachtung gezogen werden.

Im Vorgehen lassen sich zwei Gruppierungen innerhalb der agilen Softwareentwicklung nennen [Kleuker 2011, S. 22ff], [Martin 2003].

1. Die erste Gruppe beinhaltet Metamodelle, die ausgehend vom oben beschriebenen Manifest ein Rahmenwerk für die Projektorganisation vorgeben ohne eine konkrete Methodik oder Werkzeuge vorzugeben, z.B. Scrum [Pichler 2009].

2. Diese Gruppe umfasst Ansätze, die auf Grundlage bewährter Vorgehensmodelle in der Softwaretechnik konkrete Vorgaben machen, z.B. Extreme Programming [Pichler 2009].

Zusammenfassend kann man die folgende Aussage über die Agile Softwareentwicklung treffen: Die Agile Softwareentwicklung hat bestehende Vorgehensmodelle der Softwaretechnik insofern weiterentwickelt und beeinflusst, als dass deren Flexibilität mit dem Aufkommen der agilen Tendenzen deutlich verbessert wurde. Hiermit wurde der praktischen Erfahrung Sorge getragen, dass ein SWE-Prozess in den seltensten Fällen von Anfang bis Ende linear durchgeplant werden kann [Martin 2003].

Zwischenfazit

Nach eingehender Betrachtung des Themenbereichs der Modellierung aus fachwissenschaftlicher und fachdidaktischer Perspektive, scheint die Verwendung eines Vorgehensmodells aus der Softwaretechnik auch im Sinne des Weinert'schen Kompetenzverständnis

als geeignete strukturgebende Grundlage, um theoretisch fundierte Kategorien für ein Kompetenzmodell ableiten zu können.
Um einen genaueren Einblick in die unterschiedlichen Vorgehensmodelle der Softwaretechnik zu erhalten und zu prüfen, auf welcher normativ-theoretischen Grundlage die Ableitung von Kompetenzen erfolgen sollte, war es sinnvoll, einzelne exemplarische Vorgehensmodelle innerhalb der Softwaretechnik aufzuführen und deren Charakteristik darzustellen und zu vergleichen. Hierbei hat sich gezeigt, dass das Wasserfallmodell zu statisch zu sein scheint um SWE-Prozesse zu planen. Die Begründung dafür ist, dass Prozesse zur Modellierung und Entwicklung von Software in der Praxis selten linear und von vorne herein planbar ablaufen. Vielversprechend hingegen erscheinen die iterativ-inkrementellen Ansätze, da diese eine explizite Dynamik innerhalb der Abfolge der einzelnen SWE-Phasen vorsehen. Hierbei sieht das *V-Modell* einen für den schulischen Einsatz zu starken Fokus auf die Qualitätssicherung vor. Das prototypische Vorgehen und die agilen Methoden der Softwareentwicklung geben aus Sicht des Autors zu wenig Vorgaben an die einzelnen Phasen des SWE-Prozesses, um mögliche Strukturen und Inhalte für das Kompetenzmodell abzuleiten. Dennoch liefern sie wertvolle Hinweise für die Gestaltung von SWE-Prozessen. Im Sinne der agilen Softwareentwicklung sollten diese ein hohes Maß an Dynamik aufweisen.
Insgesamt sprechen die folgenden Aspekte aus Sicht des Autors für die Verwendung des *Rational Unified Process* als theoretische Grundlage um die Facetten von Modellierungskompetenz abzuleiten:
Beim *Rational Unified Process* handelt es sich um ein sehr gut etabliertes Vorgehensmodell, welches als Quasi-Standard in der Softwaretechnik angesehen werden kann. Der *RUP* sieht eine iterativ-inkrementelle Vorgehensweise bei der Entwicklung von Software-Systemen vor und setzt auf sog. *best practices* im Software-Engineering.
Darüber hinaus zeichnet sich dieser Ansatz durch eine hohe Anpassbarkeit im Hinblick auf Komplexität und Umfang aus. Hierdurch lässt sich jener auch an die jeweilige Lerngruppe anpassen.
Der *RUP* wurde von den Schöpfern der *UML* kreiert und setzt dementsprechend auf diesen Standard als Modellierungssprache, welche auch im Informatikunterricht der Sekundarstufe II zum breiten Einsatz kommt.
Ferner zeigen unterschiedliche Studien und Praxisberichte den erfolgreichen Einsatz des *RUP* in der Lehre [Roggio 2006], [Goldin und Rudahl 2009] auf und legitimieren die Verwendung dieses Ansatzes als theoretische Basis für ein Kompetenz-Strukturmodell.
Durch die Forderung der Klieme-Expertise, dass Kompetenzmodelle durch entsprechende Testinstrumentarien überprüfbar sein sollen, gilt es neben der Entwicklung eines entspre-

chenden Kompetenzmessinstruments für die informatische Modellierung eine geeignete Unterrichtsreihe als Setting für die Kompetenzmessung zu planen. Um deren theoretische Grundlage festzulegen, sollen im Folgenden unterschiedliche Unterrichtskonzepte zur objektorientierten Modellierung vorgestellt werden. Der Fokus liegt hier (aufgrund des besonderen Interesses des Autors und positiver Erfahrungen in der Ausbildung von angehenden Informatiklehrern in der Hochschullehre) auf dem Thema *Objektorientierung und Robotik*.

3.4.2. Vorgehens- & Vermittlungsmodelle zur OO-Modellierung

Neben der theoretischen Fundierung des Kompetenzmodells muss auch eine theoretische Basis für die Konzeption einer Unterrichtsreihe innerhalb der die Evaluation des Kompetenzmessinstruments erfolgen soll, geschaffen werden. Hier soll ein etablierter Ansatz zur Vermittlung objektorientierter Modellierung als theoretische Grundlage verwendet werden.
Im Folgenden werden unterschiedliche Herangehensweisen und didaktische Vorgehens- bzw. Vermittlungsmodelle am Beispiel des Informatik Anfangsunterrichts und dem Themenbereich der objektorientierten Modellierung und Programmierung aufgezählt.

Objects first

Nach Diethelm besteht ein Dilemma bei der Verwendung der Terminologie *objects first*. Hier ergibt sich eine mehrfache Verwendung für unterschiedliche Vermittlungsmodelle [Diethelm 2007, S. 21ff].

- *Classes first*: Entwicklung von Klassen, die Objekte definieren.
- *OOP first*: Programmierung in einer objektorientierten Sprache und ggf. eine Visualisierung durch Stifte & Mäuse bzw. Turtle-Grafiken (haben nach Diethelm nicht immer die Intention, die objektorientierte Modellierung zu lehren).
- *Objects first*: das Objekt steht an erster Stelle.

Diethelms eigentlicher Ansatz startet mit Objekten, die in einem Szenario aus der Lebenswelt der Lernenden eingebettet sind. Hierbei handelt es sich um Objekte, die z.B. auf einer Tafel skizziert werden können und lediglich Informationen über die Objekte selbst und deren Zusammenhang enthalten. Ggf. können diese auch Attribute und entsprechende Werte enthalten. Informationen über Klassen sind in diesem Ansatz zunächst bewusst nicht vorgesehen und sollen explizit zu einem späteren Zeitpunkt thematisiert werden [Diethelm et al. 2005].

Einstieg über Programmiersprachen

Diese Vorgehensweise umfasst das Erlernen der Syntax einer objektorientierten Programmiersprache und den Umgang mit Entwicklungsumgebungen. Hier besteht keinerlei Unterscheidung zu Vermittlungsmodellen von prozeduralen Sprachen. In Veröffentlichungen zu der Thematik spricht man hier illustrativ von einem "Wolf im Schafspelz" [Diethelm 2007, S. 25], [Penon und Spolwig 1998].
Ein weiteres Vermittlungskonzept für die Grundlagen der objektorientierten Programmierung ist das Konzept von Stiften und Mäusen. Hierbei handelt es sich um ein Unterrichtskonzept das eine unter didaktischen Gesichtspunkten entwickelte Klassenbibliothek verwendet. Statt dem Einstieg in die imperative Programmierung geschieht eine Integration entsprechender informatischer Konzepte, wie z.B. Kontrollstrukturen in ein durchgängig objektorientiertes Szenario.
Der Einstieg über visuelle Entwicklungsumgebungen zur Kapselung prozeduraler Programmierung ist ein weiteres mögliches Vorgehensmodell. Jenes steht allerdings in der Kritik, dass die Vermittlung von objektorientiertem Denken vollkommen vernachlässigt wird [Diethelm 2007, S. 25].

BlueJ-Konzept

BlueJ ist eine *integrierte Entwicklungsumgebung (IDE)* für den Informatik Anfangsunterricht. Sie basiert auf dem *Standard Java SDK* und verwendet den Standard *Java-Compiler* sowie die gängige *virtuelle Maschine (JVM = Java Virtual Machine)*. *BlueJ* ermöglicht einen visuellen und unmittelbar parametrisierten Methoden-Aufruf. Dies ermöglicht den Lehrenden, komplexe Themen, wie z.B. textuelle Schnittstellen zu einem späteren Zeitpunkt zu thematisieren und unabhängig davon einen Einstieg in die objektorientierte Programmierung zu ermöglichen [Kölling und Quig 2005]. Die Intention dieses Ansatzes ist nach Diethelm wiederum die Einführung in eine objektorientierte Programmierung. Die Vermittlung objektorientierter Denkweisen wird wieder vernachlässigt [Diethelm 2007, S. 26]. Brinda kritisiert darüber hinaus, dass *BlueJ* ausschließlich Klassendiagramme visualisiert und das Laufzeitobjekte isoliert und ohne Assoziationen untereinander dargestellt werden [Brinda 2004].

Alice-, Greenfood und Scratch-Konzept

Alice Greenfood und *Scratch* sind didaktische Werkzeuge, die einen Einstieg in die Programmierung in der voruniversitären Lehre erleichtern sollen. Die Autoren sehen ihre Software weniger als ein didaktisches Werkzeug für den Einstieg in die objektorientierte

Modellierung als ein Tool zur Erleichterung des Einstiegs in die objektorientierte Programmierung. Obwohl diese visuellen Programmierumgebungen zu unterschiedlichen Zeiten und in unterschiedlichem Kontext entwickelt wurden, sind diese nach Ansicht der Autoren *Cooper*, *Kölling* und *Maloney* vergleichbar. Sie bezeichnen Ihre Werkzeuge als *Initial Learning Environments*.

> „Although designed at different times and in different contexts, these three environments? Alice, Greenfoot and Scratch? can be classified together as sharing similar characteristics. All are visual, all aim to foster immediate engagement in an attractive activity, and all aim to introduce pre-University students to programming. We describe these as „Initial Learning Environments“ [Utting et al. 2010, S. 1].“

Ein beispielhafter Einsatz von *Alice* im Informatikunterricht der Sekundarstufe I wurde von *Dohmen* und *Engbring* durchgeführt. Sie stellten in der anschließenden Evaluation fest, dass *Alice* als alleiniges Hilfsmittel neben einem deutlich spürbaren Motivationszuwachs den Schülern zu keinen grundlegenden Kenntnissen über die objektorientierte Programmierung verhelfen konnte. Der gleichzeitige deklarative Zugang mit vorgefertigten Objekten hingegen, vereinfachte den Zugang spürbar [Dohmen et al. 2009].

Model First

Hierbei handelt es sich um einen Ansatz von *Bennedsen & Caspersen* aus dem Jahr 2004 für einen Kurs im ersten Studienjahr. Dieser sieht vor mit der Modellierung von Klassen zu beginnen und fokussiert hierbei die Übersetzung von *UML-Diagrammen* in objektorientierten *Java-Code* [Bennedsen und Caspersen 2005].

Informationszentrierter Ansatz

In der Sekundarstufe I sieht der *informationszentrierte Ansatz* vor, die Objektorientierung als „Grundstein für den Aufbau angemessener mentaler Modelle und die Verwendung einer sauberen, ausdrucksstarken Terminologie zugrunde zu legen [Hubwieser 2000, S. 59].“ Der Einstieg über die objektorientierte Modellierung erfolgt am Beispiel der Zeichnung eines Zimmers und der Identifikation von Objekten sowie deren Klassifizierung nach Form [Hubwieser 2005].
Der informationszentrierte Ansatz wird im Kapitel 6.2 ausführlicher thematisiert. Hier sollen lediglich einige Ansätze zum Einstieg in die objektorientierte Programmierung genannt werden bevor auf den Ansatz *Objektorientierte Modellierung und Robotik* eingegangen wird.

OO-Modellierung und Robotik

Traditionell sind hier virtuelle Roboter zur Vermittlung von imperativer und prozeduraler Programmierung zu nennen, wie z.B. *Kara* oder *NIKI*.
Neuere Ansätze zur Vermittlung objektorientierter Konzepte mit Hilfe von *LEGO Mindstorms* Robotern, wie z.B. Dietzel und Rinkens, die erste Unterrichtserfahrungen in einem Informatik Differenzierungskurs der Jahrgangsstufe 10 machen konnten [Dietzel und Rinkens 2001]. Das unterrichtliche Vorgehen sah vor, dass Kleingruppen die Roboter bauen und danach erste Beschreibungen über Eigenschaften und Funktionalität der Roboter anfertigen. Ausgehend davon werden die objektorientierten Begrifflichkeiten, wie z.B. *Objekt*, *Attribut* und *Methode* erläutert.
Fujii et. al untersuchten die Wirksamkeit der Vermittlung von *UML- Modellierungskompetenz* und sozialen Kompetenzen, mit Hilfe von problembasierten Lernszenarien. Hierbei wird den Lerngruppen eine vereinfachte *UML-Modellierungsvorlage* zur Verfügung gestellt. Die Zielsetzung bestand darin, einen *LEGO Mindstorms* Roboter zu bauen und dessen Funktionalität zu programmieren. Mit Hilfe der zuvor genannten *UML-Modellierungsvorlage* wurden zunächst die funktionalen Anforderungen (*functional model*) aufgenommen. Ein weiteres Modell, das sog. *Detail-Model* umfasst für jede Systemfunktion bzw. Anforderung den jeweiligen Funktionsnamen, die Funktionsbeschreibung, Vorbedingungen, mögliche rudimentäre Programmabläufe, die jeweiligen Endzustände und Grafiken. Eine weitere sog. *Relationship-Model-Vorlage* hat die Zielsetzung den Zusammenhang und die Interaktion der im Detail-Model beschriebenen Funktionen zu beschreiben. Jene korrespondiert in der Zielsetzung mit *UML-Kollaborationsdiagrammen* [Ishii et al. 2010, S. 26ff].
Die Erhebung wurde im Rahmen eines Programmierkurses im Wintersemester 2007 an der *Chubu University of Engineering* durchgeführt. Die Kursplanung sah drei Phasen vor:

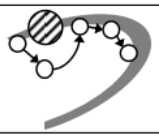
1. Die Lernenden sollten Grundlagenwissen im Hinblick auf *LEGO Mindstorms* und *UML-Modellierung* erlangen. Hierzu erhielten sie Mustervorlagen für die oben beschriebenen *UML-Templates*.

2. Die Lernenden wurden in Gruppen eingeteilt um in einer Art Wettbewerb jeweils einen Roboter zu bauen, der auf möglichst schnelle Weise folgendes leisten muss: Eine Linie bis zu einem Ziel verfolgen und Hindernissen ausweichen. Innerhalb dieser Design- und Implementierungsphase sollte die Dokumentation innerhalb der *UML-Vorlagen* erfolgen. Diese Phase wurde einmal zur Halbzeit des Projekts und zum Ende des Projekts durchgeführt.

Function model

Functions needed to attack a course
1. line trace • judgment of brightness • straight ahead movement on black lines of the track • curved movement on white areas of the track • smooth curves 2. avoiding obstructions • using ultrasonic waves for recognition • behavior when avoiding something 3. traversal of an incline • stable line trace • adjustment of motor power

Detailed model

Function name	avoidance of obstructions
Outline of function	make a line trace quickly while avoiding obstructions
Beginning conditions	when approaching obstructions
Flow of functions	1. substitution of 0 for a 2. turn to the right 3. go forward 4. go forward while turning to the left 5. recognize line 6. substitute ic + 1 for ic 7. quickly make a line trace 8. substitute 0 for ic
Final conditions	when ic is greater than 1,000
Image sketch	

Relationship model

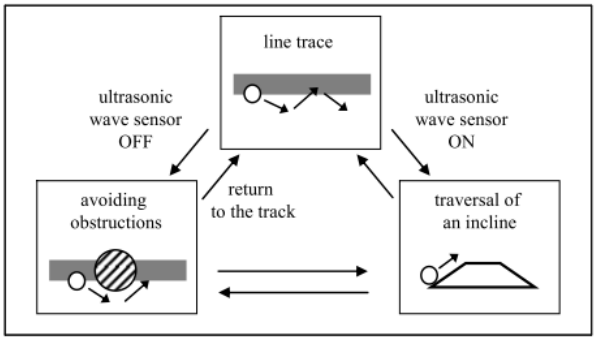


Abbildung 3.11.: UML-Template nach Fujii et. al [Ishii et al. 2010, S. 27]

3. Die dritte Phase fokussierte die Selbst-Reflexion. Die Lernenden waren in diesem Zusammenhang aufgefordert, die Ergebnisse der Gruppenarbeiten auf einem Plakat zu dokumentieren. Dies umfasste die *UML-Vorlage*, ein sog. Problemanalyse-Diagramm und ein Foto des entwickelten Roboters.

Um die *UML-Modellierungskompetenz* zu messen, wurden die Modelle der ersten und zweiten Kurshälfte miteinander verglichen. Hierbei wurden die folgenden Evaluationskriterien für die *UML-Modelle* zugrunde gelegt [Ishii et al. 2010, S. 28ff]:

- Number of functions
- Originality of functions
- Existence of unrelated links
- Improvement of new methods and functions
- Detail of image sketches
- Resemblance to the sample model

Folgende Ergebnisse haben sich im Vergleich von erster und zweiter Kurshälfte ergeben:

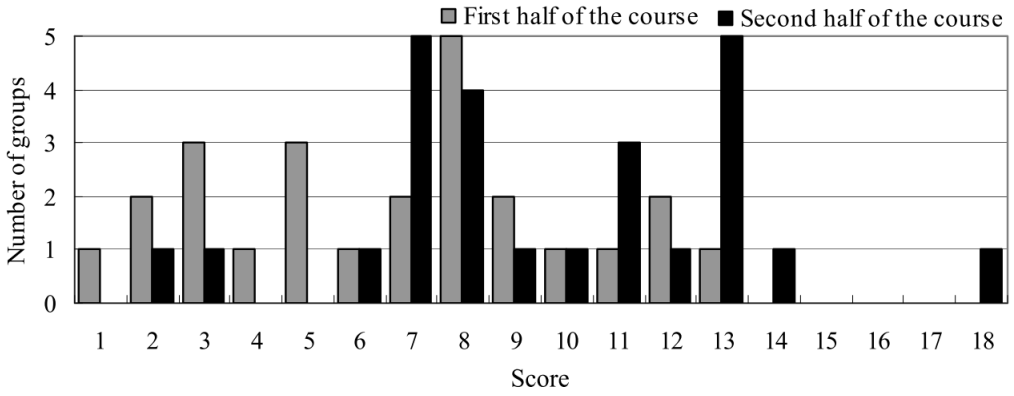


Abbildung 3.12.: Ergebnisse Fujii et. al 1/2 [Ishii et al. 2010, S. 30]

Nach Fuji et. al. zeigen diese Ergebnisse die Effektivität ihres Ansatzes, den Lernenden ein Modellierungs-Template zur Verfügung zu stellen.
Die sozial-kommunikativen Kompetenzen wurden mittels eines Fragebogens anhand des sog. *KISS-18 (Kikuchi's Scale of Social Skills)* zu Beginn (pre-test) und zum Ende (post-test) der Unterrichtsreihe gemessen. Dieser umfasst grob betrachtet drei Haupt-Kriterien sozial kommunikativer Kompetenz:

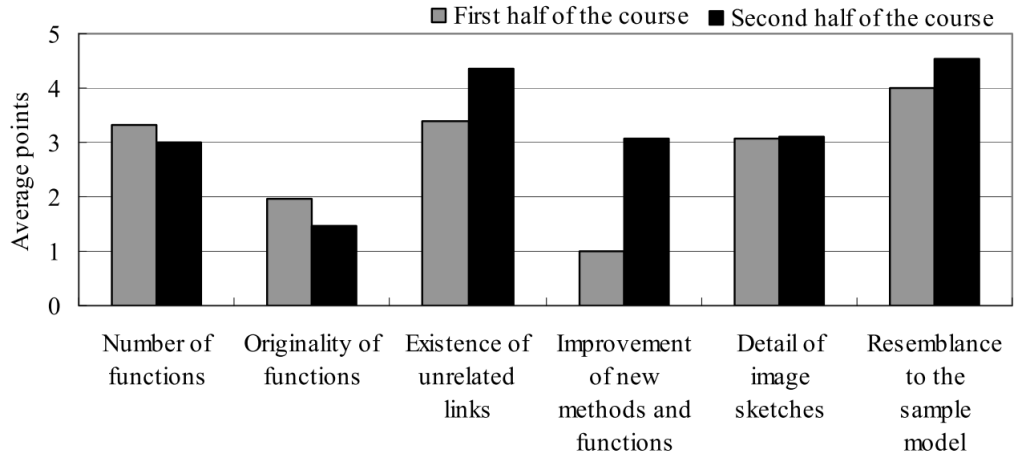


Abbildung 3.13.: Ergebnisse Fujii et. al 2/2 [Ishii et al. 2010, S. 31]

1. Problem solving skills
2. Troubleshooting skills
3. Communication skills

Die folgenden Ergebnisse zeigen nach Fuji et. al, dass die Lernenden im Rahmen dieses problematisierten Lernszenarios breit gefächerte sozial-kommunikative Kompetenzen erwerben konnten [Ishii et al. 2010, S. 30].

```
[Problem solving skills: Pre-test: 17.9; Post-test: 18.6. t(77) = 2.131, p<.05],
[Troubleshooting skills: Pre-test: 15.2; Post-test: 15.9. t(77) = 2.612, p<.05] and
[Communication skills: Pre-test: 20.8; Post-test: 21.7. t(77) = 2.745, p<.01]).
```

Zwischenfazit

Die oben vorgestellten Forschungsergebnisse bestätigen meine Erfahrungen aus der Hochschullehre bei der Verwendung von *LEGO Mindstorms* basierten Lernumgebungen zur Vermittlung von Modellierungskompetenzen. Dies deutet darauf hin, dass ein derartiges Lernszenario auch als Setting für die Evaluation des Messinstruments sinnvoll sein könnte. Dementsprechend soll das Vorgehens- bzw. Vermittlungsmodell *Modellierung & Robotik* bei der Planung der Unterrichtsreihe besondere Berücksichtigung finden.

3.5. Zusammenfassung

Nachdem im vorherigen Kapitel der bildungspolitische Stellenwert von Standards und Kompetenzen deutlich gemacht wurde, hatte dieses Kapitel das Ziel, die Wichtigkeit der objektorientierten Modellierung für die Fachwissenschaft und Fachdidaktik hervorzuheben. In diesem Zusammenhang wurde zunächst der Modellbegriff in Anlehnung an Stachowiak definiert. Diese Definition findet sowohl in Fachwissenschaft als auch in der Fachdidaktik eine große Beachtung. Insbesondere mit Blick auf die Zielsetzung der Arbeit, Facetten der Modellierungskompetenz zu definieren, erwies sich diese definitorische Festlegung als passend, da nach Stachowiak die Modellbildung stets kontextualisiert und mit einer konkreten (komplexen) Zielsetzung erfolgen soll. Im weiteren Verlauf des Kapitels wurde die Wichtigkeit der objektorientieren Modellierung aus fachwissenschaftlicher und fachdidaktischer Perspektive erläutert. Fachwissenschaftlich hat sich die Relevanz der objektorientierten Modellierung insbesondere durch die zunehmende Etablierung der modellgetriebenen Softwareentwicklung ergeben.
Aus fachdidaktischer Sicht ist die objektorientierte Modellierung ebenso ein wichtiger Bestandteil der informatischen Bildung und erfüllt zudem gängige erziehungswissenschaftliche und informatikdidaktische Allgemeinbildungskriterien. Ferner hat dieser Themenbereich darüber hinaus das Potential, den Informatikunterricht an allgemeinbildenden Schulen zu legitimieren.
Neben der Definition, Fokussierung und Legitimation der Modellierung sollte dieses Kapitel auch dazu dienen, die theoretische Grundlage für die weitere Forschungsarbeit zugrunde zu legen. Mit dem Ziel, ein Kompetenzstrukturmodell für die Modellierung zu schaffen, wurde eine theoretische Basis als Orientierungspunkt zur Vorstrukturierung der Dimensionen des Kompetenzmodells festgelegt. Hierbei haben sich informatische Vorgehensmodelle als sinnvolle strukturgebende Basis erwiesen. Aus fachwissenschaftlicher und fachdidaktischer Sicht beschreiben diese Modelle Aktivitäten in wechselndem Kontext und einer klaren Zielsetzung. Dieser theoretische Rahmen soll dementsprechend auch in Übereinstimmung mit der zugrunde gelegten Definition von Kompetenz [Weinert 2002] als normativer Ausgangspunkt verwendet werden, um die Ableitung von Kompetenzen in Kapitel 4 durchzuführen.
Als weitere Erkenntnis dieses Kapitels haben sich Vorgehensmodelle ebenso als theoretische Grundlage zur Entwicklung der Evaluations-Unterrichtsreihe des Kompetenzmessinstruments herausgestellt. Gerade aufgrund der oben beschriebenen Modellcharakteristik und den parallelen zum Weinert'schen Kompetenzverständnis soll dieser Modelltyp auch für das Kapitel 7 theoretisch richtungsweisend sein.

Im weiteren Verlauf des Kapitels wurden konkrete Ansätze für Vorgehensmodelle in der Softwaretechnik und in der Fachdidaktik aufgezeigt. Hierbei wurde zunächst ein Überblick über Vorgehensmodelle in der Softwaretechnik gegeben, ein Vergleich angestellt, die Charakteristika der einzelnen Vorgehensmodelle herausgestellt und die zugrunde liegenden Vorgehensstrategien erläutert. Es wurde deutlich gemacht, welche Einflüsse die unterschiedlichen Modelle auf die Ableitung von Kompetenzen haben könnten und welche als sinnvoll oder weniger sinnvoll erachtet wurden. Hierbei hat sich letztlich der *Rational Unified Process* als Grundlage für die Modellentwicklung in Kapitel 4 ergeben. Die Begründung hierfür ist zusammenfassend die breite Etablierung des *RUP*, die hohe Anpassbarkeit des Ansatzes und die Studien über dessen erfolgreichen Einsatz in der Lehre.
Zur Auswahl einer theoretischen Grundlage für die Unterrichtsreihe zur Evaluation des Messinstruments wurden unterschiedliche didaktische Vorgehens- und Vermittlungsmodelle für den Einstieg in die OO-Modellierung dargestellt und verglichen. Hierbei wurde das Vorgehensmodell *Modellierung & Robotik* als theoretische Grundlage ausgewählt. Die Begründung lieferten Studien über den erfolgreichen Einsatz in der informatischen Bildung und die positiven Erfahrungen des Autors im Einsatz dieser Thematik in der fachdidaktischen Hochschullehre.
Nachdem hiermit die theoretische Basis für die folgenden Kapitel gelegt wurde wird im nächsten Kapitel die Entwicklung der Dimensionen und Komponenten des Kompetenzmodells vorgestellt. Die Erkenntnisse dieses Kapitels sind maßgeblich in die normativ-theoretische Entwicklung des Kompetenzmodells eingeflossen.

4. Theoretische Entwicklung eines Kompetenzstrukturmodells für informatisches Modellieren

Im Folgenden wird die theoretische Ableitung von Kompetenzdimensionen des Strukturmodells und den jeweils aggregierten Kompetenzkomponenten beschrieben. Hierbei wird die theoretische Fundierung der jeweiligen Kompetenzdimensionen dargelegt. In diesem Zusammenhang sind die Erkenntnisse aus dem vorherigen Kapitel maßgeblich in die Kompetenzdimension *K1 Aufgabenbereiche* eingeflossen. Anhand des ausgewählten Vorgehensmodell *RUP* sind die sog. *Prozess Workflows* des *RUP* als Kompetenzkomponenten der Dimension *K1.3 Systemgestaltung* eingeflossen.
Weiterhin gibt das Kapitel einen Einblick in die differenzierte theoretische Ableitung von Kompetenzkomponenten der einzelnen Dimensionen *K1 Aufgabenbereiche*, *K2 Nutzung informatischer Sichten*, *K3 Umgang mit Komplexität* und *K4 Nicht-kognitive Kompetenzen*. Hierbei liegt der Fokus auf den Kompetenzen, die die objektorientierte Modellierung adressieren.
Das in der vorliegenden Arbeit vorgestellte Kompetenzrahmenmodell wurde mit der Zielsetzung entwickelt, fachwissenschaftlich, fachdidaktisch und psychologisch fundierte Kategorien für die Operationalisierung von informatischer Modellierungskompetenz und informatischem Systemverständnis zu gewinnen [Nelles et al. 2009]. Auf Grundlage dieses Kapitels soll in einem weiteren Arbeitsschritt die empirische Bestimmung von Kompetenzaspekten durch Expertenbefragungen (siehe Kapitel 5) stattfinden.

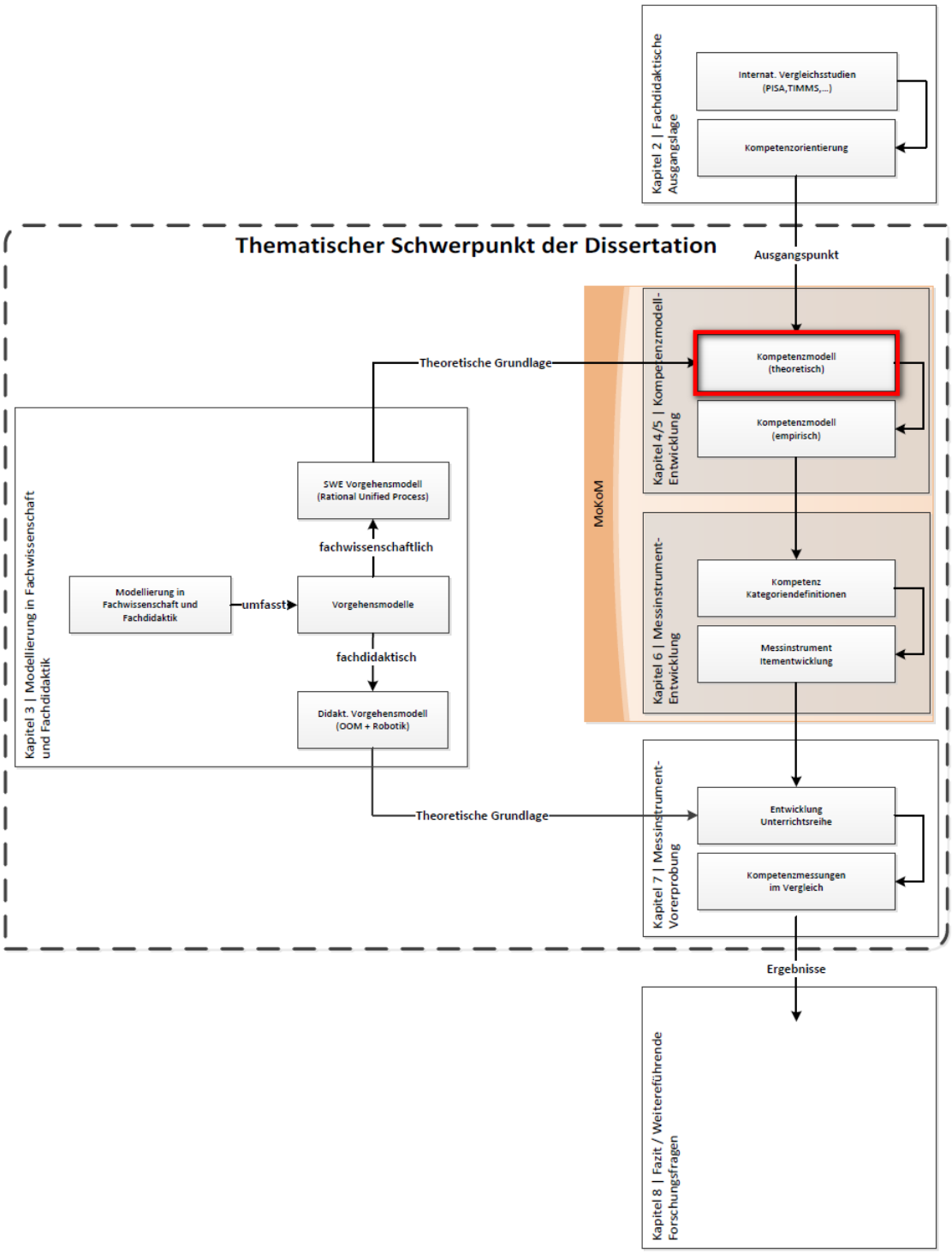


Abbildung 4.1.: Kapitel 4 im Gesamtkontext der Arbeit

4.1. Entwicklung der Kompetenzdimensionen

[1]Das vorliegende Kompetenzrahmenmodell für informatisches Modellieren und Systemverständnis ist in Anlehnung an andere Rahmenmodelle der empirischen Bildungsforschung [Schaper und Hochholdinger 2006] folgendermaßen strukturiert:
Zunächst wird inhaltlich auf einer übergeordneten Ebene zwischen vier Kompetenzdimensionen unterschieden: (*Aufgabenbereiche(K1)*[2], *Nutzung informatischer Sichten (K2)*, *Anforderungen an den Umgang mit Komplexität (K3)* und *nicht-kognitive Kompetenzen (K4)*), die auf weiteren Ebenen durch verschiedene Kompetenzfacetten oder -komponenten inhaltlich differenziert werden. Zwei der Kompetenzdimensionen sind darüber hinaus durch unterschiedliche Stufen bzw. Niveaus der Wissensnutzung charakterisiert. Zunächst werden diese Kompetenzstufen vorgestellt, bevor differenzierter auf die Kompetenzdimensionen eingegangen wird.
Kompetenzen beziehen sich auf Anforderungssituationen, in denen problemlösend gehandelt werden muss und zu deren effektiver Bewältigung Kenntnisse, Strategien, Fähigkeiten und Einstellungen erforderlich sind. Um die im Modell erfassten Kompetenzanforderungen zu veranschaulichen, soll bei der theoriegeleiteten Beschreibung der Kompetenzdimensionen und -stufen auf ein exemplarisches Aufgabenszenario Bezug genommen werden. Basierend auf dem hier entwickelten Rahmenmodell erfolgt unter anderem mit Hilfe dieses Aufgabenszenarios die empirische Verfeinerung der jeweiligen Kompetenzdimensionen und -komponenten. In diesem Zusammenhang kommt ein Interviewverfahren zum Einsatz, das sich methodischer an der *Critical Incident Technique* orientiert. Hierbei wird hauptsächlich ermittelt, wie die Probanden in kompetenzrelevanten Anwendungsszenarien problemlösend handeln und welche Kompetenzen hierfür erforderlich sind.
Das Szenario *Chatsystem* soll in diesem Kapitel nur zur Veranschaulichung der normativ-theoretischen Ableitung von Kompetenzen verwendet werden. Die ausführlichere Beschreibung der einzelnen Interviewszenarien erfolgt in Kapitel 5:

> Sie erhalten den Auftrag, ein verteiltes *Chatsystem* zu entwickeln. Im Rahmen der Designphase sollen Sie die potenziellen Programmmodule (Klassen) jeweils dem Client oder Server zuordnen. Zudem erhalten Sie als Projektleiter die Aufgabe, nach Abschluss der Analyse- und Designphase, eine zeitlich parallele Implementierung von Client- und Server-Softwarekomponenten in

[1]Das Kapitel 4.1 enthält die für die Modellierungskompetenz relevanten Anteile aus der eigenen Veröffentlichung [Nelles et al. 2009].
[2]Die Bezeichnung Aufgabenbereiche wurde in der späteren Verfeinerung des Modells außerhalb dieser Dissertation nicht mehr verwendet. Hier hat eine Optimierung der Terminologie im Sinne des Weinert'schen Kompetenzverständnis stattgefunden.

Projektgruppen zu koordinieren.

4.1.1. Kompetenzstufung

Die ersten beiden Dimensionen *K1 Aufgabenbereiche* und *K2 Nutzung informatischer Sichten* sind hinsichtlich der Kompetenzausprägungen gestuft, wobei diese Stufung den *Anforderungsbereichen I, II* und *III* der *Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA)* im Fach Informatik entspricht [KMK (Hrsg) 2004]. Die Stufen *Wissen*, *Anwenden* und *Gestalten* lassen sich wie folgt charakterisieren:
Auf der Kompetenzstufe *Wissen* sind Lernende zur Wiedergabe bekannter Sachverhalte, zur Beschreibung und Darstellung bekannter Verfahren, Methoden und Prinzipien der Informatik imstande, während sie auf der Kompetenzstufe *Anwenden* bereits befähigt sind, selbständig bekannte Sachverhalte zur Bearbeitung unbekannter Frage- und Problemstellungen zu verwenden und bekannte Verfahren, Methoden und Prinzipien der Informatik zur Lösung einer Aufgabe aus einem neuen Problemkreis anzuwenden. Lernende auf der Kompetenzstufe *Gestalten* treffen eine bewusste und selbständige Auswahl geeigneter Methoden und Verfahren in für sie neuartigen und komplexen Problemsituationen.

4.1.2. K1 Aufgabenbereiche

Im Hinblick auf die geforderte aufgaben- und anforderungsbezogene Ausrichtung von Kompetenzmodellen werden im Rahmen der ersten Kompetenzdimension verschiedene Grundanforderungen bzw. -kompetenzen in Bezug auf informatisches Modellieren und informatisches Systemverständnis beschrieben. Diese beziehen sich auf die Anwendung von Informatiksystemen, Systemverständnis und den Prozess der Systemgestaltung als Kompetenzkomponenten. Die Kompetenzstufung *Wissen*, *Anwenden* und *Gestalten* bezieht sich auf alle drei Kompetenzkomponenten, jedoch soll sie in diesem Zusammenhang lediglich an der Kompetenzkomponente *Systemverständnis* exemplarisch erörtert werden. Die Komponente *System anwenden* repräsentiert die Befähigung zur Anwendung eines Informatiksystems. Dazu gehören Auswahl geeigneter Anwendungsprogramme als Werkzeuge, Zielgerichtetheit sowie Angemessenheit der Eingaben und der Reaktion auf Ausgaben des Rechners. Ein *Chatsystem* soll beispielsweise als Lernmedium bewusst angewendet werden können. Es eignet sich für die synchrone Kommunikation der Lernenden untereinander sowie für die Kommunikation mit Lehrenden. Bei geeigneter Speicherung der Beiträge wird darüber hinaus asynchrone Kommunikation unterstützt. Da Beiträge parallel abgesendet werden können, entstehen jedoch häufig Missverständnisse, die auf die Positionierung der Beiträge zurückzuführen sind. Wissen um Möglichkeiten und

Grenzen von Informatiksystemen ist deswegen notwendig. Grundlage dafür ist eine informatische Bildung, aber auch (Bedien-)Fertigkeiten. Entmystifizierung des Rechners und Abwendung des Gefühls des „Ausgeliefertseins“ gegenüber technischen Systemen sind zu erreichen. Dafür ist die bewusste Anwendung mit typischen Informatiksystemen von den Lernenden gefordert. Dabei ist zwischen Anwendungssoftware und Systemsoftware zu unterscheiden [GI 2008], und es sind immer auch grundlegende Prinzipien der Funktionsweise zu behandeln, indem möglichst viele Arten von Fehlfunktionen thematisiert werden. Innerhalb der Komponente *Systemverständnis* wird ein Verstehen der jeweiligen Bestandteile eines Informatiksystems und der zugrunde liegenden informatischen Prinzipien adressiert. Für Informatiksystemverständnis sind die Analyse der inneren Struktur und des nach außen sichtbaren Verhaltens von Informatiksystemen notwendig [Schubert und Stechert 2007]. Ziel ist die Vermeidung von Versuch-Irrtum-Strategien. Anforderungen an Lernende zur Erlangung des Informatiksystemverständnisses können das systematische Erkunden und Experimentieren mit Informatiksystemen beinhalten. Solche Vorgehensweisen können bei geeigneter Reduktion und Gestaltung in Anlehnung an Methoden der Fachwissenschaft umgesetzt werden. Experimente, in denen die Lernenden selbständig Hypothesen formulieren, unterstützen die Kombination der Perspektiven *Außensicht* und *Innensicht*. Hierbei sind Lernende beispielsweise gefordert, funktionale Modelle zu nutzen, um das beobachtete Verhalten des Systems zu formalisieren. Außerdem werden hierdurch *fundamentale Ideen* und Strukturmodelle identifiziert. Ziel ist es, dass die Lernenden verstehen, wie die innere Struktur das Systemverhalten beeinflusst und warum sie in der Entwurfsphase gewählt wurde. Durch systematisches Testen kann das Systemverhalten analysiert werden. Ist der Quelltext gegeben, so können Lernende im Sinne eines *Whitebox-Tests* das Systemverhalten auf Ursachen untersuchen. Das Systemverhalten kann klassifiziert werden, während im Quelltext besonders wichtige Stellen variiert werden. Exemplarisch werden im Folgenden Lernziele zu den drei Anforderungsstufen formuliert.

Für die Stufe *Reproduktion von Wissen* bedeutet dies, dass die Lernenden befähigt werden müssen, Funktionsweise und Aufbau bekannter Informatiksysteme zu beschreiben [Magenheim 2005], z.B. des *Chatsystems* mit *Client-Server-Architektur*.

Auf dem Niveau der *Anwendung* müssen Lernende in der Lage sein, die bekannte Vorgehensweise des systematischen Testens auf ähnliche Informatiksysteme aus einem bekannten Bereich einzusetzen, z.B. Unterschiede zwischen *Instant Messaging* oder *webbasierten Chaträumen*.

Zu der Stufe des Gestaltens ist im Kontext des Systemverständnisses das bewusste und selbstständige systematische Testen unbekannter Informatiksysteme zu zählen. Dabei

sind die gelernten Vorgehensweisen zur Bewältigung der Aufgabe selbstständig auszuwählen und an das unbekannte Informatiksystem und den Kontext anzupassen, z.B. das *Chatmodul* einer Lernplattform.
Die Kompetenzkomponente *Systemgestaltung* soll eine Befähigung zur Konstruktion und zum Reengineering von Informatiksystemen widerspiegeln. Sie untergliedert sich in die Phasen der Systemgestaltung in Anlehnung an den *Rational Unified Process (RUP)* [Rational Software Corporation IBM. 1998]. Dieser stellt, wie in Kapitel 3 erläutert, einen möglichen Ansatz (neben anderen) dar, um viele in der Software-Engineering-Praxis erfolgreich eingesetzte Vorgehensweisen und Modellierungstechniken in einem umfassenden Prozess zu integrieren. Die Unterkomponenten von Systemgestaltung spiegeln die sog. Prozess-Workflows des *RUP* wider. Hierbei werden sowohl die Abläufe von Aktivitäten als auch die Interaktion zwischen relevanten Personen (Entwicklern, Auftraggebern und Benutzern) beschrieben. Bei der Entwicklung eines *Chatsystems* können Lernende und Lehrende diese Rollen im Rahmen der Projektarbeit einnehmen. Es muss geklärt werden, welche Kenntnisse und Fähigkeiten zum Design des Client-Server-Systems notwendig sind. Weiterhin muss entschieden werden, welche Modellierungstechnik, z.B. der *Unified Modeling Language (UML)*, während eines bestimmten Arbeitsablaufs zum Einsatz kommt. Die Ableitung von Kompetenzkomponenten auf der Grundlage der Prozess-Workflows ist dadurch begründet, dass der *RUP* ein umfassendes Prozess-Framework zur Verfügung stellt, welches es für das jeweilige Einsatzszenario anzupassen gilt.

Abbildung 4.2.: K1 Aufgabenbereiche

4.1.3. K2 Nutzung informatischer Sichten

Zur Bewältigung von Aufgaben bzw. Anforderungen der Systemanwendung, des Systemverständnisses und der Systemgestaltung ist es außerdem erforderlich, dass die Lernenden befähigt werden, Informatiksysteme aus unterschiedlichen Perspektiven zu betrachten. Damit ist nicht nur der Erwerb unterschiedlicher Wissensbestände zur Systembeschreibung und -analyse, sondern auch die Befähigung zum flexiblen Wechseln zwischen diesen Sichtweisen – je nach Aufgabenanforderung – impliziert.
Informatische Bildung mit einem Schwerpunkt auf informatischem Modellieren und Systemverständnis sollte daher auch die kognitive Flexibilität der Lernenden im Umgang mit solchen Systemen fördern.
Informatiksysteme können z.B. anhand ihres nach außen sichtbaren Verhaltens, ihrer inneren Struktur analysiert werden. Denning beschreibt hierzu fünf Perspektiven auf die Informatik (*windows of computing mechanics*). Diese erweitert er zu sieben Kategorien der Informatik: computation, communication, coordination, automation, recollection, evaluation und design [Denning 2007]. Denning setzt die Kategorien mit den Hauptfunktionen von Informatiksystemen in Verbindung:

> „These categories cover the main functions of computing systems [Denning 2007, S. 15]."

Damit wird implizit eine Antwort auf die Frage geliefert, welche Sichten Lernende auf Informatiksysteme einnehmen können und müssen, um ein konsistentes Gesamtbild von Informatiksystemen zu bekommen. Nach Denning können die Perspektiven auf die Informatik mit einem fachdidaktischen Sichtenkonzept kombiniert werden. Dementsprechend sollte der zu explorierende Lerngegenstand in unterschiedlichen, ggf. interaktiv erfahrbaren und synchronisierten Sichten dargestellt werden. So ist eine zuverlässige Datenübertragung im *Chatsystem*, wie sie unter der Perspektive der Kommunikation (communication) analysiert wird, unabdingbar für die Kommunikation der Chatteilnehmer, während bei der digitalen Sprachübertragung einzelne Bitfehler toleriert werden können. Diesbezüglich kann der Entwurf (design) des *Chatsystems* untersucht werden. Die Nebenläufigkeit von Prozessen bzw. von eingegebenen Nachrichten in das *Chatsystem* können im Rahmen der Koordinierung (coordination)betrachtet werden. Die Sichten überschneiden sich jedoch. Dies wird dadurch verstärkt, dass in jeder Sicht mehrere Modelle und Darstellungsformen zur Hervorhebung eines Aspektes zum Einsatz kommen, die wiederum auch für andere Perspektiven genutzt werden. Daher sind die Sichten schwierig zu synchronisieren und bedürfen einer weiteren Strukturierung. Ausgehend von diesen Perspektiven lautet die Frage, welche und ggf. wie viele Perspektiven auf Informatiksysteme

zu beherrschen sind, um bei deren Einsatz Anwendungsprobleme zu lösen. Nievergelt erstellt dazu ein Schichtenmodell der Informatik: den Informatikturm. Als oberste und größte Ebene nennt er die Anwendungsmethodik, die beschreibt, wie Informatiksysteme zur Lösung eines Anwendungsproblems eingesetzt werden. Darunter liegt die Ebene der Systemrealisierung mit Entwurf und Implementierung in Hard- und Software. Nievergelt zählt zu dieser Ebene das Programmieren im Großen. Eine weitere Ebene tiefer liegt die Algorithmik, d. h. das Programmieren im Kleinen. Auf der untersten Ebene liegen fundamentale theoretische Erkenntnisse der Informatik [Nievergelt 1995].
Zur Strukturierung der Perspektiven wurde beschlossen, den Blick auf das System (Außensicht) und den Blick in das System (Innensicht) zu unterscheiden. Um das nach außen sichtbare Verhalten von Informatiksystemen zu verstehen, seien als Beispiele Erwartungshaltungen und Handlungsmuster der Nutzer sowie Usability angeführt. Brauer und Brauer fordern mit Blick auf komplexe Informatiksysteme die Informatik auf, vernetztes Denken und Handeln mit zu gestalten:

> „Ja, es wird immer klarer, daß sequentielle geschlossene Systeme sehr grobe Idealisierungen darstellen, daß aber konkrete Systeme i. a. verteilt, offen (interaktiv) und nichtsequentiell sind. Deshalb ist die Änderung der Denkgewohnheiten nötig; vernetztes, nicht länger sequentielles Denken wird gebraucht [Brauer und Brauer 1992, S. 17]."

Es ist daher notwendig, Lernende im Umgang mit Perspektiven zu befähigen; diese Perspektiven lassen die Verteilung der Systemkomponenten und deren Vernetzung erkennen. Für den Blick in das System können deshalb Architekturmodelle betrachtet werden, denn sie verdeutlichen Aufbau und grundsätzliche Arbeitsweise von Informatiksystemen. Darunter fallen Rechnerarchitekturen wie Entwurfsmuster oder Schichten, z.B. eine 3-Schichten-Architektur bestehend aus Graphical User Interface (GUI), Fachkonzept und Datenhaltung. Dazu kommen theoretische Maschinenmodelle und Rechenmodelle, z.B. Kalküle und Sprachtypen wie imperativ, funktional, prädikativ, objektorientiert sowie entsprechende höhere Programmiersprachen mit Algorithmen und Datenstrukturen.
Ausgehend vom fachdidaktischen Konzept der fundamentalen Ideen der Informatik können Wirkprinzipen in Informatiksystemen bei gleichzeitiger Sicherung des Bildungswertes beschrieben werden [Schubert und Schwill 2004]. Der Erfolg der Förderung des Informatiksystemverständnisses hängt davon ab, ob es gelingt, komplexe Zusammenhänge, wie sie zwischen der inneren Struktur und dem Verhalten von Informatiksystemen bestehen, zu strukturieren, um Aneignung und Anwendung des Wissens zu unterstützen. Solche Wissensorganisation fördert den Erwerb, die Kommunikation über den Wissensbereich

und die Anwendung des Wissens.
Ausschlaggebend hierfür ist die Visualisierung als externe Repräsentationsform. Grafische Beschreibungsmittel wie die *UML* ermöglichen daher weitere Perspektiven auf Informatiksysteme und unterstützen das Verstehen. Analog zu K1 können die Lernzielebenen Wissen, Anwenden und Gestalten adressiert werden. Da *Chatsysteme* sowohl synchrone als auch asynchrone Kommunikation unterstützen, hängt die Erwartungshaltung als Außensicht im hohen Maße von den Zielen des Anwenders im Umgang mit der Software ab (z.B. Freizeit, Lernprozess usw.). Bei der Entwicklung eines verteilten *Chatsystems* sind einige der beschriebenen Innensichten relevant: Bei der Zuordnung von Programmkomponenten zu Client und Server ist es beispielsweise unabdingbar, sich mit der Architektur (z.B. Client/Server-Prinzip) zu beschäftigen. Ferner muss man sich mit Aspekten der Verteilung und Vernetzung von Komponenten befassen, um das zu entwickelnde *Chatsystem* später auf verschiedenen verteilten Rechnern einsetzen zu können. Verfolgt man in einem weiteren Schritt die Implementierung von Programmmodulen, wird deutlich, dass man sich mit Kalkülen und Sprachtypen befassen muss. Hierbei müssen sich Entwickler notwendigerweise mit Algorithmen und Datenstrukturen beschäftigen.

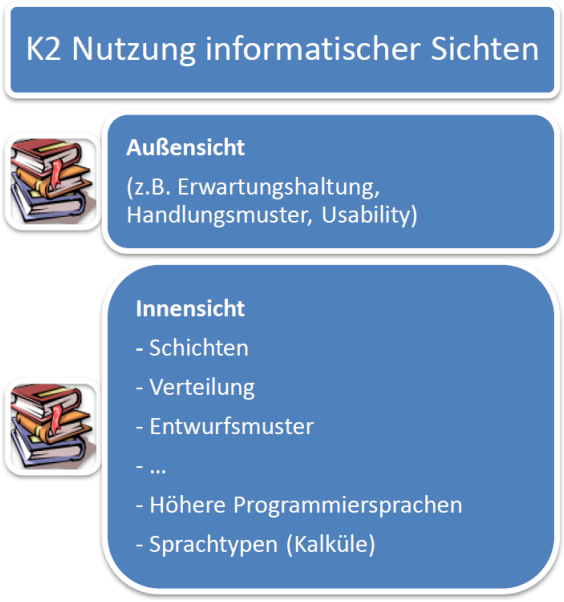


Abbildung 4.3.: K2 - Nutzung informatischer Sichten

4.1.4. K3 Anforderungen an den Umgang mit Komplexität

Anforderungen an informatische Kompetenzen werden auch durch die Komplexität der Systeme, die jeweils thematisiert oder verwendet werden, determiniert. Die Frage, die sich in diesem Zusammenhang stellt, ist allerdings, welche Faktoren die Komplexität eines Informatiksystems ausmachen. Ein einfaches Kriterium ist die Anzahl an Komponenten eines Systems. Hinsichtlich eines *Chatsystems* bedeutet dies, dass ein kleines System beispielsweise zwei Clients und einen Server als Komponenten umfassen könnte. Damit gekoppelt ist der Grad der Vernetzung unterschiedlicher Komponenten, die in Architekturmodellen beschrieben werden. Die Clients sind ausschließlich über den Server miteinander vernetzt. Komponenten und deren Vernetzung werden als Facetten zur Beschreibung von Komplexitätsanforderungen herangezogen.
Brauer und Brauer weisen darauf hin, dass Algorithmen als formalisierbare Dimension von Informatiksystemen nicht ausreichen, um Prozesse in Rechnern, parallele Prozesse oder verteilte Systeme in ihrer ganzen Komplexität zu erfassen [Brauer und Brauer 1992]. Menschen und Maschinen machen Eingaben in Informatiksystemen und beeinflussen damit die Systemzustände und den auf Protokollen basierenden Datenaustausch. Das Systemverhalten ist damit im Detail kaum noch deterministisch nachvollziehbar.
Man spricht in diesem Zusammenhang auch von der Intransparenz komplexer Systeme. Zur Messung des Interaktivitätsgrades eines Informatiksystems bietet sich eine Stufung an, die nicht am Informatiksystem, sondern an den kognitiven Prozessen des Lernenden orientiert ist. Ein *Chatsystem* ist beispielsweise geeignet, um schriftlich zu kommunizieren, die Nutzung grafischer Darstellungen zur Kommunikation wird jedoch meist nicht unterstützt. Auch sind intelligente Systemrückmeldungen meist auf eine Rechtsschreibprüfung beschränkt.
Im *Computing Curricula Information Technology* der *ACM* wird Abstraktion als wichtigstes Mittel zur Komplexitätsbewältigung hervorgehoben und es werden weitere genannt:

> „The ability to manage complexity through abstraction & modeling, best practices, patterns, standards, and the use of appropriate tools [ACM 2008, S. 47]."

Diese Konzepte liefern Hinweise, wie Lernende adäquat mit Komplexitätsanforderungen bei Informatiksystemen umzugehen haben.
Außerdem nehmen wir an, dass auch der Grad der Verteilung von Systemkomponenten (lokal versus verteilt) die Komplexitätsanforderungen determiniert. Die scheinbare Abgeschiedenheit eines Chatraums täuscht unerfahrenen Anwendern vor, dass es sich um

ein lokales statt verteiltes System handelt.

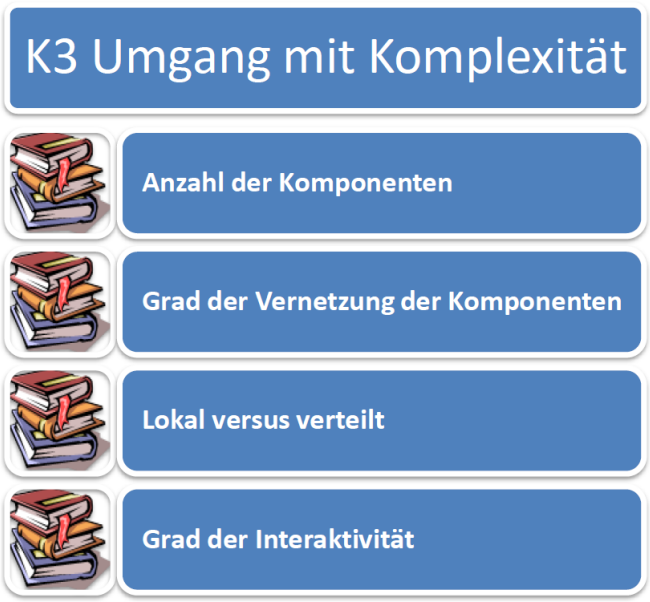


Abbildung 4.4.: K3 - Anforderungen an den Umgang mit Komplexität

4.1.5. K4 Nicht-kognitive Kompetenzen

In Anlehnung an das Kompetenzverständnis nach Weinert umfassen Kompetenzen neben kognitiven Bereichen auch nicht kognitive Kompetenzen, wie Einstellungen, sozialkommunikative Fähigkeiten und motivationale und willensmäßige Fähigkeiten [Weinert 2002]. Dementsprechend wird in diesem Unterkapitel erläutert, in welcher Art und in welchem Ausmaß nicht-kognitive Kompetenzfacetten beim informatischen Modellieren und informatischem Systemverständnis gefordert bzw. zu entwickeln sind.
Die Erwartungshaltungen an den Umgang mit einem Informatiksystem sind insofern relevant, weil sie einen wesentlichen Bedingungsfaktor für das erfolgreiche Gelingen des Lernens informatischer Inhaltsbereiche darstellen.
Als wichtige Fähigkeit sind sowohl die Wahrnehmung eines Informatiksystems im sozialen Kontext als auch die Entwicklung eines prospektiven Blicks auf das System gefordert.
Die Wichtigkeit, Informatiksysteme im Kontext wahrnehmen und antizipieren zu können, wird bei Betrachtung des *soziotechnischen Systembegriff* und den sich daraus ergebenden Implikationen für die Betrachtung von Informatiksystemen deutlich [ACM 2008].
Im Rahmen der arbeitspsychologischen Expertiseforschung wurde untersucht [Sonnentag 2006], worin sich *High-Perfomer* von *Medium-Performern* im Bereich Software-Design

und -entwicklung unterscheiden. Die Ergebnisse weisen darauf hin, dass *High-Perfomer* im Vergleich zu *Medium-Perfomern* über eine bessere Kommunikation und Kooperation verfügen, sie ferner nicht mehr Zeit auf die fachspezifischen Tätigkeiten der Softwareentwicklung aufwenden, sehr wohl aber engagierter während Projekttreffen und Konsultationen sind. *High-Performer* verfügen außerdem über höher entwickelte interpersonale Fähigkeiten als *Medium-Performer*.
Sozial-kommunikative Fähigkeiten und Kooperationsbereitschaft stellen folglich bedeutsame Kompetenzanforderungen in der Informatikdomäne (insbesondere unter Berücksichtigung des sozio-technischen Systembegriff) dar. Hierbei muss die Teamfähigkeit der Lernenden entwickelt werden, um vielfältige sozial-kommunikative Anforderungen, z.B. bei der kooperativen Systemgestaltung, bewältigen zu können. Die Lernenden sollten ebenfalls die Fähigkeit zum Perspektivwechsel (Empathie) entwickeln, um etwaige Rollen (z.B. Benutzer, Entwickler, Auftraggeber) und die jeweiligen subjektiven Sichten auf das Informatiksystem nachvollziehen und deuten zu können. Dabei ist zu beachten, dass es sehr unterschiedliche und dennoch schlüssige Sichten auf ein Informatiksystem geben kann. Die Wichtigkeit, ein derartiges Einfühlungsvermögen zu entwickeln, wird somit offensichtlich.
Neben Einstellungen und sozial-kommunikativen Fähigkeiten müssen bei den Lernenden entsprechende motivationale und volitionale Fähigkeiten entwickelt werden. Diese sind eng mit Prozessen der Selbstregulation verknüpft und stellen die Vorbedingung zu selbständigem Handeln und dem entschlossenen und gewissenhaften Verfolgen der eigenen Ziele dar. Der Grad der Motivation spiegelt eine wesentliche Facette der Handlungskompetenz in diesem Kontext wider. Daher gilt es, eine Offenheit für neue Ideen und Anforderungen zu fördern. Darüber hinaus soll die Bereitschaft zu lernen und sich zu engagieren gestärkt werden. Motivationale Kompetenzen befähigen die Lernenden dazu, ihr Wissen in komplexen Situationen erfolgreich und verantwortungsvoll anwenden zu können. Ferner sind sie entscheidend dafür, dass die Lernenden ihre Kompetenzen bezüglich des informatischen Modellierens und Systemverständnisses auf einer hohen Ausprägungsstufe entwickeln.
Mit Blick auf das oben beschriebene Aufgabenszenario zeigt sich, inwiefern die Erwartungshaltung von Entwicklern Einfluss auf ein effektives Implementieren hat: Sollen die Lernenden sich beispielsweise in eine objektorientierte Programmiersprache einarbeiten, so ist eine ablehnende Haltung diesem Sprachtyp gegenüber sicherlich als erschwerend einzustufen. Eine positive Einstellung hingegen wird der effektiven Einarbeitung förderlich sein. Auch die sozial-kommunikativen Fähigkeiten sind von Relevanz: Eine zeitgleiche Implementierung von Programmmodulen kann nur dann stattfinden, wenn eine ange-

messene Kommunikation in und zwischen den verschiedenen Entwicklergruppen erfolgt. Hierbei müssen beispielsweise Absprachen über Schnittstellen getroffen werden. Außerdem wird von Softwareentwicklern Empathie erwartet, um das Nutzerverhalten eines zu implementierenden Informatiksystems zu antizipieren. In Bezug auf das Beispielszenario ist es demnach wichtig, sich in die Rolle des Benutzers hineinzuversetzen und dessen künftiges Verhalten im Umgang mit dem *Chatsystem* in Entwurfsentscheidungen (z.B. bei der Entwicklung der Benutzungsschnittstelle) mit einzubeziehen. Im Rahmen eines solchen arbeitsaufwändigen Projektes im Team ist sowohl zielorientiertes, frustrationstolerantes und verantwortungsvolles Verhalten als auch die Bereitschaft, neues, nicht vorhandenes Fachwissen selbständig zu erwerben, von erfolgsrelevanter Bedeutung.
In Anbetracht dieser theoretischen Überlegungen ergeben sich wichtige Implikationen bei der Analyse und im Umgang mit Informatiksystemen: Während man die technischen Aspekte eines Informatiksystems betrachtet (z.B. Hard- und Software), gilt es, dessen soziale Aspekte und Auswirkungen mit in die Betrachtung einzubeziehen.

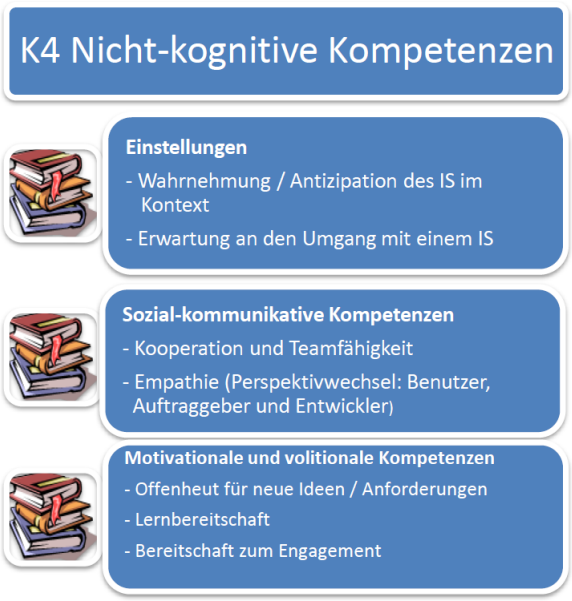


Abbildung 4.5.: K4 - Nicht-kognitive Kompetenzen

Die Dimensionen K1 bis K4 setzen sich zu einem theoretisch abgeleiteten Rahmenmodell (Kompetenzstrukturmodell) zusammen.

Abbildung 4.6.: Theoretisch Hergeleitetes Rahmenmodell

4.2. Förderung von Schlüsselkompetenzen

[3]Neben der Ableitung informatik-spezifischer Kompetenzaspekte im Rahmenmodell, bestand der Anspruch, allgemeinbildende Schlüsselkompetenzen mit zu berücksichtigen. Dementsprechend wurde theoretisch untersucht, inwiefern sich die im Kompetenzrahmenmodell enthaltenen fachspezifischen Kompetenzaspekte zur Förderung von Schlüsselkompetenzen (in Anlehnung an OECD Definition and Selection of Competencies) eignen. Im Rahmen der vorliegenden Arbeit soll hierbei der Fokus auf die informatische Modellierungskompetenz und deren möglicher Zuordnung zu Schlüsselkompetenzen gelegt werden.

4.2.1. Allgemeinbildender Wert des Informatikunterrichts an Gymnasien der Sekundarstufe II

Aufgrund seines Ursprungs innerhalb der Ingenieurwissenschaften ist Informatik das wichtigste und geeignetste Fach, um IT-relevante Kompetenzen im Rahmen der Sekundarstufe zu vermitteln. Insbesondere im Hinblick auf die ständig wachsende Relevanz der IT im schulischen, privaten und beruflichen Alltag gilt es, im Rahmen des Informatikunterrichts an allgemeinbildenden Schulen, den Umgang mit dieser Technologie in verschiedenen sozialen Umgebungen zu vermitteln.
Um einen genaueren Einblick in die damit verbundenen Lehr- und Lernziele, Unterrichtsinhalte und die Methodik der Vermittlung zu erhalten, ist es hilfreich, den Begriff des *Informatiksystems* im Sinne der systemtheoretischen Didaktik der Informatik in Paderborn zu definieren:

> „Unter Soziotechnischen Informatiksystemen (IS) verstehen wir die Vereinigung von Software (inkl. der grafischen Benutzeroberfläche GUI), Hardware und eines assoziierten sozialen Systems von Personen die miteinander und mit dem technischen Part des Informatiksystems interagieren [Magenheim 2000, S. 42]."

Dieser Begriff hat seinen Ursprung sowohl in der Informatik als auch in der technischen Soziologie [Ropohl 1999].
Insbesondere im Rahmen von Software Engineering Prozessen, ist die Modellierung ein Prozess mit hohem kommunikativen und interaktiven Anteil, der eine intensive Zusammenarbeit zwischen Entwicklern und Kunden bzw. Auftraggebern erfordert. Hierbei kre-

[3]Das Kapitel 4.2 enthält die für die Modellierungskompetenz relevanten Anteile aus der eigenen Veröffentlichung [Kollee et al. 2009].

ieren Software-Entwickler Modelle, die einen prospektiven Blick auf die künftige Systemfunktionalität eines Informatiksystems gewähren. Ferner wird deren Integration in die Geschäfts- und Arbeitsdomäne sowie in den jeweiligen sozialen Kontext aufgezeigt [ACM 2008].
Damit der Informatikunterricht allgemeinbildende Kompetenzen vermitteln kann, sollte die Modellierung innerhalb von Software Engineering Prozessen als ein zentrales Thema behandelt werden. Da es sich bei der Gestaltung von Informatiksystemen um eine Tätigkeit handelt, die von vielen sozialen Faktoren abhängig ist, muss der Lerner ein Prozessverständnis entwickeln. Dies umfasst insbesondere die Interessen verschiedener kooperierender Stakeholder (Entwickler, Benutzer und Auftraggeber), deren Ziele und Absichten sowie die daraus resultierenden Designentscheidungen im Hinblick auf das Informatiksystem [ACM 2008].
Indem man diese mit dem Prozess der Modellierung verknüpften sozialen Aspekte und Folgewirkungen bewusst thematisiert, kann der Informatikunterricht einen Beitrag zur Entwicklung von Schlüsselkompetenzen (z.B. die interaktive Nutzung von technischen Systemen) leisten. Ein derartiger Unterricht bietet im Hinblick auf Klafkis epochaltypische Probleme einen allgemeinbildenden Wert: Nach Klafki ist der zu vermittelnde Lerninhalt im Hinblick auf seine Vergangenheits-, Gegenwarts- und Zukunftsrelevanz zu legitimieren. Dies geschieht im Hinblick auf die Modellierung im Kontext von Software Engineering Prozessen durch die Thematisierung von epochaltypischer Probleme, z.B. sozialer Folgewirkungen von Designentscheidungen [Horton 2007].

4.2.2. DESECO-Schlüsselkompetenzen

Aufgrund der stetig steigenden Anforderungen im Alltag, den globalen Märkten und dem sog. *Informationszeitalters* sind Schlüsselkompetenzen von fundamentaler Bedeutung für jedes Individuum. Schlüsselkompetenzen sind in diesem Zusammenhang mehr als Wissen und Fähigkeiten. Sie umfassen komplexe psychologische Anforderungen und sollen es dem Individuum ermöglichen, den unterschiedlichen Anforderungen als Mitglied einer modernen Gesellschaft gerecht zu werden. Schlüsselkompetenzen können erlernt werden und können als wichtige Bereiche der Allgemeinbildung verstanden werden [OECD 2005].

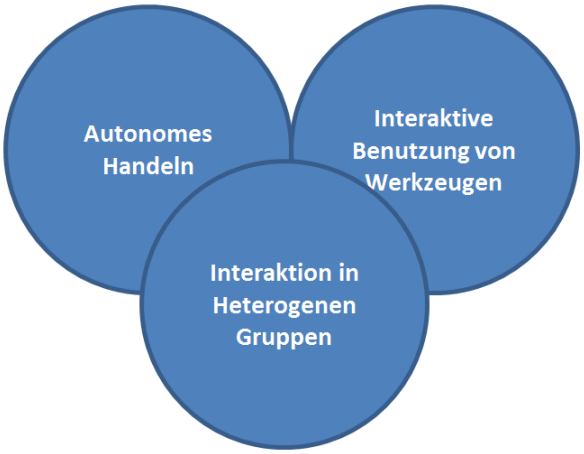


Abbildung 4.7.: DESECO Schlüsselkompetenzen

Eine mögliche Kategorisierung von Schlüsselkompetenzen wird im Ansatz des OECD Projekts *Definition and Selection of Competencies (DeSeCo)* vorgeschlagen. Hierbei werden Schlüsselkompetenzen in drei Gruppen kategorisiert [OECD 2005]:

- *Using Tools Interactively*
  Diese Gruppe umfasst Kompetenzen, die es einem Individuum ermöglichen *Sprache und Symbole*, *Wissen und Information* sowie *Technologie* interaktiv verwenden zu können. Der interaktive Umgang mit Technik erfordert Wissen und ein Bewusstsein, wie die zeitgemäße Nutzung von Technologie und deren Integration in den Alltag eines Individuums geschieht.

  Fördert man eine derartige interaktive Nutzung von IT hat dies Auswirkungen auf die Art und Weise, wie Individuen (z.B. ortsunabhängig mit Hilfe von IT) miteinander arbeiten, sich Zugang zu umfassenden Informationsressourcen verschaffen und miteinander interagieren, indem globale IT-gestützte soziale Netzwerke aufgespannt werden. Die interaktive Nutzung von IT bietet die Chance, benachteiligte Personen nicht auszuschließen und die geschlechtliche Gleichbehandlung zu gewährleisten.

- *Interacting in Heterogeneous Groups*
  Dieser Bereich umfasst Schlüsselkompetenzen, die eng mit sozialen, interpersonalen Fähigkeiten und Fertigkeiten verknüpft sind. Dementsprechend ermöglichen derartige Kompetenzen die Integration eines Individuums in pluralistischen Gesellschaften und den Umgamg und die Bewältigung komplexer Anforderungen. Diese umfassen den *sozial-konformen Umgang mit Mitmenschen*, *Kooperation und Teamarbeit*

sowie eine *soziale Konfliktfähigkeit*. Der sozial-konforme Umgang mit Mitmenschen bedeutet die Fähigkeit, soziale Bindungen zu Mitmenschen – unabhängig von verschiedenen Berufen, Karrieren und anderen sozialen Hintergründen – zu knüpfen, diese aufrecht zu erhalten und mit selbigen umzugehen. Somit wird es Individuen ermöglicht, angemessen miteinander zu kommunizieren und eine soziale Empathie herauszubilden, um sich in den Mitmenschen und seine Perspektive auf einen Sachverhalt hineinzuversetzen.

- *Acting Autonomously*
  Diese Kategorie besteht aus Schlüsselkompetenzen, die das Individuum befähigen, in komplexen und verantwortungsvollen sozialen Kontexten zu agieren und soziale Muster und Systeme zu verstehen. Ferner werden Individuen befähigt, explizite und implizite Folgewirkungen des eigenen Handelns zu antizipieren. Folglich adressiert diese Gruppe von Schlüsselkompetenzen die Entscheidungsfähigkeit und die Fähigkeit zur Übernahme von Verantwortung eines Individuums. Zusammenfassend lässt sich feststellen, dass diese Art von Schlüsselkompetenzen entscheidend zur Entwicklung von Selbstständigkeit und Arbeitsfähigkeit eines Individuums ist.

Im Hinblick auf die Entwicklung von adäquaten Lernumgebungen, die z.B. im Rahmen des Informatikunterrichts der Sekundarstufe II an allgemeinbildenden Schulen zum Einsatz kommen, verdeutlicht der DeSeCo-Ansatz die Wichtigkeit, nicht nur fachspezifische Fähigkeiten und Fertigkeiten zu vermitteln, sondern die Förderung von ganzheitlichen Kompetenzen (also insbesondere auch nicht-kognitiven Kompetenzfacetten und Schlüsselkompetenzen) zu fokussieren.

4.2.3. Modellierungskompetenz und Schlüsselkompetenzen

Nachfolgend wird an zwei exemplarischen Kompetenzdimensionen des Rahmenmodells der Zusammenhang zwischen den im Modell abgebildeten Kompetenzfacetten und den oben beschriebenen Schlüsselkompetenzen illustriert. Hierbei soll im Sinne der theoretischen Absicherung des Kompetenzstrukturmodells gezeigt werden, wie die Vermittlung der im Modell enthaltenen Kompetenzen zur Entwicklung von Schlüsselkompetenzen beitragen kann.

K1 Aufgabenbereiche

Die Lerner sollen verschiedene Aufgabenbereiche informatischer Kompetenz erlernen. Diese können den Bereichen *Systemanwendung*, *Systemverständnis* und *Systemgestaltung* zugeordnet werden.

Wie zuvor erwähnt, umfasst die Komponente *Systemanwendung* Kompetenzfacetten zur Anwendung eines Informatiksystems. Sie ermöglichen dem Lerner, Technologie interaktiv im Sinne einer der oben beschriebenen Schlüsselkompetenzen zu nutzen. Hierzu gilt es seitens der Lerner ein Gespür herauszubilden, wie Technologie gewinnbringend in den persönlichen Alltag eingebracht werden kann. Hierbei muss ein Einschätzungsvermögen entwickelt werden, die Vorteile von Technologie abzuwägen, und selbiges entsprechend der persönlichen Umstände und Ziele sinnvoll einzusetzen, anstatt diese ohne Bedacht und Reflexion zu verwenden. Im Hinblick auf ein Informatiksystem kann dies bedeuten, jenes bewusst für die persönlichen Zwecke einzusetzen (z.B. als Lernmedium).
Die Kompetenzkomponente *Systemverständnis* fokussiert das Verständnis der Elemente eines Informatiksystems und der dahinterliegenden informatischen Prinzipien. Neben der bewussten Anwendung von Informatiksystemen ist die Entwicklung derartiger Kompetenzen (wie in der Kompetendimension abgebildet) eine wichtige Voraussetzung, um sich ein Informatiksystem sinnvoll zu Nutze zu machen und es interaktiv verwenden zu können. Gerade um das persönliche Potential eines Informatiksystems einschätzen zu können, ist es hilfreich deren Natur zu kennen.
Die für die Entwicklung von Informatiksystemen besonders relevante Komponente *Systemgestaltung* repräsentiert das Vermögen, Informatiksysteme zu entwickeln und reverse engineering zu betreiben. Wie bereits beschrieben, wurden die Unterkomponenten von *Systemgestaltung* anhand der statischen Phasen des *Rational Unified Process* [Rational Software Corporation IBM. 1998] theoretisch abgeleitet. Hier handelt es sich um ein mögliches Prozess-Framework, um viele bewährte Vorgehensweisen (*best practices*) im Softwareengineering in einem umfassenden Prozess-Framework zu kapseln. Dieser wird in vielen breit gestreuten Einsatzbereichen verwendet und lässt sich auf verschiedene Anwendungsszenarien adaptieren.
Die Unterkomponenten orientieren sich an den sog. *Process Workflows* des *RUP*. Diese beschreiben Abfolgen von diversen Aktivitäten innerhalb eines Software Engineering Prozesses und die Kommunikation der Personen, die die Modellierung betreiben.
Der *RUP* orientiert sich an einem iterativen Vorgehen bei der Entwicklung von Informatiksystemen und setzt auf komponentenbasierte Architekturen, die aus bereits bestehenden und neu entwickelten Komponenten bestehen.
Insbesondere im Bildungsbereich gilt es den *RUP* entsprechend der Komplexität von Software Engineering Projekten im schulischen Umfeld und der Kompetenz der Lerner anzupassen.
Wie im Verlauf des folgenden Unterkapitels beschrieben, ist die Fähigkeit ein Informatiksystem zu entwickeln (z.B. bei der Modellierung der Architektur des Systems) eng mit

verschiedenen nicht-kognitiven Kompetenzen verknüpft. Diese können wiederum verschiedenen Schlüsselkompetenzen entsprechend dem DeSeCo-Ansatz zugeordnet werden.

4.2.4. Nicht-kognitive Kompetenzen und Schlüsselkompetenzen

Die Kompetenzdimension fokussiert, wie zu Beginn des Kapitels beschrieben, die nicht kognitiven Kompetenzen eines Lerners, d.h. seine Einstellung gegenüber dem Informatiksystem, sozial-kommunikative und motivationale Kompetenzen.
Die Komponente *Einstellung gegenüber dem Gegenstandsbereich* umfasst die Art und Weise, wie ein Lerner ein Informatiksystem wahrnimmt inklusive eines prospektiven Blicks auf ein derartiges System. Ferner adressiert diese Komponente auch die Erwartungshaltung des Lerners gegenüber dem Umgang mit einem Informatiksystem. Der erneute Bezug auf das Konzept eines *sozio-technischen Informatiksystems* und die sich daraus ergebenden Implikationen im Umgang mit solchen Systemen, machen die Wichtigkeit deutlich, ein Informatiksystem im Kontext zu interpretieren und jenes während des Modellierungsprozesses zu antizipieren.
Die Begründung für die Integration der Komponente *Erwartungshaltung gegenüber einem IS* ergibt sich aus einer empirischen Studie von Magenheim und Schulte, welche die Wichtigkeit der Berücksichtigung der Erwartungshaltung der Lerner gegenüber dem Informatikunterricht betont. Somit lässt sich feststellen, dass die Erwartungshaltung bzw. Einstellung gegenüber dem Gegenstandsbereich als grundlegende Bedingung für ein erfolgreiches Lehren und Lernen zu verstehen ist [Magenheim und Schulte 2005]. Insbesondere mit der Zielsetzung, die interaktive Nutzung von Werkzeugen (z.B. bei der Integration von IT in den Alltag) zu fördern, erfordert dies Kenntnis bzgl. der Erwartungshaltung der Schüler gegenüber dem Gegenstandsbereich.
Wie zuvor erläutert, ist es wichtig, soziale Aspekte und Folgewirkungen in Betrachtung zu ziehen, wenn man sich mit sozio-technischen Informatiksystemen befasst. Diese Perspektive, die sich schon aus der Definition des Begriffs *Informatiksystem* ergibt, akzentuiert die Erfordernis von *sozial-kommunikativen Kompetenzen* wenn im Rahmen der Modellierung (z.B. im Rahmen der Phase Requirements) die unterschiedlichen Anforderungen und Interessen der Beteiligten (Auftraggeber, Benutzer und Entwickler) analysiert werden. Daraus lässt sich außerdem folgern, dass die Lerner die Fähigkeit entwickeln müssen, als aktiver Part in einem sozialen Handlungssystemen (z.B. in einer kooperativen Lernumgebung oder im Rahmen der kooperativen Softwareentwicklung) zu interagieren.
Die Lerner müssen weiterhin eine *Empathie* entwickeln, die es ihnen ermöglicht, die oben genannten Rollen innerhalb der Systemgestaltung und deren Perspektive auf das IS nachzuvollziehen. Nygaard postuliert, dass gerade jene höchst unterschiedlichen Perspektiven

auf ein IS durchaus kohärent und stimmig sein können [Nygaard 1986]. Dies zeigt die enorme Wichtigkeit eine solche Empathie herauszubilden. Ein weiterer Aspekt, der die Wichtigkeit von Empathie auch im schulischen Umfeld unterstreicht ist die Tatsache, dass Schüler der Sekundarstufe II häufig einen projektorientierten Informatikunterricht durchlaufen. Hierbei agieren sie in einer Art Rollenspiel innerhalb der relevanten Rollen des Software-Engineerings (Auftraggeber, Benutzer, Entwickler) und müssen auch den Perspektivwechsel der verschiedenen Rollen bei der Betrachtung von Informatiksystemen bewerkstelligen können. Durch diesen Anspruch der im Kompetenzmodell zu vermittelnden Kompetenzfacetten erhalten die Lerner zudem die Chance zu lernen in Teams zu arbeiten, zwischenmenschlich angemessenes Verhalten zu erlernen und Konflikte bei der kooperativen Arbeit lösen zu können.
Weiterhin wird dem Lerner die Möglichkeit geboten, Schlüsselkompetenzen zu erwerben, die sie dazu befähigen, autonom zu handeln. Hierbei kann der Prozess der Softwareentwicklung und Modellierung hilfreich sein, um die eigene Identität innerhalb des sozio-technischen Informatiksystems zu realisieren. Dementsprechend gilt es, eine Sensibilität im Hinblick auf die eigene Umwelt, Gruppendynamik und die eigene Rolle des SW-Engineering Prozesses zu entwickeln. Hierzu gehört auch die Fähigkeit, die sozialen Folgewirkungen des eigenen Handelns einschätzen zu können.
Neben der Einstellung und den sozial kommunikativen Kompetenzen gilt es auf Seiten des Lerners, motivationale Fähigkeiten herauszubilden (die eng mit dem persönlichen Prozess der Selbstregulation verknüpft sind), die es ihm ermöglichen, autonom zu handeln und komplexe Anforderungen nachhaltig zu bewältigen.
Die Lerner müssen auch dazu angehalten werden, eine willensmäßige Bereitschaft zu entwickeln, um eine Offenheit gegenüber neuen Ideen und Anforderungen zu haben und sich mit dem Gegenstandsbereich der informatischen Modellierung im Rahmen von SWE-Prozessen zu beschäftigen. Der Grad an Motivation spiegelt hierbei auch die Zweckmäßigkeit des eigenen Handelns, Entscheidungen zu treffen und Verantwortung zu übernehmen, wider. Folglich sind motivationale (und selbstregulative) Fähigkeiten von hoher Relevanz um unterschiedlichen Anforderungen gerecht zu werden und erfolgreich zu sein, da Personen mit hohen motivationalen Fähigkeiten nicht von unzweckmäßigen und im Sinne der Zielsetzung irrelevanten Gedanken und Kognitionen abgelenkt werden.
Zusammenfassend ermöglichen gerade jene zuvor beschriebenen nicht-kognitiven Kompetenzen dem Lerner, sein Wissen in komplexen Anforderungen des täglichen Lebens erfolgreich einsetzen zu können.

4.2.5. Zusammenfassung

Im Folgenden soll angelehnt an den vorherigen Abschnitt tabellarisch aufgeführt werden, inwiefern der Kompetenzerwerb in der Domäne des informatischen Modellierens im Rahmen der Systemgestaltung dazu beitragen kann Schlüsselkompetenzen zu fördern.

- *Systemgestaltung*
  - *Interaktion in heterogenen Gruppen*
    Kooperative SWE-Prozesse fördern die Kommunikation zwischen den beteiligten Personen. Die Lerner müssen selbstständig die Projektleiterrolle vergeben und diese sowie die weiteren durchaus heterogenen Rollen (Entwickler, Auftraggeber und Endnutzer) innerhalb des SW-Engineering Projekts bekleiden. Ferner müssen sie sich gemeinschaftlich (z.B. bei der Analyse von Klassendiagrammen) in die jeweiligen zugrundeliegenden informatischen Konzepte des zu entwickelnden Informatiksystems einarbeiten und herausfinden, wie diese implementiert werden können. Hierbei wird von den Lernern gefordert, dass die Gruppe trotz äußerst unterschiedlicher Rollen und Perspektiven auf ein Informatiksystem erfolgreich agiert und den gestellten Auftrag zielgerichtet absolvieren kann.
  - *Autonomes Handeln*
    Die Prozesse zur Entwicklung und zum Re-Engineering von Informatiksystemen umfassen die selbstständige Analyse von nicht bekannten Problemen und Anforderungen. Ferner gilt es im Rahmen des Re-Engineerings, das bestehende Informatiksystem um weitere Funktionen zu erweitern und dieses Wissen autonom auf ein neues, nicht bekanntes Informatiksystem zu transferieren. Diese Tätigkeiten erfordern ein hohes Maß an autonomen Handlungsvermögen seitens der Lerner.

Innerhalb dieses Kapitels wurde ein Einblick in die theoretische Ableitung von Kompetenzdimensionen und -komponenten für das Rahmenmodell aufgezeigt. Ferner wurden die abgeleiteten Kompetenzaspekte zur Modellierung auf deren Tauglichkeit zur Förderung von Schlüsselkompetenzen und zur Bildung von Allgemeinwissen geprüft. Somit wurde auch demonstriert, wie Schlüsselkompetenzen als zentraler Bereich der Allgemeinbildung in die Kompetenzdimensionen des Rahmenmodells integriert werden können.

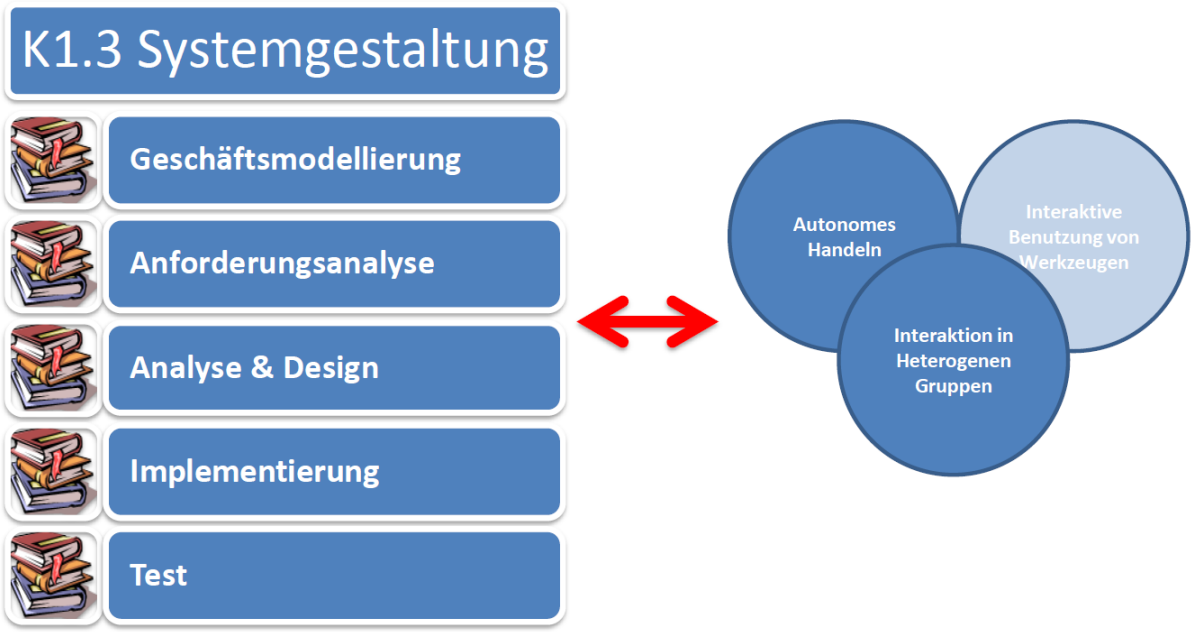


Abbildung 4.8.: Informatisches Modellieren und Schlüsselkompetenzen

Es bleibt zu klären, inwieweit das auf Grundlage des Kompetenzstrukturmodells entwickelte Instrument in der Lage ist, den Erwerb von Schlüsselkompetenzen zu messen. Dies ist allerdings nicht Gegenstand der vorliegenden Arbeit. Hier sollte das Kapitel vorwiegend zur weiteren theoretischen Fundierung der abgeleiteten Kompetenzdimensionen und -komponenten zur informatischen Modellierung dienen.

5. Empirische Entwicklung eines Kompetenzstrukturmodells für informatisches Modellieren

[1]Auf Grundlage des theoretisch abgeleiteten Rahmenmodells erfolgt die empirische Verfeinerung der jeweiligen Kompetenzdimensionen und -komponenten. In diesem Zusammenhang kommt ein Interviewverfahren zum Einsatz, das sich an bewährten aufgabenanalytischen Methoden und der *Critical Incident Technique* orientiert. Hierbei wird insbesondere ermittelt, wie die Befragten in kompetenzrelevanten Anwendungsszenarien problemlösend handeln und welche Kenntnisse, Strategien, Fähigkeiten und Einstellungen zu dessen effektiver Bewältigung erforderlich sind. Hiermit sollen die zuvor entwickelten Kompetenzdimensionen und -komponenten empirisch überprüft sowie konkretisiert und weiter ausdifferenziert werden.
Alle Interviews wurden audiotechnisch aufgezeichnet, transkribiert und unter Verwendung der qualitativen Inhaltsanalyse nach Mayring ausgewertet [Mayring 2010]. Hierbei kamen die drei Hauptfunktionen der qualitativen Inhaltsanalyse: *Zusammenfassung*, *Explikation* und *Strukturierung*, in kombinierter Form zur Anwendung, wobei Letztere die zentrale Rolle spielte. Die Kategorienbildung für die strukturierte Inhaltsanalyse erfolgte sowohl anhand der Sichtung der Expertenaussagen als auch anhand der vorab bestimmten Kompetenzkategorien und -facetten des theoretisch abgeleiteten Rahmenmodells mit Hilfe zusammenfassender und explikativer Arbeitsschritte.

[1]Das Kapitel 5 enthält die für die Modellierungskompetenz relevanten Anteile aus den eigenen Veröffentlichungen [Magenheim et al. 2010a] und [Magenheim et al. 2010b].

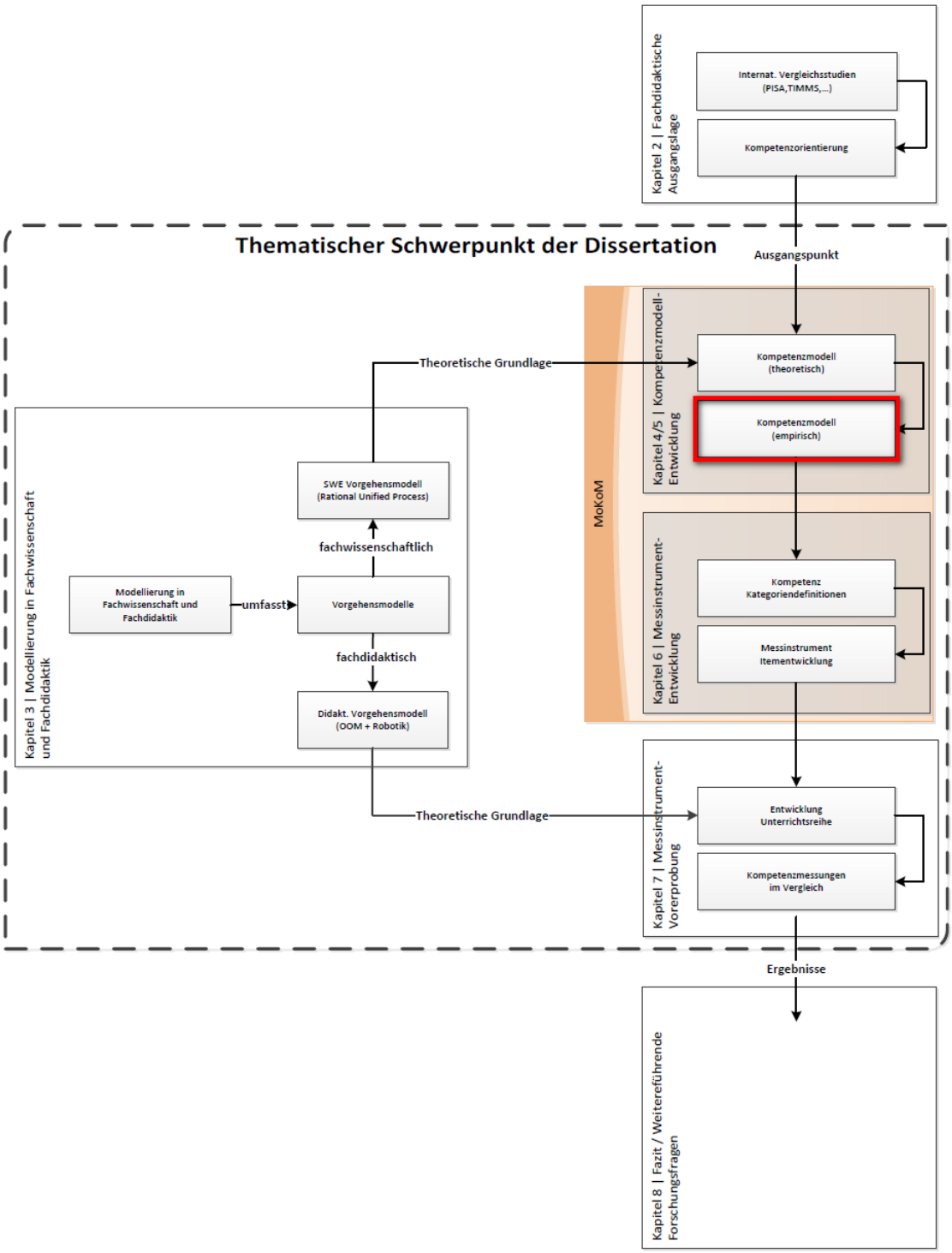


Abbildung 5.1.: Kapitel 5 im Gesamtkontext der Arbeit

Zur Vorbereitung der strukturierten Inhaltsanalyse wurden die transkribierten Interviewtexte außerdem hinsichtlich elementarer Sinneinheiten kodiert. Bei der eigentlichen strukturierten Inhaltsanalyse wurden die Sinneinheiten anhand der zuvor entwickelten Kategoriendefinitionen, Ankerbeispielen und Kodierregeln den Kompetenzkategorien zugeordnet. In Fällen, in denen eine Zuordnung durch vorhandene Ähnlichkeiten zweier Kategorien erschwert war, wurden Kodierregeln zur definitorischen Abgrenzung verwendet. Hierdurch erfolgte sowohl eine empirische Überprüfung des Kompetenzrahmenmodells hinsichtlich der normativ-deduktiv bestimmten Kompetenzdimensionen und -facetten als auch eine Korrektur, Ergänzung und Ausdifferenzierung des Modells hinsichtlich neuer und zusätzlich zu berücksichtigender Kompetenzaspekte [Lehner et al. 2010], [Schubert und Stechert 2010]. In einem weiteren Arbeitsschritt wurden die Ergebnisse der Inhaltsanalyse in ein ausformuliertes Kompetenzmodell transformiert. Hierbei wurden die Expertenaussagen weiter verallgemeinert und aggregiert sowie vor allem sprachlich in relevante und vereinheitlichte Kompetenzbeschreibungen transformatiert. Dabei wurde auf Auswertungsmethoden nach [Schaper und Horvath 2008] zurückgegriffen. Bei der Formulierung der Kompetenzbeschreibungen wurde unter Zuhilfenahme einer Operatorenliste [Informatik Zentralabitur NRW, 2005] unter anderem darauf geachtet, dass die Formulierungen kenntlich machen, ob die Kompetenzen aus den Anforderungsbereichen *(i) Kenntnisse und Wissen*, *(ii) Fähigkeiten und Können* oder *(iii) soziale und motivationale Bereitschaften* stammen. Die Formulierungen wurden darüber hinaus so gestaltet, dass jeweils eine Tätigkeitsausführung mit folgenden Elementen beschrieben wird (Subjekt „Die Lernenden...“, Prädikat „sind in der Lage, Anforderungen zu ermitteln.“, Objekt „...der zu entwickelnden Software“), die sich auf die fokussierten informatischen Domänen bezieht. Anhand zweier Beispiele in stark reduzierter Form werden innerhalb dieses Kapitels die Transformationen der inhaltsanalytischen Auswertungen in vereinheitlichte Kompetenzbeschreibungen dargestellt.

5.1. Rahmenbedingungen und empirische Grundlage bei der Durchführung der Interviews

In jedem Interview haben jeweils zwei Interviewer einen Interviewten befragt. Aus ökonomischen Gesichtspunkten wurden die Befragungen per Telefon oder VoIP (Skype) durchgeführt. Zur erfolgreichen Bewältigung des Interviews war keinerlei Vorbereitung seitens des Interviewten erforderlich. Die Interviews wurden standardisiert; deren Abfolge soll im Folgenden dargestellt werden:

1. Willkommensgespräch und Smalltalk um eine angenehme Gesprächsatmosphäre und einen gewissen Grad an Vertrauen des Interviewten zu gewinnen.
2. Vorstellung der Rahmenbedingungen und technischen Modalitäten des Interviews:
   - Wahrung der Anonymität
   - Nachfrage um Erlaubnis, das Interview audiotechnisch aufzeichnen zu dürfen um diese voll transkribieren zu können
   - Darlegung, dass die gestellten Fragen und Aufgaben nicht als Wissenstest zu verstehen sind sondern dazu dienen, bisher nicht bedachte Kompetenzfacetten innerhalb des Kompetenzstrukturmodells aufzufinden und bestehende Facetten zu verifizieren, konkretisieren oder auszudifferenzieren
3. Vorstellung des zugrunde liegenden Kompetenzverständnisses nach Weinert
4. Vorstellung der *Critical Incident Technique* als empirisches Vorgehen bei der Durchführung des Interviews
5. Information, dass innerhalb des Interviews vier hypothetische Szenarien vorgestellt werden und die Interviewten dazu aufgefordert sind, deren Vorgehen bei der Lösung der präsentierten informatischen Problem- und Aufgabenstellungen innerhalb der Szenarien genau zu beschreiben.
6. Hinweis, dass zum Ende der Befragung eine quantitative Evaluation der Typikalität der jeweiligen Szenarien im Hinblick auf den jeweiligen Kompetenzbereich erfolgt
7. Danksagung und Verabschiedung

Die Stichprobe besteht aus insgesamt 30 Informatik-Experten, die in drei gleichgroße Gruppen aufgeteilt sind: *Fachwissenschaftler*, *Fachdidaktiker* und *Fachleiter* des Unterrichtsfaches Informatik an der gymnasialen Oberstufe. Diese Zusammenstellung der Interviewten wurde gewählt, um ein möglichst breites Spektrum an Expertise im Bereich Didaktik der Informatik zu gewährleisten. Die Rekrutierung der Interviewkandidaten erfolgte mündlich, telefonisch oder per E-Mail.

5.1.1. Critical Incident Technique

Die *Critical Incident Technique* hat ihre Wurzeln in der Arbeits- und Organisationspsychologie. Sie wird vorrangig eingesetzt, um Anforderungen abzuleiten, die zur beruflichen Ausübung innerhalb eines bestimmten Tätigkeitsfelds erfolgsrelevant sind. In ihrer

ursprünglichen Form sind die Befragten dazu eingeladen, kritische (Anforderungs-) Situationen zu nennen und zu beschreiben, die innerhalb der jeweiligen Domäne relevant sind. Ferner werden die Befragten dazu aufgefordert, die persönliche Vorgehensweise zur Bewältigung dieser Situation genau darzulegen [Mayring 2010].
Innerhalb des Projekts *MoKoM* war es erforderlich, die methodische Vorgehensweise bei der Durchführung der Interviews zu modifizieren: Hier wurden die Interviewten nicht nach kritischen Anforderungssituationen befragt, sondern ihnen wurden lediglich hypothetische Anforderungsszenarien vorgestellt, bei denen sie ihr persönliches Vorgehen zur Bewältigung beschreiben sollten. Dieses methodische Vorgehen bei der Durchführung der Interviews und die Anpassung der *Critical Incident Technique* wurde gewählt, da zur Befragung bereits das (im vorherigen Kapitel beschriebene) theoretisch hergeleitete Kompetenzrahmenmodell vorlag. Dementsprechend wurden die hypothetischen Szenarien so gewählt, dass diese die verschiedenen Facetten informatischer Modellierungskompetenz entsprechend der Dimensionen und Komponenten im Rahmenmodell umfassen und adäquat repräsentieren.

Interviewszenarien für informatisches Modellieren

Insgesamt wurden 12 Szenarien entwickelt, wobei pro Interview jeweils vier Szenarien zum Einsatz kamen. Die Zusammenstellung wurde pro Interview so gewählt, dass Fragen zur Systemanwendung, zum Systemverständnis und zur Systemgestaltung berücksichtigt wurden.

1. Modellierung & Implementierung einer Software für ein Warenwirtschaftssystem
2. Modellierung & Implementierung eines verteilten Chatsystems
3. Modellierung & Implementierung eines web-basierten Spiels
4. Modellierung eines Klassendiagramms für eine Kontoführungssoftware
5. Implementierung eines Visualisierungsmoduls für Sortieralgorithmen
6. Software zur Verwaltung persönlicher Gegenstände auf Ergonomie prüfen
7. Systematische Erkundung der neuesten Version einer Standardsoftware
8. Umsatzübersicht über Werkzeugkategorien
9. Testen einer Softwareanwendung (Computer-Konfigurator)
10. Informationsbeschaffung über Online-Katalog und Suchmaschine
11. Datenbanken
12. Softwaretest im Team

Szenario 1: ***Modellierung & Implementierung einer Software für ein Warenwirtschaftssystem***	
Zusammenfassung	Hierbei ist der Experte gefragt, ein Warenwirtschaftssystem für einen Kiosk zu entwickeln. In diesem Zusammenhang gilt es, im Rahmen der Geschäftsmodellierung und Anforderungsanalyse, das Vorgehen zur Erfassung typischer Geschäftsvorgänge und funktionaler Anforderungen zu beschreiben. Ferner sollen die weiteren Phasen des Software-Engineering-Prozesses geplant werden und eine zeitgleiche Entwicklung der verschiedenen Module der Software in Kleingruppen koordiniert werden.
Szenariobeschreibung I	Sie erhalten den Softwareentwicklungsauftrag, ein Warenwirtschaftssystem für einen (Schul-)Kiosk zu entwickeln. Im Rahmen der Geschäftsmodellierung und Anforderungsanalyse sollen typische und alltägliche Geschäftsvorgänge erfasst werden.
Fragen I	- Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten? - Welche grafischen Beschreibungsmittel würden Sie dafür einsetzen? - Welche Kenntnisse und Fähigkeiten benötigen Sie zur Modellierung der Geschäftsprozesse und zur Anforderungsanalyse? - Welche informatischen Sichten sind hierbei von Bedeutung? - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf? - Welche (motivationalen) Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zur Modellierung der Geschäftsprozesse und zur Anforderungsanalyse? - Welche informatik-fremden Personen könnten (oder sollten) bei der Modellierung miteinbezogen werden? Welche Anforderungen kämen auf Sie zu, wenn Sie mit informatischen Laien über dieses SE-Projekt kommunizieren wollen? - Wie würde ein Schüler die Aufgabe angehen?
Szenariobeschreibung II	Sie erhalten den Auftrag, die weiteren Phasen des Software-Engineerings-Prozesses zu planen.
Fragen II	- Welche weiteren Phasen müssen Ihrer Meinung nach bis zur Verteilung des Software-Produkts durchlaufen werden? - Wie würden Sie hierbei vorgehen und was muss dabei beachtet werden? - Welche Kenntnisse und Fähigkeiten benötigen Sie in diesen Phasen des SE-Prozesses (insbesondere welche informatische Sichten)? - Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten sind in diesen SE-Phasen besonders relevant? - Welche Phasen würden Sie im Rahmen eines Schulprojekts: *Schulkiosk* im Informatikunterricht der Sekundarstufe durchlaufen wollen? - In welcher Form würden Sie informatik-fremde Personen auch in die weiteren SE-Phasen mit einbeziehen? Was wäre dabei zu beachten?

Szenariobeschreibung III	In der Implementierungsphase des Projekts sollen Kleingruppen gebildet werden, um die verschiedenen Module der Software zeitgleich zu entwickeln.
Fragen III	- Was muss bei der Einteilung von SE-Gruppen im professionellen Umfeld berücksichtigt werden? Welche Anforderungen ergeben sich an die Gruppenmitglieder? - Was muss bei der Gruppeneinteilung im Informatikunterricht beachtet werden? Welche sozialen und motivationalen Fähigkeiten und Einstellungen müssen seitens der Schüler vorhanden sein? - Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren?

Szenario 2: ***Modellierung & Implementierung eines verteilten Chatsystems***	
Zusammenfassung	Hier sollen im Rahmen eines Software-Engineering-Prozesses zur Entwicklung eines verteilten Chat-Systems in der Design-Phase die jeweiligen Programmmodule Client- und Server zugeordnet werden. Nach Abschluss der Analyse- und Designphase soll eine kooperative Implementationsphase in Kleingruppen geplant werden.
Szenariobeschreibung I	Sie erhalten den Auftrag ein Chat-System zu entwickeln. Im Rahmen der Designphase sollen Sie die potentiellen Programmmodule (Klassen) jeweils dem Client oder Server zuordnen.
Fragen I	- Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten? - Welche grafischen Beschreibungsmittel würden Sie dafür einsetzen? - Welche Kenntnisse und Fähigkeiten benötigen Sie zum Design des Client-Server-Systems? - Welche informatischen Sichten sind hierbei von Bedeutung? - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf? - Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zum Design des Client-Server-Systems? - Wie würde ein Schüler die Aufgabe angehen?
Szenariobeschreibung II	Nach Abschluss der Analyse- und Designphase soll eine zeitlich parallele Implementierung von Client- und Server-Softwarekomponenten geschehen. Sie als Projektleiter stehen nun vor der Aufgabe, die Aufgaben sinnvoll auf Teilgruppen ihres Teams zu verteilen.
Fragen II	- Wie würden Sie dabei vorgehen? - Was müsste in einem professionellen Umfeld bei der Gruppeneinteilung beachtet werden? - Wie würden Sie die Einteilung der Gruppen im schulischen Umfeld vornehmen um eine chancengleiche Kompetenzentwicklung zu ermöglichen? - Welche sozialen bzw. motivationalen Fähigkeiten der Schüler sollten zur erfolgreichen Implementierung vorhanden sein? Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren? - Durch welche kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv? - Welche arbeitsbezogenen sozialen Umstände könnten den Erfolg gefährden?

Szenario 3: ***Modellierung & Implementierung eines web-basierten Spiels***	
Zusammenfassung	Dieses Szenario thematisiert die Entwicklung eines webbasiertes Spiels. Der Fokus liegt hierbei zunächst auf der Entwicklung eines umfassenden Klassendiagramms anhand von gegebenen *CRC-Karten*. Der zweite Abschnitt des Szenarios sieht einen Robustheitstest, dessen Koordinierung und Vorgehensstrategie vor.
Szenariobeschreibung I	Im Rahmen eines Softwareprojekts soll das Web-basierte Spiel „Mensch ärgere Dich nicht“ implementiert werden. Sie haben bereits mit Hilfe von *CRC-Karten* Verantwortlichkeiten von Klassen herausgestellt und mögliche Zusammenhänge von Klassen lokalisiert. In einem weiteren Schritt soll nun ein umfassendes Klassendiagramm entwickelt werden.
Fragen I	- Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten? - Welche Kenntnisse und Fähigkeiten benötigen Sie zur Modellierung des Klassendiagramms? - Welche informatischen Sichten sind hierbei von Bedeutung? - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf? - Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zur Modellierung einer solchen, web-basierten Anwendung? - Beschreiben Sie die Unterschiede in der methodischen Vorgehensweise, die sich bei Anfängern, Fortgeschrittenen und Experten zeigen würden. - Wie würde ein Schüler die Aufgabe angehen?
Szenariobeschreibung II	In einem späteren Schritt (kurz vor Abschluss des Projekts) soll die Software im Rahmen der Testphase bzgl. Ihrer Robustheit überprüft werden. Hierbei soll sichergestellt werden, dass keinerlei unerwartete Benutzereingaben das Programm zum Absturz bringen.
Fragen II	- Wie würden Sie bei einem derartigen Test vorgehen, und was müssen Sie dabei beachten? - Wie würde ein Schüler die Aufgabe angehen?

Szenario 4: ***Modellierung eines Klassendiagramms für eine Kontoführungssoftware***	
Zusammenfassung	Hierbei galt es zunächst ein einfaches Klassendiagramm für eine rudimentäre Bankingsoftware zu modellieren. In einer weiteren Ausbaustufe bzw. Iteration wurden die Anforderungen an die Funktionalität der Software deutlich erweitert. Diese Anpassungen sollten auch in der Modellierung des erweiterten Systems mit berücksichtigt werden. Im Gegensatz zur ersten Iteration galt es zudem die Interaktion zwischen verschiedenen Klassen zu modellieren.
Szenariobeschreibung I	Sie erhalten im Rahmen der Entwicklung einer einfachen Kontoführungs-Software den Auftrag, ein Klassendiagramm zu entwickeln. Die Software soll zunächst einfache Ein- und Auszahlvorgänge auf **einem** Bankkonto realisieren.
Fragen I	- Wie gehen Sie dabei vor, und was müssen Sie dabei beachten? - Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareentwicklung? - Welche informatischen Sichten sind hierbei von Bedeutung? - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf? - Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareentwicklung? - Welche möglichen Probleme könnten bei Schülern der Sekundarstufe auftreten? - Welche grafischen Beschreibungsmittel würden Sie einsetzen und warum?
Szenariobeschreibung II	Für eine weitere Ausbaustufe der oben genannten Software soll nun ebenfalls ein Klassendiagramm erstellt werden. Im Gegensatz zu der ersten Ausbaustufe lassen sich nun beliebig viele Konten eröffnen. Neben Ein- und Auszahlungen auf das jeweilige Konto lassen sich nun auch Überweisungen zwischen den Konten vornehmen.
Fragen II	- Wie würden Sie dabei vorgehen? - Welche zusätzlichen Anforderungen ergeben sich durch den Übergang zur erweiterten Ausbaustufe der Kontoführungs-Software? - Rechtfertigen diese zusätzlichen Anforderungen eine Einteilung in Kleingruppen? - Durch welche kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv? - Welche arbeitsbezogenen sozialen Umstände könnten den Erfolg gefährden? - Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren? - Welche Anforderungen für die Schüler ergeben sich bei dieser komplexeren Version der Software? Wie würde ein Schüler die Aufgabe angehen?

Szenario 5: ***Implementierung eines Visualisierungsmoduls für Sortieralgorithmen***	
Zusammenfassung	Hierbei wird der Interviewte aufgefordert, ein Visualisierungsmodul für bereits implementierte Sortieralgorithmen zu entwickeln. In diesem Zusammenhang wurde der Fokus auf die Integration der jeweiligen Programmmodule (Sortier-Module und zu entwickelndes Visualisierungsmodul) gelegt.
Szenariobeschreibung	Sie haben im Informatikunterricht der Sekundarstufe II Sortieralgorithmen thematisiert und hierbei ausgewählte Sortierverfahren innerhalb von Programmmodulen implementiert. Zum Abschluss der Unterrichtsreihe soll nun ein Visualisierungsmodul implementiert werden. Dieses soll das zu sortierende Feld (Array) visualisieren und die jeweiligen Teilschritte während der Sortierung darstellen, indem sämtliche Änderungen im Feld grafisch hervorgehoben werden.
Fragen	- Wie würden Sie in diesem Zusammenhang vorgehen? - Was muss bei der Auswahl der Architektur, bei der Gestaltung der Schnittstellen und bei der Entwicklung der Benutzungsschnittstelle beachtet werden? - Welche Phasen ergeben sich bei der Entwicklung des Visualisierungsmoduls? - Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareimplementierung? - Welche informatischen Sichten sind hierbei von Bedeutung? - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf? - Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareimplementierung? - Wie würde Schüler an eine derartige Aufgabe herangehen? - Welche Phasen sehen Sie im Rahmen der schulischen Projektarbeit?

5.2. Empirisches Vorgehen zur Analyse Auswertung der Interviews

Die zuvor audiotechnisch aufgezeichnenten und voll transkribierten Interviews wurden auf Grundlage der Qualitativen Inhaltsanalyse nach Mayring analysiert und ausgewertet. Das Hauptziel dieser Methode ist es, die großen Mengen an transkribiertem Text zu bewältigen und für eine empirische Auswertung nutzbar zu machen.
Diesbezüglich lassen sich drei wichtige Techniken im Umgang mit großen Textmengen identifizieren [Mayring 2010]:

- *Zusammenfassung*

Als wichtige Funktion ist die Zusammenfassung und Reduktion von komplexen und umfassenden Textmaterialbeständen auf eine handhabbare Menge zu nennen. Hierbei sollten keine wichtigen Informationen verloren gehen.

- *Explikation*
  Mit Hilfe dieser Technik werden relevante beschreibende und interpretierende Textpassagen denjenigen hinzugefügt, die es zu interpretieren gilt bzw. die weiterer Erläuterung bedürfen.

- *Strukturierung*
  Bei Verwendung der dritten Technik soll eine Struktur aus dem Textmaterial abgeleitet werden. Hierzu wird versucht, das bestehende Textmaterial mit einer vorgegebenen Strukturierung in Form eines Kategoriensystems zu verknüpfen. Alle Textelemente, die sich den jeweiligen Kategorien dieses Systems zuordnen lassen, werden systematisch aus dem Interviewmaterial extrahiert.

Die oben aufgeführten Haupttechniken der qualitativen Inhaltsanalyse nach Mayring sind nicht zwingend unabhängig voneinander. Sie ergänzen sich gut, sodass eine kombinierte Verwendung dieser Techniken sinnvoll ist.
Im Rahmen der Analyse des Interviewmaterials im Projekt *MoKoM* kamen alle drei Techniken (auch in kombinierter Form) zum Einsatz. Hierbei repräsentiert das theoretisch abgeleitete Kompetenzmodell das Kategoriensystem, dem die jeweiligen relevanten Textpassagen zugeordnet wurden.
Insgesamt wurde im Rahmen des Projekts *MoKoM* ein interpretativer Ansatz bei der Auswertung des Interviewmaterials verfolgt: Hierbei wurden relevante Textpassagen, die Wissen, Fähigkeiten und Fertigkeiten sowie motivationale und sozial-kommunikative Fähigkeiten repräsentieren, den jeweiligen kognitiven und nicht-kognitiven Kompetenzfacetten des theoretisch hergeleiteten Kompetenz-Rahmenmodells zugeordnet.
Im nächsten Kapitel wird anhand von drei exemplarischen Interviewtranskripten das methodische Vorgehen bei der Auswertung der Interviews illustriert. Hierbei wurden in einem ersten Schritt Textelemente lokalisiert und markiert, die mögliche *Sinneinheiten* repräsentieren. *Sinneinheiten* sind als Textelemente zu verstehen, die relevante Informationen über informatik-spezifische Kompetenzen enthalten. Die Kompetenzen innerhalb einer Sinneinheit sind diejenigen, die kritisch (*critical*), also entscheidend zur Bewältigung von Problemen in der Domäne des informatischen Modellierens im Rahmen von SWE-Prozessen sind. Im nächsten Schritt wurden weitere Textelemente innerhalb des Interviewmaterials extrahiert. Jene repräsentieren sog. *Explikationen*, die mit den zuvor aufgefundenen *Sinneinheiten* verknüpft sind. Diese Textelemente stellen den Inhalt (also

die jeweilige Kompetenzfacette) der verknüpften Sinneinheit prägnant und anschaulich dar. Folglich können *Explikationen* den Inhalt von *Sinneinheiten* weiter präzisieren. Im nächsten Vorgehensschritt wurde untersucht, ob sich die aufgefundenen Sinneinheiten und Explikationen den Dimensionen und Komponenten des Rahmenmodells zuordnen lassen. Auf diese Weise lässt sich in einem ersten Schritt herausfinden, welche der aus den Transkripten extrahierten Sinneinheiten und Explikationen von den Dimensionen und Komponenten des Rahmenmodells adressiert werden und welche nicht. Darüber hinaus wird logischerweise auch deutlich, welche Sinneinheiten und Explikationen den Elementen des Rahmenmodells nicht zugeordnet werden können. In solchen Fällen gilt es, das bestehende Kompetenz-Rahmenmodell zu ergänzen.

5.3. Exemplarische Darstellung der Analyse der Experteninterviews

Innerhalb dieses Kapitels wird das Vorgehen zur Verfeinerung des theoretisch abgeleiteten Kompetenzmodells in Anlehnung an die qualitative Inhaltsanalyse nach Mayring beschrieben. Hierbei liegt der Fokus auf Kompetenzen zur informatischen Modellierung im Rahmen von SWE-Prozessen. In diesem Zusammenhang soll beispielhaft aufgezeigt werden, wie das Kompetenzmodell empirisch verfeinert werden kann, d.h. wie zusätzliche Komponenten anhand des Interviewmaterials abgeleitet werden können. Ferner soll im Sinne einer zeitlichen Momentaufnahme eine Ausbaustufe des empirisch verfeinerten Modells vorgestellt werden und erste, aus damaliger Sicht potentielle Verfeinerungen, vorgestellt werden. Im Hinblick auf die Kompetenzkomponente *Systemgestaltung* wurde - wie zuvor erläutert - ein Szenario mit Fragen entwickelt, welches genau jene Kompetenzaspekte dieser Domäne widerspiegelt. Im Folgenden wird ein Auszug des Szenarios *Chat System* mit den entsprechenden Fragen vorgestellt:

Szenario *Chat System*

Sie erhalten den Auftrag ein Chat-System zu entwickeln. Im Rahmen der Designphase sollen Sie die potentiellen Programmmodule (Klassen) jeweils dem Client oder Server zuordnen.

Frage 1: *„Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?“*

Frage 2: *„Welche grafischen Beschreibungsmittel würden Sie dafür einsetzen?“*

Frage 3: *„Welche Kenntnisse und Fähigkeiten benötigen Sie zum Design des Client-Server-Systems?“*

Frage 4: *„Wie würde ein Schüler die Aufgabe angehen?“*

Frage 5: *„Durch welche Einstellungen sowie kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv? Welche motivationalen Voraussetzungen spielen hierbei eine Rolle?“*

Frage 6: *„Wie würden Sie die Einteilung der Gruppen im schulischen Umfeld vornehmen um eine chancengleiche Kompetenzentwicklung zu ermöglichen?“*

Methodisch bezugnehmend auf Mayring haben wir das strukturelle Gerüst des theoretisch abgeleiteten Kompetenzmodells mit dem Interviewmaterial verknüpft, um die Struktur der Sinneinheiten innerhalb der Interviewtranskripte ableiten zu können. Die Zielsetzung der **Frage 1** war es herauszufinden, ob die vom Interviewten beschriebene Vorgehensweise und die damit ableitbaren Sinneinheiten die Komponenten (Workflows des Softwareengineerings) der Kompetenzdimension laden. Im weiteren Verlauf werden ausgewählte Antworten von Interviewten vorgestellt. Diese sind fortlaufend nummeriert. Die Antwort 1 ist darüber hinaus unterteilt in 1a bis 1c.

Antwort 1a: *„In einem ersten Schritt würde ich mir überlegen, was die Funktionalität des* Chatsystems *ausmacht. In diesem Zusammenhang müsste ich herausfinden, welche funktionalen Anforderungen bestehen.“*

Nachdem eine zusammenfassende Analyse der ersten Antwort durchgeführt wurde, konnte die Sinneinheit *Ableitung von funktionalen Anforderungen* aufgefunden werden. Im nächsten Schritt wurde untersucht, ob diese Sinneinheit einer Kompetenzkomponente des Rahmenmodells zugeordnet werden konnte. Falls dies nicht der Fall war, war diese Sinneinheit ein möglicher Hinweis für die Erweiterung der Kompetenzdimension um eine weitere Komponente. Weiterhin können derartige Sinneinheiten ein Indiz dafür sein, dass entsprechende Komponenten weiter auszudifferenzieren sind. In diesem konkreten Beispiel konnte die Sinneinheit *Ableitung von funktionalen Anforderungen* der Kompetenzkomponente *K1.2 Analysephase* zugeordnet werden. Folglich lädt jene Komponente die genannte Sinneinheit.

Antwort 1b: *„Danach würde ich mir überlegen, wie der Datentransfer von statten gehen könnte. Daher würde ich mir die folgenden Fragen stellen: Welches Protokoll ist hier sinnvoll? Wie kann die Verbindung zwischen den Endpunkten hergestellt werden. Wie wird über die Verbindung kommuniziert? Was passiert, wenn die Verbindung zusammenbricht? Was passiert, wenn die Verbindung vom Benutzer unterbrochen wird.“*

Um die Erläuterungen des Interviewten analysieren zu können, war es hilfreich eine Kombination der Techniken *Zusammenfassung* und *Explikation* zu verwenden. Zunächst konnte mit der zusammenfassenden Inhaltsanalyse der Textpassage die Sinneinheit *Entwurf der Kommunikation und des Datentransfers* aufgefunden werden. Um sicherzustellen, dass diese Sinneinheit der Kompetenzkomponente *Analyse & Design* zuzuordnen ist, wurde eine Analyse des jeweiligen Kontext der Sinneinheit vorgenommen. Hierbei war es hilfreich, sich auf den zweiten Satz der Antwort 1b zu beziehen: Hier hat der Interviewte geäußert, dass er sich über ein technisch angemessenes Protokoll klar werden müsse. Daraus geht hervor, dass seine Vorgehensweise dazu dient, die technischen Aspekte der Kommunikation mittels eines Protokolls zu klären. Diese Überlegungen könnten ferner als Grundlage für ein mögliches Entwurfs-Modell dienen, das eine Abstraktion des späteren Quellcodes darstellt. Die Erstellung eines derartigen Modells hat die Zielsetzung, das System – so wie es in der Implementierungsphase umgesetzt werden soll – darzustellen. Folglich kann *Entwurf der Kommunikation und des Datentransfers* der Komponente *Analyse & Design* zugeordnet werden und hat uns darüber hinaus auf die Idee gebracht, dass es sinnvoll sein könnte, die Komponente *Analyse & Design* in die Komponenten *Analyse* und *Design* zu untergliedern. Die Sinneinheit würde dann, falls sich diese Trennung nach Auswertung aller Interviews als sinnvoll erweist, logischerweise der Komponente *Design* zugeordnet.

Antwort 1c: *„In Bezug auf diese Vorüberlegungen kann eine Zuordnung der SW- Komponenten zu Client und Server erfolgen: Der Client implementiert sämtliche GUI-Klassen, das Event-Management und die Verbindung zum Server. Der Server hingegen, implementiert das Verbindungs-Management, die Nachrichtenverarbeitung und die Benutzerverwaltung.“*

Nach einer zusammenfassenden Analyse dieser Frage ließen sich die Sinneinheiten *Client umfasst die Funktionalität zur Interaktion mit dem Benutzer* und *Server umfasst Verbindungsmanagement, Nachrichtenverarbeitung und Benutzerverwaltung* ableiten. Mit „Vorüberlegungen“ spricht der Interviewte erneut seine Überlegungen hinsichtlich der Design-Phase an. Ausgehend von diesem Kontext können die aufgefundenen Sinneinheiten der Komponente *Analyse & Design* zugeordnet werden. Weiterhin könnten die Erläuterungen des Interviewten einen Hinweis darauf geben, dass Lernende in der Lage sein müssen, ausgehend von der jeweiligen Iteration der zu entwickelnden Software und des Kontextes ein zielführendes Vorgehen und angemessene Modellierungstechniken zu wählen. Ausgehend von dieser Annahme, wurde die Sinneinheit *Fähigkeit, ausgehend von der aktuellen Phase und Iteration des IS, ein sinnvolles Vorgehen auszuwählen* und *Fähigkeit, aus-*

gehend von der aktuellen Phase und Iteration des IS, eine dem Kontext angemessene Modellierungstechnik auszuwählen.
Die zweite Frage dieses Szenarios adressiert die Abfrage weiterer Informationen hinsichtlich verwendeter (graphischer) informatischer Modellierungstechniken und Beschreibungsmittel. Hierbei bestand die Zielsetzung mit Hilfe der strukturierenden Inhaltsanalyse die Komponente *K1.3 Systemgestaltung* mit für den Prozess der Systemgestaltung relevanten Modellierungstechniken weiter auszudifferenzieren und zu ergänzen.

Antwort 2: *„Wie zuvor erwähnt würde ich* UML-Klassendiagramme *und* UML - Deploymentdiagramme *verwenden.“*

Nach der Zusammenfassung dieser Textpassage erhielten wir *UML-Klassendiagramm* und *UML-Deploymentdiagramm* als Sinneinheiten. Da der Interviewte hinsichtlich des Klassendiagramms keinerlei Angaben dazu gemacht hat, ob es sich hierbei um ein Analyse- oder Entwurfs-Klassendiagramm handelt, spricht diese Aussage hingegen nicht eindeutig für eine weitere Differenzierung zwischen Analyse- und Designphase innerhalb des Kompetenzmodells. Unabhängig davon können diese Sinneinheiten jedoch zweifelsfrei der globaleren Kompetenzkomponente *Analyse & Design* zugeordnet werden.
Innerhalb der dritten Frage sollten konkrete Kompetenzfacetten für den Prozess der Modellierung im Rahmen der Systemgestaltung und zugehöriger relevanter Fähigkeiten und Fertigkeiten abgeleitet werden. Auch hierbei bestand die Zielsetzung in der Verfeinerung und Ergänzung der Komponenten der Dimension *K1 Basic Competencies.*

Antwort 3: *„Ich würde zur Lösung der Aufgabenstellung eine objektorientierte Vorgehensweise wählen. Hierzu muss ich bzw. müssen die Lernenden Fähigkeiten hinsichtlich der objektorientierten Analyse, des objektorientierten Entwurfs und der objektorientierten Programmierung erwerben. Ferner muss derjenige, der mit der Umsetzung der Software beauftragt ist, in der Lage sein, sich in eine spezifische objektorientierte Programmiersprache einzuarbeiten und mit dieser umzugehen. Darüber hinaus müssen sich die Entwickler im Klaren sein, dass sie es bei der zu entwickelnden Software mit einem verteilten System zu tun haben. Daher müssen sie sich zudem mit den jeweiligen Programmiertechniken von Client-/Server-Architekturen vertraut machen.“*

Durch eine zusammenfassende Inhaltsanalyse erhält man die Sinneinheiten *oo-Analyse*[2], *oo-Design*, *oo-Programmierung* und *Kenntnis und Umgang mit einer oo-Programmiersprache.* Diese können wiederum den Software Engineering Phasen der Kompetenzkomponente

[2] oo = objektorientiert

Systemgestaltung zugeordnet werden. In diesem Zusammenhang laden *oo-Analyse* und *oo-Design* die Unterkomponente *Analyse & Design* und *oo-Programmierung* und *Kenntnis und Umgang mit einer oo-Programmiersprache* die entsprechende Unterkomponente *Implementierung*. Hierbei bleibt anzumerken, dass getrennte Nennung von Analyse und Design für eine Ausdifferenzierung der Komponenten sprechen könnte.
Im weiteren Verlauf wurden die Interviewten gefragt, wie sie die Vorgehensweise von Schülern der Sekundarstufe zur Lösung der im Szenario präsentierten Anforderungssituation einschätzen würden. Hierdurch sollte untersucht werden, welche Kompetenzaspekte für den Informatikunterricht in der Sekundarstufe II relevant sein könnten. Weiterhin bestand das Ziel, die Unterschiede hinsichtlich der Vorgehensweise zur Lösung dieses Problems zwischen Experten und Novizen in Erfahrung zu bringen.
Mit dem oben beschriebenen Anspruch, sowohl kognitive als auch nicht kognitive Kompetenzen für die Modellierung innerhalb von SWE-Prozessen abzuleiten, hat die Frage 5 den Fokus auf nicht kognitiven Kompetenzen. Um diesbezüglich eine strukturierende Inhaltsanalyse der Antwort 5 vorzunehmen, gilt es einen weiteren Blick in die entsprechende Kompetenzdimension *K4 Nicht kognitive Kompetenzen* zu werfen. Diese umfasst die Kompetenzkomponenten *Einstellung gegenüber einem IS*, *Sozial-kommunikative Kompetenzen* sowie *motivationale Kompetenzen*. Im Folgenden soll exemplarisch gezeigt werden, wie die Frage 5 zur Verfeinerung der Kompetenzdimension K4 beitragen könnte.

Antwort 5: *„Zunächst müssen die Lernenden die Bereitschaft und den Willen dazu haben, sich in neue und für sie unbekannte Themengebiete einzuarbeiten. Eng damit verbunden ist auch die Einstellung, dass die verwendeten Technologien sinnvoll sind. Ferner muss den Lernenden im Rahmen einer Aufgabenstellung ein möglichst naher Bezug zur eigenen Lebenswelt vermittelt werden. Der Kontext steht im Vordergrund und nicht die Lösung von einzelnen isolierten Problemen. Im Hinblick auf soziale und kommunikative Fähigkeiten müssen die Lernenden befähigt werden, kooperativ zu arbeiten. Hierzu müssen sie in der Lage sein, Ideen anderer aufzugreifen und diese auf kreative Weise weiterzuentwickeln.“*

Zur Auswertung der Frage 5 wurde wiederum eine Kombination aus strukturierender und zusammenfassender Inhaltsanalyse durchgeführt. Hierdurch konnten die folgenden Sinneinheiten aufgefunden werden und Unterkomponenten von *K4 Nicht kognitive Kompetenzen* zugeordnet werden: *Bereitschaft Wissenslücken selbstständig zu schließen*, *die Einstellung haben, dass die verwendete Technologie sinnvoll ist*, *Einstellung, im Kontext mit Lebensweltbezug zu arbeiten*, *Fähigkeit, die Ideen anderer aufzugreifen und zu*

verwenden und *Fähigkeit, die Ideen anderer kreativ und konstruktiv weiterzuentwickeln.* Die Sinneinheiten *die Einstellung haben, dass die verwendete Technologie sinnvoll ist* und *Einstellung, im Kontext mit Lebensweltbezug zu arbeiten* laden die Kompetenzkomponente *Einstellung gegenüber einem IS.* Schlussendlich können die Sinneinheiten *Fähigkeit, die Ideen anderer aufzugreifen und zu verwenden* und *Fähigkeit, die Ideen anderer kreativ und konstruktiv weiterentwickeln* zweifellos der Komponente *Sozial-kommunikative Fähigkeiten* zugeordnet werden.
Die letzte Frage (Frage 6) hatte die Intention herauszufinden, welche nicht kognitiven Fähigkeiten kritisch für die Bewältigung von Aufgaben in der Domäne des informatischen Modellierens in SWE-Prozessen im Rahmen der Sekundarstufe II sind.

5.4. Ergebnisse der Auswertung der Experteninterviews

Insgesamt konnten innerhalb der oben dargestellten exemplarischen qualitativen Inhaltsanalyse verschiedenen Sinneinheiten abgeleitet werden. Diese konnten uns erste Hinweise zur Validierung, zur Verfeinerung und zur Ergänzung der Kompetenzaspekte im Rahmenmodell liefern:

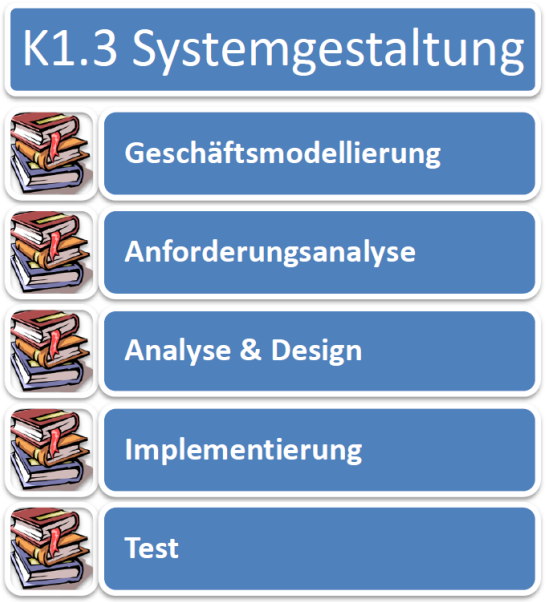


Abbildung 5.2.: Theoretisches Teilmodell *Modellierung*

Diejenigen Sinneinheiten, die den Kompetenzkomponenten des Modells zugeordnet werden konnten, zeigen auf, welche Bereiche des theoretisch abgeleiteten Rahmenmodells

sich als sinnvoll erwiesen haben (grün markierte Komponenten in Abbildung 5.5). Hier sind die Kompetenzkomponenten *Systemanwendung*, *Systemgestaltung* und die jeweiligen Unterkomponenten *Anforderungsanalyse*, *Analyse*, *Design* und *Implementierung* zu nennen.
Außerdem konnten weitere Sinneinheiten Aufschluss geben, wie das Kompetenzmodell im Anschluss an die Auswertung sämtlicher Experteninterviews empirisch verfeinert werden könnte. In diesem Zusammenhang könnte eine Ergänzung von weiteren Kompetenzkomponenten oder eine Verfeinerung bestehender Komponenten vorgenommen werden. Konkret könnte es daraus folgend sinnvoll sein, die Komponente *Analyse & Design* in *Analyse* und *Design* aufzugliedern. Obwohl diese Phasen voneinander abhängig und teilweise eng miteinander verknüpft sind, gab es Sinneinheiten (z.B. bei der Zuordnung von Klassen zu Client oder Server) die eindeutig der Design-Phase und nicht der Analyse-Phase zugeordnet werden konnten (und umgekehrt). Eine weitere Verfeinerung, die sich aus der Analyse dieser exemplarischen Interviewtranskripte ergeben hat, ist die Erweiterung der Kompetenzdimension *K1 Aufgabenbereiche* um die Komponente *Fähigkeit, zur Auswahl eines geeigneten Vorgehensmodell sowie zur Auswahl von relevanten Modellierungstechniken in Abhängigkeit zur jeweiligen Iteration der zu entwickelnden SW* (Abk.: *Iteratives Vorgehen*). Insbesondere die Tatsache, dass der Prozess der Modellierung und Systemgestaltung gerade bei komplexen Systemen selten linear abläuft, scheint die Förderung derartiger Kompetenzen zur kontextsensitiven Auswahl von Vorgehen und Modellierungstechnik sinnvoll.
Nachdem die Zuordnung sämtlicher hier aufgefundener Sinneinheiten erfolgt ist, müssen die verbleibenden Komponenten unter Bezugnahme der restlichen Interviewtranskripte legitimiert werden. Falls hierbei nach vollständiger Interviewauswertung keine Zuordnung von Sinneinheiten zu bestimmten Kompetenzkomponenten des Rahmenmodells möglich ist, könnte dies ein Indiz dafür sein, dass diese Komponenten ggf. in einer späteren Version des Modells keine Berücksichtigung mehr finden.
Die folgenden Abbildungen illustrieren den Prozess der Verfeinerung der theoretisch hergeleiteten Kompetenzdimension. Zum besseren Verständnis werden zunächst die verwendeten ikonischen Abbildungen in einer Legende (Abbildung 5.3) aufgeführt.

Symbol	Bedeutung
	Theoretisch hergeleitete Komponente
	Empirisch bestätigte Komponente
	Empirisch hinzugefügte Komponente
	Empirisch ausdifferenzierte Komponente
	Empirisch verworfene Komponente

Abbildung 5.3.: Legende zu den folgenden Abbildungen

5.5. Ergebnisse der qualitativen Inhaltsanalyse

5.5.1. Exemplarische Darstellung der Auswertung mit Zuordnung zu den Komponenten des Rahmenmodells

Wie im vorherigen Kapitel dargelegt, zeichnet sich ein typisches Szenario für den Kompetenzbereich des informatischen Modellierens dadurch aus, dass der Interviewte eine bestimmte Phase eines SWE-Prozesses durchläuft. Hierbei wurden die Antworten bzw. die daraus abgeleiteten Sinneinheiten im Sinne einer strukturierenden Inhaltsanalyse den einzelnen Dimensionen und Komponenten des theoretisch abgeleiteten Rahmenmodells zugeordnet. In Anbetracht der Gesamtauswertung aller 30 Experteninterviews wurde die Kompetenzdimension *K1 Aufgabenbereiche* und insbesondere die Komponente *K1.3 Systemgestaltung* folgendermaßen empirisch verfeinert:
Insgesamt 25 Interviewte erwähnten, dass der Prozess der Systemgestaltung einen wichtigen Bereich informatischer Kompetenz darstellt.

Antwort K1.3: *„Die verschiedenen Phasen des Wasserfallmodells müssen durchlaufen werden.“*

Antwort K1.3: *„Wir müssen die Phasen eines professionellen SWE-Prozesses durchlaufen.“*

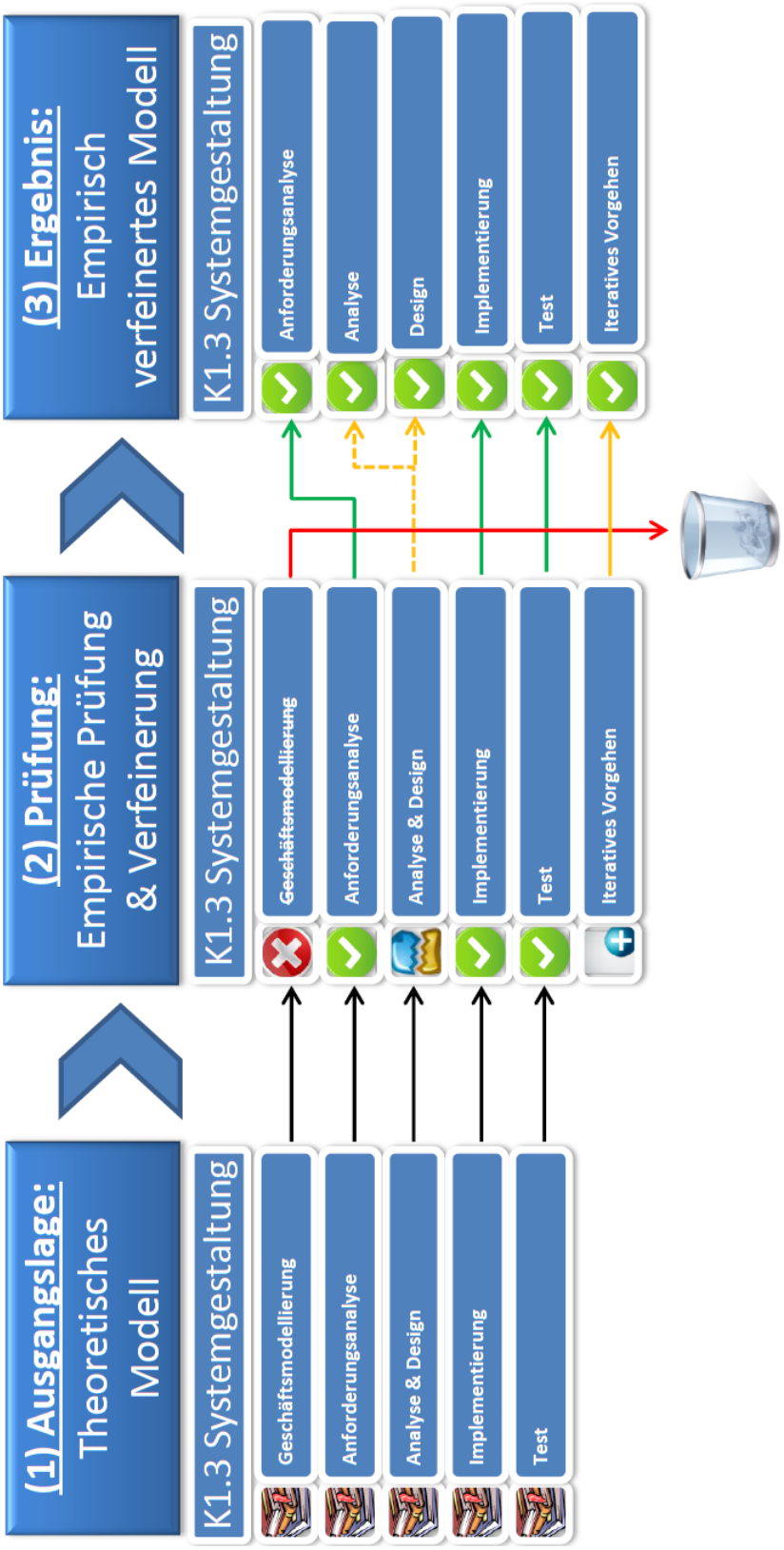


Abbildung 5.4.: Prozess zur empirischen Verfeinerung des Teilmodells *Modellierung*

Abbildung 5.5.: Empirische Verfeinerung des Teilmodells *Modellierung*

Die folgenden exemplarischen Aussagen konnten den Unterkomponenten von *K1.3 Systemgestaltung* K1.3.1 - K1.3.5 zugeordnet werden:
Hierbei erwähnten 12 Experten *K1.3.2 Anforderungsanalyse* und erläuterten, dass Lerner in der Lage sein müssen, Use Cases zu analysieren und funktionale Anforderungen zu spezifizieren.

Antwort K1.3.2: *„Es sollten verschiedene Use Cases analysiert werden, um herauszufinden, wie das spätere System genutzt wird."*

Antwort K1.3.2: *„Spezifikation von funktionalen Anforderungen."*

Die Komponente *K1.3.3 Analyse & Design* wurde von einer großen Anzahl Interviewten genannt. Hierbei wurde erklärt, dass dieser SWE-Prozess in mehreren Iterationen durchlaufen werden muss. Hierbei hat es sich als sinnvoll erwiesen die Komponenten *K1.3.3 Analyse & Design* in *Analyse* und *Design* aufzuteilen. Somit konnte eine präzisere Zuordnung von Sinneinheiten vorgenommen werden. Die weiteren aufgefundenen Aussagen adressieren die Analysephase.

Antwort K1.3.3 (Analyse): *„Die Analyse-Phase muss durchlaufen werden."*

Antwort K1.3.3 (Analyse): *„Durchführung einer objektorientierten Analyse.“*

Antwort K1.3.3 (Analyse): *„Im Rahmen der Analysephase müssen sich die Lerner den Problembereich vergegenwärtigen.“*

Hinsichtlich der Designphase erläuterten die Interviewten, dass die Lerner befähigt sein müssen (kontextabhängig) sinnvolle Konzepte auszuwählen und eigene Design-Konzepte eines IS entwickeln zu können.

Antwort K1.3.3 (Design): *„Man muss sich mit verschiedenen Design-Techniken vertraut machen.“*

Antwort K1.3.3 (Design): *„Die Lernenden müssen befähigt werden, adäquate Design-Techniken auszuwählen um den Entwurf der zu entwickelnden SW voranzutreiben.“*

Antwort K1.3.3 (Design): *„Ich muss in der Lage sein, Programm-Module passend zu den jeweiligen Aufgaben abzuleiten.“*

Antwort K1.3.3 (Design): *„Im Anschluss muss den Lernenden vermittelt werden, wie sie ein eigenes Konzept für das Design eines SW-Systems entwickeln.“*

Unabhängig von dieser möglichen Unterscheidung zwischen *Analyse* und *Design* zählten die Interviewten diverse Modellierungstechniken auf. Dies zeigt, dass die Lernenden in der Lage sein müssen, sinnvolle Modellierungstechniken in Abhängigkeit von der aktuellen SWE-Phase und dessen aktueller Iteration auszuwählen.

Antwort K1.3.3 (Analyse & Design): *„Anwendung von UML-Diagrammen.“*

Antwort K1.3.3 (Analyse & Design): *„Entwickeln von Klassendiagrammen“*

Antwort K1.3.3 (Analyse & Design): *„Identifizierung von potentiellen Klassen mit dem Ziel ein Klassendiagramm zu entwickeln“*

Antwort K1.3.3 (Analyse & Design): *„Identifizierung und Zuordnung von inhaltlichen Bestandteilen eines Klassendiagramms, z.B. Attribute, Methoden, Assoziationen und Vererbung.“*

Antwort K1.3.3 (Analyse & Design): *„Anwendung von Zustandsdiagrammen“*

Antwort K1.3.3 (Analyse & Design): *„Verwendung von UML-Deployment-Diagrammen“*

Die Komponente *K1.3.4 Implementierung* wurde sechs mal genannt. In diesem Zusammenhang betonten die Interviewten, dass die Lerner in der Lage sein müssen, Algorithmen zu verstehen und diese nachvollziehen zu können und sich in eine objektorientierte Programmiersprache einarbeiten müssen.

Antwort K1.3.4 (Implementierung): *„Verstehen von Algorithmen (insbesondere rekursive Algorithmen).“*

Antwort K1.3.4 (Implementierung): *„Wir müssen lernen in einer bestimmten objektorientierten Programmiersprache zu implementieren und mit den Syntaxfehlern umzugehen.“*

Antwort K1.3.4 (Implementierung): *„Sich in die objektorientierte Programmierung einarbeiten.“*

Antwort K1.3.4 (Implementierung): *„Den Unterschied zwischen Klassen- und Objekt-Attributen und -Methoden verstehen.“*

Darüber hinaus erwähnten einige Interviewte, dass die Lernenden in der Lage sein müssen, SW-Module (die beispielsweise in Teamarbeit implementiert wurden) in ein umfassendes SW-System zu integrieren. Diese Äußerungen (Sinneinheiten) können wiederum der Komponente *K1.3.4 Implementierung* zugeordnet werden. Ferner können jene allerdings auch ein Hinweis darauf sein, dass es sinnvoll sein könnte, das Rahmenmodell um eine neue Komponente z.B. *K1.3.4a Integration von SW-Modulen* zu ergänzen. Ferner kann die Sinneinheit *Implementierung in Teams* auch der Dimension *K4 Nicht-kognitive Kompetenzen* und der Komponente *K4.2 Sozial-kommunikative Fähigkeiten* zugeordnet werden. Dies zeigt wiederum, dass die Dimensionen des Kompetenzmodells eng miteinander verknüpft sind und nicht isoliert voneinander betrachtet werden sollten.

Antwort K1.3.4 (Implementierung): *„Synchronisation von Programmodulen, die von verschiedenen Teams entwickelt wurden.“*

Antwort K1.3.4 (Implementierung): *„Integration von Programmmodulen, die von verschiedenen Teams implementiert wurden.“*

Darüber hinaus kam in diesem Zusammenhang seitens der Interviewten zum Ausdruck, dass die Lerner mit Informatiksystemen eines unterschiedlichen Komplexitätsgrades konfrontiert sind und mit diesen umgehen lernen müssen. Dementsprechend könnten die folgenden Sinneinheiten ebenso für eine Zuordnung zur Dimension *K3 Komplexität* sprechen.

Antwort K1.3.4 (Implementierung): *„Die Lernenden müssen Software „from the scratch“ entwickeln lernen.“*

Antwort K1.3.4 (Implementierung): *„Die Lernenden müssen befähigt werden, sich in bestehende Informatiksysteme einzuarbeiten und re-engineering zu betreiben.“*

Antwort K1.3.4 (Implementierung): *„Analyse von Schnittstellen eines bereits bestehenden Systems“*

Die Relevanz der Komponente *K1.3.5 Test* wurde von 8 Interviewten bestätigt.

Antwort K1.3.5 (Test): *„Testen der Produktqualität.“*

Antwort K1.3.5 (Test): *„Durchführung der Testphase um u.a. sicherzustellen, dass die Anforderungen aus der Requirements-Phase (Anforderungsanalyse) korrekt umgesetzt wurden.“*

Weiterhin nannten die Experten bereits bewährte Vorgehensstrategien im Rahmen der Testphase.

Antwort K1.3.5 (Test): *„Durchführung von Blackbox-Tests.“*

Antwort K1.3.5 (Test): *„Whitebox-Tests als mögliche Vorgehensstrategie in der Testphase.“*

Antwort K1.3.5 (Test): *„Durchführung von Regressionstests.“*

Zusammenfassend illustriert dieses Kapitel verschiedene Sinneinheiten, die von den Antworten der Interviewten im Rahmen der Expertenbefragung abgeleitet wurden. Diese wurden im Hinblick auf die Unterkomponenten von *K1.3 Systemgestaltung* strukturiert. Ausgehend von dieser Zuordnung wurden Verfeinerungen des Rahmenmodells vorgenommen. Im Folgenden Unterkapitel wird dementsprechend aufgezeigt, welche Komponenten sich als zutreffend erwiesen haben, welche Komponenten weiter ausdifferenziert werden sollten und welche neuen Komponenten dem Modell ergänzt werden sollten. Weiterhin wird dargestellt, welche Komponenten ggf. aufgrund fehlender Bestätigung durch die Experteninterviews aus dem Modell entfernt werden sollten.

5.5.2. Empirische Verfeinerung des Rahmenmodells

In diesem Unterkapitel sollen die Auswirkungen der qualitativen Inhaltsanalyse der Experteninterviews auf das Kompetenzstrukturmodell aufgeführt werden.

Abbildung 5.6.: Empirisch Verfeinertes Kompetenzstrukturmodell

Die Komponente *K1.3 Systemgestaltung* hat sich als wichtiger Part des Kompetenzmodells bestätigt, da diese grundlegende Prozesse und Kompetenzen zur Entwicklung von Informatiksystemen umfasst. Dennoch hat die Auswertung der Interviews auch gezeigt, dass Verfeinerungen am Rahmenmodell vorgenommen werden müssen.

Abbildung 5.7.: Empirisches Teilmodell *Modellierung*

In diesem Zusammenhang hat kein Interviewter *K1.3.1 Geschäftsmodellierung* genannt. Ferner konnten keine Sinneinheiten aufgefunden werden, die sich dieser Komponente eindeutig haben zuordnen lassen. Infolgedessen gilt es, die Daseinsberechtigung dieser Komponente weiter zu verifizieren. Im Rahmen dieser Arbeit wurde selbige als nicht relevant für das verfeinerte Kompetenzmodell deklariert, insbesondere im Hinblick auf den späteren Einsatz der daraus resultierenden Kompetenzmessinstrumente im Informatikunterricht der Sekundarstufe II an allgemeinbildenden Schulen.
Die Kompetenzkomponente *K1.3.2 Anforderungsanalyse* wurde vielfach genannt und konnte diversen Sinneinheiten zugeordnet werden. Infolgedessen hat sich diese Komponente als sinnvoll erwiesen.
K1.3.3 Analyse & Design wurde von zahlreichen Interviewten erwähnt. Dennoch konnten viele Sinneinheiten eindeutig der Analysephase oder der Designphase zugeordnet werden. Folglich scheint es sinnvoll, eine Aufgliederung der Komponente in zwei separate Kompetenzkomponenten vorzunehmen um eine präzisere Kategorisierung von Modellierungskompetenzen vornehmen zu können. Wie zuvor erwähnt sind diese Phasen eng

miteinander verknüpft und werden häufig in mehreren Iterationen durchlaufen [Rational Software Corporation IBM. 1998, S. 2]. Unabhängig davon hat die Analyse der Experteninterviews gezeigt, dass Lerner dahingehend gefördert werden müssen, damit sie der aktuellen Iteration und Phase des SWE-Prozesses entsprechend sinnvolle Vorgehensweisen und Modellierungstechniken auswählen können. Die Förderung derartiger Kompetenzen umfasst die Neustrukturierung von Wissen und kann als wichtige Voraussetzung für situiertes Lernen gesehen werden. In diesem Kontext wird auch der Zusammenhang der Kompetenzdimension *K1 Aufgabenbereiche* und *K2 Informatische Sichten* deutlich und soll im Folgenden anhand verschiedener Beispiele illustriert werden:
Zur Förderung der oben genannten Auswahl und Anwendung relevanter Modellierungstechniken ist die Fähigkeit zum *K2.3 Perspektivwechsel* unabdingbar.
Die situierte Auswahl von angemessenen und zielführenden Modellierungstechniken umfasst zudem die Anwendung diverser UML-Diagramme und kann sowohl der Komponente *K1.3.3 Analyse & Design* als auch der Komponente *K2.2.6 grafische Beschreibungsmittel* zugeordnet werden. Dies zeigt wiederum die inhaltliche Verknüpfung der einzelnen Kompetenzdimensionen.
Wie im vorherigen Kapitel erläutert, ist die Phase der Modellierung ein essentieller Bereich innerhalb des SWE-Prozesses. Die in diesem Zusammenhang aufgefundenen Sinneinheiten demonstrieren erneut die enge Verknüpfung von *K1 Aufgabenbereiche* und *K2 Informatische Sichten*. Die Sinneinheit *Verstehen von Algorithmen* könnte der Kompetenzkomponente *K1.3.5 Implementierung* oder der Komponente *K2.2.4 Algorithmen & Datenstrukturen* zugeordnet werden. Ein weiteres Ergebnis ausgehend von der Analyse der Experteninterviews ist der Umgang mit Informatiksystemen in unterschiedlichen Fertigstellungsgraden. Dies könnte ein Hinweis sein, dass die Kompetenzdimension *K3 Umgang mit Komplexität* und die Komponente *Grad der Fertigstellung des Informatiksystems* erweitert werden sollten.
Neben dem Grad der Fertigstellung eines Informatiksystems erwähnten einige Interviewte ausdrücklich, dass die Lernenden in der Lage sein müssten, Programmmodule (in Teams entwickelt) zu synchronisieren und diese in ein umfassendes Informatiksystem zu integrieren. Hierbei gilt es beispielsweise, gemeinsame Schnittstellen innerhalb der Arbeitsgruppen abzustimmen und erfordert *K4.2 Sozial-kommunikative Fähigkeiten*. Dies zeigt, dass die Kompetenzdimension *K1 Aufgabenbereiche* eng mit der Kompetenzdimension *K4 Aufgabenbereiche* verknüpft ist.
Die Kompetenzkomponente *K1.3.5 Test* wurde von vielen Interviewten erwähnt und hat sich als essentieller Bestandteil der Kompetenzdimension erwiesen.

5.5.3. Fallstudie Charakteristika der Interviewten

[3]Im Rahmen der Auswertung der Experteninterviews haben wir uns parallel mit der Fragestellung beschäftigt, inwieweit sich die Gruppen der interviewten Fachleiter, Fachdidaktiker und Fachwissenschaftler unterscheiden und welche Charakteristika auszumachen sind. Hierbei wurde eine gezielte inhaltliche Analyse anhand des Fallbeispiels (Szenario: Warenwirtschaftssystem) vorgenommen.
Für den Kompetenzbereich *Informatisches Modellieren* wurden, wie bereits beschrieben, fünf Szenarien verwendet um das zuvor theoretisch abgeleitete Rahmenmodell zu verfeinern.
Ausgehend von den Interviewantworten soll ein Vorschlag für die mögliche empirische Verfeinerung des Rahmenmodells vorgestellt werden und die möglichen Lösungsstrategien der Szenarien der einzelnen Interviewgruppen ermittelt werden.
Zusammenfassend lässt sich feststellen, dass die Interviewten die zentrale Bedeutung der Kompetenzkategorie *K1.3 Systemgestaltung* als wichtigen Bestandteil des Kompetenzmodells bestätigt haben. Unabhängig davon konnten unterschiedliche Vorgehensweisen bei den Vorgehensstrategien der einzelnen Gruppen festgestellt werden.
Ein Proband aus der Gruppe der Fachwissenschaftler gibt vor, einen SWE-Prozess zu wählen, der verschiedene Iterationen durchläuft. Dies bestätigt uns die Kompetenzkomponente *K1.3 Systemgestaltung*, die von der sog. *core workflows* des *Rational Unified Process* abgeleitet wurde. Dieser sieht ebenfalls einen iterativen Prozessverlauf vor. Deshalb erschien es uns als sinnvoll, eine weitere Komponente *K1.3 Sequencing Pattern* dem Modell hinzuzufügen. Diese umfasst diejenigen Kompetenzfacetten, die es dem Lernenden ermöglichen, zweckmäßige Problemlösungsstrategien und -techniken (hier Modellierungstechniken) in Abhängigkeit zur aktuellen Phase und Iteration des SWE-Prozesses zu wählen.
Im Gegensatz zu dem oben beschriebenen Vorgehen bevorzugt ein Proband aus der Gruppe der Befragten Fachleiter den klassischen linearen Prozessaufbau und erwähnt in diesem Zusammenhang das *Wasserfallmodell*. Dies bestätigt uns wiederum die Relevanz der Komponenete *K1.3 Systemgestaltung*, da die Phasen des Wasserfallmodells ähnlich der core workflows des *Rational Unified Process* strukturiert sind. Die Phasen des Wasserfallmodells können in gewisser Weise als statischer Teil des *Rational Unified Process* verstanden werden.
Dieses gewählte Vorgehen könnte ein Indiz dafür sein, dass die Hochschullehrer tieferen Einblick in *best practices* und moderne Prozessmodelle des Softwareengineerings haben.

[3]Dieser Abschnitt enthält die für die Modellierungskompetenz relevanten Anteile aus der eigenen Veröffentlichung [Lehner et al. 2010].

Ein Proband aus der Gruppe der Fachdidaktiker beschreibt in seiner Antwort lediglich den Durchlauf der Requirements-Phase und bestätigt somit die zugehörige Komponente im Rahmenmodell.
Ferner empfindet der Fachwissenschaftler die Softwareentwicklung als einen höchst kommunikativen und kooperativen Prozess. Neben sozialen Kompetenzen, wie die Fähigkeit im Team zu arbeiten, nennt er soziale Kompetenzen als wichtige Voraussetzung, damit eine enge Zusammenarbeit zwischen Entwicklern und Kunden geschehen kann. Beide genannten Gruppen repräsentieren unterschiedliche fachliche Expertise, die es zusammenzubringen gilt. Der Verzicht auf das jeweilig domänenrelevante Vokabular und der Einsatz von zweckmäßigen grafischen Beschreibungsmitteln vereinfachen diesen Prozess in gemeinsamen Austausch.

Antwort: *„Die Verwendung von kontextrelevant angepassten grafischen Beschreibungsmitteln ermöglichen gute Diskussionen zwischen Kunden und Domänenexperten.“*

Kommunikative Fähigkeiten sind weiterhin eine Bedingung für die Integration von informatikfremden Personen in den SWE-Prozess. Derartige Kompetenzen fördern zudem die Effektivität und Performanz innerhalb der Entwicklergruppe.
Ein befragter Fachdidaktiker sieht in der Zusammenarbeit in Teams ebenso einen wichtigen Faktor, von dem der Erfolg eines SWE-Projekts (auch im schulischen Kontext) abhängt. Dies umfasst insbesondere die Fähigkeit konstruktive Kritik auszuüben und mit Kritik umgehen zu können. Die Rolle des Lehrers sieht der Fachwissenschaftler hierbei als Coach und Beobachter des kooperativen Prozesses. Hierbei ist seine Aufgabe Gruppen bei Problemen ggf. neu zu arrangieren.
Nachfolgend werden die Ergebnisse dieser Untersuchung vorgestellt. Dies beinhaltet das oben dargestellte Fallbeispiel (Informatische Modellierung und sozial-kommunikative Kompetenzen) als auch ein weiteres durchgeführtes Fallbeispiel (Systemverständnis).
Neben den bereits im Fallbeispiel erläuterten Ergebnissen der Untersuchung wurden wiederkehrende Charakteristika bei den einzelnen befragten Gruppen identifiziert. Hierzu werden nachfolgend einige Beispiele dargestellt.
Unsere Annahme gegenüber den Fachwissenschaftlern bestand darin, dass diese hinsichtlich der Tauglichkeit der Szenarien im schulischen Bereich sehr skeptisch sind und diese nicht befürworten. Im Gegensatz dazu konnten wir ein gegenteiliges Muster in den Antworten der Fachwissenschaftler auffinden, die den Einsatz des Szenarios *Warenwirtschaftssystem* im schulischen Einsatz befürworten.
Die folgenden Auszüge aus den Interviews beziehen sich jeweils auf einen beispielhaften

Vertreter aus der Gruppe der Fachwissenschaftler, Fachdidaktikder und Fachleiter.

Antwort (Fachwissenschaftler): *„Die Implementierung eines Warenwirtschaftssystem kann im schulischen Kontext zweifelsfrei umgesetzt werden.“*

Antwort (Fachdidaktiker): *„Ich bin nicht in der Lage in diesem Zusammenhang didaktische und methodische Empfehlungen auszusprechen, ohne mir darüber im Vorfeld Gedanken zu machen.“*

Antwort (Fachleiter): *„Dieses Szenario könnte angemessen für den schulischen Einsatz im Informatikunterricht sein. Es können allerdings Schwierigkeiten in bestimmten Fällen auftreten.“*

In diesem Beispiel war es der Fachwissenschaftler, der sich entgegen der Annahme positiv (und nicht wie angenommen negativ) bzgl. des Einsatzes des Szenarios im schulischen Kontext und dessen Tauglichkeit geäußert hat. Konträr waren die Aussagen des Fachdidaktikers und des Fachleiters, die grundsätzlich skeptischer gegenüber dem Einsatz im schulischen Umfeld eingestellt waren.
Anhand dieser Äußerungen der verschiedenen Probandengruppen wird deutlich, dass eine Generalisierung, dass Fachwissenschaftler grundsätzlich negativ gegenüber schulischen Themen eingestellt sind, kritisch geprüft werden muss. Derartige negative Einstellungen gegenüber schulischen Themen können auch von vorherigen Erfahrungen in diesem Themenbereich und anderen Faktoren abhängen.
Die Auswertungen haben überdies gezeigt, dass insbesondere Fachwissenschaftler Unsicherheiten gegenüber Themengebieten geäußert haben, die außerhalb ihres Forschungsgebiets lagen. In diesem Zusammenhang äußerte ein Fachwissenschaftler im Bereich Algorithmen und Datenstrukturen:

Antwort: *„Ich möchte in diesem Zusammenhang erwähnen, dass diese Thematik im Bereich von Datenbanksystemen außerhalb meiner Expertise liegt.“*

Unabhängig davon gab er uns hilfreiche Hinweise zu einer möglichen Lösungsstrategie des vorgestellten Szenarios. Ein weiterer Fachwissenschaftler weigerte sich eine Frage im Bereich der objektorientierten Modellierung zu beantworten, da ihm die insbesondere im schulischen Einsatz verwendeten *CRC-Karten* nicht geläufig waren. Dieses Antwortmuster war allerdings nicht nur bei den Fachwissenschaftlern festzustellen: Ein befragter Fachdidaktiker lehnte die Beantwortung und Schilderung zur persönlichen Vorgehensweise im Rahmen des Szenarios *Warenwirtschaftssystem* ab. Er äußerte, dass er ohne

intensive vorherige Planung keine Hinweise zu seinem Vorgehen zur Lösung des Aufgabenszenarios geben könne.
Eine weitere Gegebenheit, die im Rahmen der Interviews aufgetreten ist, ging von einem Fachwissenschaftler (Experte im Bereich Softwaretests) aus: Dieser wollte das Szenario zum Thema *Software-Test* überspringen, da seiner Meinung nach die Anforderungsanalyse nicht korrekt durchgeführt wurde.

Antwort: *„Ich möchte hier die Bearbeitung des Szenarios abbrechen. Die Anforderungsanalyse wurde nicht akkurat durchgeführt; deshalb kann die Testphase nicht geplant werden.“*

Die oben genannten Verhaltensmuster der Interviewten bei der Beantwortung der Fragen waren im Hinblick auf die gesamte Befragung eine Ausnahme. In den meisten Fällen konnten die Interviewten ihre Vorgehensweise zur Lösung des jeweiligen Szenarios detailliert beschreiben und nannten uns relevante Kompetenzfacetten, die ihrer Meinung nach erfolgsrelevant wären. In diesem Zusammenhang wurde erwähnt, dass Informatikunterricht zunächst ein breites Grundlagenwissen vermitteln muss, bevor man sich mit derartigen Fallbeispielen beschäftigt.
Eine weitere Gruppe von Fachdidaktikern versuchte ihre Vision von Informatikunterricht bei der Beantwortung der Szenarien zu propagieren.

Antwort: *„Bevor dieses Szenario in der Schule thematisiert wird, gilt es die dahinterliegenden fundamentalen Konzepte zu verstehen.“*

Diese Gruppe der Befragten hat häufig keinen direkten Kontakt zu Schülern. Daher fielen die Antworten im Hinblick auf das erwartete Schülerverhalten zur Lösung des Szenarios eher abstrakt aus und hatten keinen Bezug zu konkreten Lernprozessen.

Antwort: *„Es hängt davon ab, was die Schüler zuvor gelernt haben.“*

Je erfahrener die Experten im Bereich der Schulinformatik waren, desto einfacher fiel uns die Zuordnung von Sinneinheiten zu den jeweiligen Komponenten des theoretisch abgeleiteten Kompetenzmodells. Hierbei fiel insbesondere auf, dass fast alle befragten Fachleiter das Szenario aus Perspektive des Lehrenden sahen, obwohl bei der Fragestellung explizit nach dem persönlichen Vorgehen zur Lösung des Problems gefragt wurde. In einigen Fällen stellte sich jenes Dilemma vor der Beantwortung der jeweiligen Frage durch den Experten heraus.

Antwort: *„Bevor ich diese Frage beantworte möchte ich eine Gegenfrage stellen: Soll ich mein persönliches Vorgehen zur Lösung des Szenarios vorstellen oder mein Vorgehen als Lehrer in einem konkreten Lernprozess?“*

Andere Experten schilderten ihr Vorgehen unmittelbar aus Perspektive des Lehrenden:

Antwort: *„So wie ich Sie verstanden habe, soll ich das Szenario aus Sicht des Lehrers beschreiben: Dann würde ich zunächst ein Mindmap zu den Themen, die meine Schüler in diesem Zusammenhang lernen müssen, erstellen.“*

Weitere Experten ergänzten ihre Beschreibung der eigenen Lösungsstrategie mit möglichen Eindrücken aus Perspektive eines Schülers, obwohl sie nicht dazu aufgefordert wurden.
Zusammenfassend konnten verschiedene Vorgehensmuster bei den drei Interviewtengruppen *Fachwissenschaftler*, *Fachdidaktiker* und *Fachleiter* aufgefunden werden und beispielhaft illustriert werden. Dennoch ist bei der Generalisierung jener Ergebnisse Vorsicht geboten, da sich diese Ergebnisse auf einen kleinen Teil des empirischen Materials beziehen. Ferner könnte es sein, dass diese Ergebnisse eher auf die individuellen Erfahrungen des Interviewten zurückzuführen sind als auf seine Gruppenzugehörigkeit.
Insgesamt lässt sich bezüglich der hier vorgestellten Fallstudie feststellen, dass wir wertvolle Hinweise im Hinblick auf die Eignung unseres theoretisch abgeleiteten Kompetenzmodells in Erfahrung bringen konnten. Insbesondere die Kompetenzdimension *K1 Aufgabenbereiche* mit den Komponenten *Systemgestaltung* und deren Unterkategorien wurden durch die Ausführungen der Interviewten bestätigt. Diese ist von besonderer Wichtigkeit, da jene einen strukturgebenden Hauptbestandteil des Kompetenzmodells ausmacht.
Zur Entwicklung eines empirisch gesicherten Kompetenzmodells gilt es, die Inhalts- und Kriteriumsvalidität des entwickelten Kompetenzmodells zu überprüfen. Die inhaltliche Validität soll durch ein Expertenrating sichergestellt werden. Die Kriteriumsvalidität wird im Rahmen dieser Arbeit überprüft, indem – wie im folgenden Kapitel beschrieben – ein Messinstrument entwickelt wird, das die verschiedenen Kompetenzfacetten misst und Aufschluss über die Kriteriumsvalidität gibt. Die Korrelation zwischen diesen beiden Kriterien kann als Indikator für Kriteriumsvalidität des Kompetenzmodells gesehen werden.

5.6. Kategoriendefinitionen zum informatischen Modellieren

Ausgehend vom empirisch verfeinerten Kompetenzmodell mussten Kategoriendefinitionen verfasst werden, sog. Kompetenzprofile. Hierzu wurden ausgehend von den Bezeich-

nungen der jeweiligen Unterkomponenten von *K1.3 Systemgestaltung* und standardisierter Operatorenlisten Kompetenzen definiert. Hierbei bestand die Zielsetzung, überprüfbare Definitionen der einzelnen Kompetenzbereiche zu verfassen, die als Grundlage für die Konzeption von Aufgaben für das Messinstrument Modellierung dienen. Folglich mussten die teilweise abstrakten Bezeichnungen der Kompetenzkomponenten derart konkretisiert werden, dass diese operationalisiert werden konnten und auf dessen Grundlage die Item-Entwicklung stattfinden konnte.
Im Folgenden ist die standardisierte Operatorenliste, die für die Kompetenzprofildefinition zugrunde gelegt wurde, aufgeführt(Übersicht Operatoren NRW 2007).

Anforderungsbereich I

- Angeben:
  Ohne nähere Erläuterungen und Begründungen aufzählen, nennen.
- Beschreiben
  Sachverhalte oder Zusammenhänge unter Verwendung der Fachsprache in eigenen Worten verständlich wiedergeben.
- Darstellen, Dokumentieren
  Zusammenhänge, Sachverhalte oder Arbeitsverfahren in strukturierter Form graphisch oder sprachlich wiedergeben.
- Einordnen
  Mit erläuternden Hinweisen in einen genannten Zusammenhang einfügen.
- Erläutern*
  Einen Sachverhalt auf der Grundlage von Vorkenntnissen so darlegen, dass er verständlich wird.
- Überführen, Übertragen
  Eine Darstellung in eine andere Darstellungsform bringen.

Anforderungsbereich II

- Analysieren
  Eine konkrete Materialgrundlage untersuchen, einzelne Elemente identifizieren und Beziehungen zwischen den Elementen erfassen. Der Operator *Analysieren* wird oft in Kombination mit einem weiteren Operator benutzt, der angibt, wie das Analyseergebnis darzustellen ist.

- Bestimmen, Ermitteln
  Mittels charakteristischer Merkmale einen Sachverhalt genau feststellen und beschreiben.
- Entwerfen, Entwickeln
  Herstellen und Gestalten eines Systems von Elementen unter vorgegebener Zielsetzung.
- Erweitern, Vervollständigen
  Eine gegebene Struktur um Bestandteile erweitern.
- Herleiten, Ableiten
  Die Entstehung oder Ableitung eines gegebenen oder beschriebenen Sachverhaltes aus anderen oder aus allgemeinen Sachverhalten darstellen
- Implementieren**
  Algorithmen und Datenstrukturen in einer Programmiersprache aufschreiben.
- Modellieren**
  Zu einem Ausschnitt der Realität ein informatisches Modell anfertigen.
- Vergleichen
  Nach vorgegebenen oder selbst gewählten Gesichtspunkten Gemeinsamkeiten, Ähnlichkeiten und Unterschiede ermitteln und darstellen.
- Zeigen
  Eine Aussage, einen Sachverhalt nach Berechnungen, Herleitungen oder logischen Begründungen bestätigen.

Anforderungsbereich III

- Begründen**
  Einen Sachverhalt oder eine Entwurfsentscheidung durch Angabe von Gründen erklären.
- Beurteilen
  Zu einem Sachverhalt ein selbstständiges Urteil unter Verwendung von Fachwissen und Fachmethoden formulieren und begründen.
- Stellung nehmen
  Unter Heranziehung relevanter Sachverhalte die eigene Meinung zu einem Problem argumentativ entwickeln und darlegen.

[4]Unter Berücksichtigung der Operatoren ergeben sich folgende Kategoriendefinitionen:

K1.3.1	**Anforderungsanalyse**	
	K1.3.1.1	Die Lernenden können eine geeignete (Software) Plattform/Basistechnologie auswählen, um das zu erstellende SW-Projekt zu entwickeln.
	K1.3.1.2	Die Lernenden sind in der Lage, Anwendungsfälle (Use Cases) zu ermitteln und anzugeben (benennen), diese zu analysieren (durchzuspielen); sie sind diesbezüglich auch in der Lage, Use Case Diagramme zu entwickeln. Hierbei können sich die Lernenden einen Eindruck verschaffen, was die zu entwickelnde Software zu leisten hat.
	K1.3.1.3	Die Lernenden sind in der Lage, funktionale Anforderungen an die zu entwickelnde Software zu ermitteln; dabei sind sie befähigt, die Ziele (z.B. funktionale Anforderungen), Grenzen (z.B. Abgrenzung zu bestehenden Softwaresystemen) und Stakeholder innerhalb der Problemdomäne zu ermitteln. Hierbei besteht wiederum die Zielsetzung herauszufinden, was das zu entwickelnde System leisten soll.
	K1.3.1.4	Die Lernenden sind in der Lage, eine tabellarische Use Case Beschreibung in ein Aktivitätendiagramm zu überführen. Hierdurch können mögliche Anwendungsszenarien genauer analysiert werden.
	K1.3.1.5	Die Lernenden sind in der Lage, die zuvor ermittelten funktionalen Anforderungen für andere verständlich und nachvollziehbar darzustellen (dokumentieren). Hierbei besteht die Zielsetzung, ein gemeinsames Dokument (im Sinne eines Pflichtenhefts) für die SWE-Teams im Hinblick auf die weiteren Phasen des SWE-Prozesses zu entwickeln.

[4]Hinweis: Die mit * gekennzeichneten Operatoren können sowohl dem AFB I als auch dem AFB II und die mit ** gekennzeichneten Operatoren dem AFB II oder dem AFB III zugeordnet werden.

K1.3.2	Analyse	
	K1.3.2.1	Die Lernenden können objektorientierte Begrifflichkeiten angeben und erläutern. Dies ist Grundvoraussetzung, um eine objektorientierte Dekomposition durchführen zu können.
	K1.3.2.2	Die Lernenden sind in der Lage, eine objektorientierte Dekomposition durchzuführen; d.h., sie können anhand einer textuellen Beschreibung des Problembereichs mögliche Klassenkandidaten, Attribute und Methoden ermitteln (auffinden) und diese in eine formale Darstellungsform überführen. Hierbei besteht die Zielsetzung ein Modell des Problembereichs zu erstellen.
	K1.3.2.3	Die Lernenden sind in der Lage, relevante statische und dynamische UML-Diagramme ohne implementierungsspezifische Details zu entwickeln (z.B. *CRC-Karten*). Durch diese formale konzeptionelle Modellierung erhalten die Lernenden einen vertieften Einblick in die Problemdomäne.
	K1.3.2.3.1	**CRC-Karten:** Die Lernenden sind in der Lage, ein (textuelles) Szenario in Form von *CRC-Karten* darzustellen.
	K1.3.2.3.2	**Objektdiagramm:** Die Lernenden sind in der Lage, Objektdiagramme zu entwickeln; diese können sie ggf. ausgehend von Use Case Diagrammen überführen; sie können relevante Objekte ermitteln, erläutern wie die Objekte untereinander kommunizieren und von gleichartigen Objekten Klassen (im Hinblick auf ein Klassendiagramm) ableiten.
	K1.3.2.3.3	**Sequenzdiagramm:** Die Lernenden sind in der Lage, Sequenzdiagramme zu entwickeln; diese können sie ggf. ausgehend von Use Case Diagrammen überführen.
	K1.3.2.3.4	**Klassendiagramm:** Die Lernenden sind in der Lage, Analyse-Klassendiagramme zu entwickeln; diese können sie ggf. ausgehend von textuellen Beschreibungen, *CRC-Karten*, Use Case Diagrammen oder Objektdiagrammen überführen; sie können Klassen inklusive Attributen und Methoden definieren, Assoziationen festlegen und sinnvolle Vererbungsstrukturen entwickeln.

K1.3.3	Design	
	K1.3.3.1	Die Lernenden sind in der Lage, die Architektur der zu entwickelnden Software zu bestimmen; dabei wählen sie eine geeignete Programmiersprache aus, berücksichtigen Aspekte der Verteilung, Nebenläufigkeit/Parallelität und möglicher Entwurfsmuster. Dies ist eine wichtige Voraussetzung für die Entwicklung von entwurfsspezifischen UML-Diagrammarten.
	K1.3.3.2	Die Lernenden sind in der Lage, sinnvolle Schnittstellen zu bestimmen um eine spätere erfolgreiche Integration von Programmmodulen zu ermöglichen.
	K1.3.3.3	Die Lernenden sind in der Lage, relevante **statische und dynamische** UML-Diagramme mit implementierungsspezifischen Details zu entwickeln. Hierdurch entsteht ein entwurfsspezifisches Modell, welches in Quellcode einer objektorientierten Hochsprache überführt werden kann.
	K1.3.3.3.1	**Klassendiagramm:** Die Lernenden sind in der Lage, Entwurfs-Klassendiagramme zu entwickeln.
	K1.3.3.3.2	**Zustandsdiagramm:** Die Lernenden sind in der Lage, Zustandsdiagramme zu entwickeln.
	K1.3.3.3.3	**Verteilungsdiagramm:** Die Lernenden sind in der Lage, Verteilungsdiagramme zu entwickeln; sie können mit Hilfe der Verteilungsdiagramme Programmmodule (z.B. auf Server und Client) aufteilen.

K1.3.4	Implementierung	
	K1.3.4.1	Die Lernenden sind in der Lage, die Architektur der zu entwickelnden Software zu bestimmen; dabei wählen sie eine geeignete Programmiersprache aus, berücksichtigen Aspekte der Verteilung, Nebenläufigkeit/Parallelität und möglicher Entwurfsmuster. Dies ist eine wichtige Voraussetzung für die Entwicklung von entwurfsspezifischen UML-Diagrammarten.
	K1.3.4.1.1	Die Lernenden sind in der Lage, Programmierkonzepte, wie z.B. das Variablenkonzept und Kontrollstrukturen (Bedingte Anweisung, Schleifenkonstruktion) in der Programmiersprache zu implementieren.
	K1.3.4.1.2	Die Lernenden sind in der Lage, ein Klassendiagramm in objektorientierten *Java-Code* zu überführen; Sie können Klassen, Attribute und Methoden sowie Assoziationen und Vererbungsstrukturen in *Java-Code* implementieren.
	K1.3.4.1.3	Die Lernenden sind in der Lage, Programmbibliotheken (z.B. *Java-Swing*) erfolgreich in eigene Programmmodule einzubinden.
	K1.3.4.2	Die Lernenden sind in der Lage, mit Hilfe von integrierten Entwicklungsumgebungen (IDEs) Programmmodule zu implementieren und zu integrieren.
	K1.3.4.3	Die Lernenden sind in der Lage, mit Hilfe einer Versionsverwaltungssoftware (z.B. Subversion) Programmmodule und deren Versionierung zu verwalten.
	K1.3.4.4	Die Lernenden sind in der Lage, die selbst implementierten Programmmodule nachvollziehbar (im Hinblick auf gute Wartbarkeit) zu dokumentieren (z.B. mit *Java-Doc*).
	K1.3.4.5	Die Lernenden sind in der Lage, Programmmodule sinnvoll in ein bestehendes Softwaresystem zu integrieren. Somit können Teile der zu entwickelnden Software zu einem lauffähigen System aggregiert werden.

K1.3.5	**Test**	
	K1.3.5.1	Die Lernenden sind in der Lage, ein bestehendes Softwaresystem systematisch zu testen. Hierbei besteht die Zielsetzung unter anderem darin, zu überprüfen, ob die zuvor spezifizierten funktionalen Anforderungen erfolgreich umgesetzt wurden.
	K1.3.5.1.1	Die Lernenden sind in der Lage, zu Beginn der Testphase einen geeigneten Testplan zu entwickeln.
	K1.3.5.1.2	Die Lernenden sind in der Lage, gängige Vorgehensmodelle des Testens durchzuführen (z.B. Model-Checking, Whitebox-, Blackbox-Testverfahren, ...).
	K1.3.5.1.3	Die Lernenden sind in der Lage, Testfälle zu ermitteln (Extremfälle und unerwartete Eingabedaten erzeugen) oder zu entwickeln; sie können diese zum Test verwenden und die daraus resultierenden Ausgaben protokollieren.
	K1.3.5.1.4	Die Lernenden sind in der Lage, automatisierte Tests durchzuführen.

K1.3.6	**Iteratives Vorgehen**	
	K1.3.6.1	Die Lernenden sind in der Lage, abhängig von der jeweiligen Iteration des SWE-Prozesses, sinnvolle Modellierungstechniken auszuwählen, anzuwenden und zu beurteilen.
	K1.3.6.2	Die Lernenden sind in der Lage zu beurteilen, ob ein erneutes Durchlaufen einer bereits absolvierten Phase des SWE-Prozesses erforderlich ist; sie können abhängig von den auftretenden Problemen in der aktuellen Phase eine sinnvolle vorherige Phase auswählen, die es erneut zu durchlaufen gilt.

Im Hinblick auf die Item-Entwicklung besteht die Zielsetzung, für jeden Kompetenzbereich (definiert durch die Kategoriendefinitionen) mehrere spezifische Items zu entwickeln. Hierbei gilt es zu beachten, dass die Testitems in einigen Fällen auch mehreren Kompetenzprofilen zugeordnet werden können.

Im folgenden Kapitel sollen in einem ersten Schritt die Item-Entwicklung auf Basis der Kategoriendefinitionen erläutert werden. Diese dienen als Bestandteile des zu entwickelnden Kompetenzmessinstruments.

5.7. Zusammenfassung

Das Ergebnis dieses Kapitels ist ein wichtiger Meilenstein für die Entwicklung des Messinstruments: Das resultierende empirisch verfeinerte Kompetenzrahmenmodell und die dazugehörigen Kategoriendefinitionen sind eine wichtige Voraussetzung zur Entwicklung von Aufgabenitems sowie deren inhaltlicher Fokussierung.
Dementsprechend erfolgte die empirische Verfeinerung der jeweiligen Kompetenzdimensionen und -komponenten. Hierbei kam ein Interviewverfahren zum Einsatz, das sich an der *Critical Incident Technique* orientiert und mit dem ermittelt wurde, welche Kenntnisse, Strategien, Fähigkeiten und Einstellungen im Sinne der Befragten erforderlich sind, um problemlösend zu handeln.
Alle Interviews wurden aufgezeichnet, transkribiert und unter Verwendung der qualitativen Inhaltsanalyse ausgewertet. Hier wurden zuvor im Interviewmaterial aufgefundene Sinneinheiten anhand der im Vorfeld entwickelten Kategoriendefinitionen, Ankerbeispielen und Kodierregeln den Kategorien des Kompetenzmodells zugeordnet. Hierdurch erfolgte sowohl eine empirische Überprüfung des Kompetenzrahmenmodells aus Kapitel 4 als auch eine Korrektur, Ergänzung und Ausdifferenzierung des Modells hinsichtlich neuer und zusätzlich zu berücksichtigender Kompetenzaspekte.
Konkret hat sich die Komponente *K1.3 Systemgestaltung* als wichtiger Part des Kompetenzmodells gefestigt, da jene grundlegende Kompetenzen zur Entwicklung von Informatiksystemen umfasst. Darüber hinaus ist ein wichtiges Ergebnis dieses Kapitels die empirische Verfeinerung der Unterkomponenten von K1.3. Hier hat sich gezeigt, dass bestimmte Kompetenzkomponenten, wie z.B. die Geschäftsmodellierung, sich als weniger relevant erwiesen haben. Diese wurden somit auch bei der Entwicklung des Kompetenzmessinstruments weniger berücksichtigt. Bei weiteren Kompetenzkomponenten war es sinnvoll eine Binnendifferenzierung vorzunehmen. So wurde die Kompetenzkomponente *Analyse & Design* in *Analyse* und *Design* aufgeteilt. Ferner wurde in dem Zusammenhang eine weitere Komponente *Iteratives Vorgehen* dem Kompetenzmodell hinzugefügt.
Anhand des empirisch verfeinerten Kompetenzmodells wurden Kategoriendefinitionen, sog. Kompetenzprofile formuliert. Hierzu wurden ausgehend von den Bezeichnungen der jeweiligen Kompetenzkomponenten und standardisierten Operatorenlisten Kompetenzen definiert. Hierbei bestand die Zielsetzung, überprüfbare Definitionen der einzelnen Kom-

petenzbereiche zu verfassen, die als Grundlage für die Konzeption von Aufgabenitems dienen. Einzelne abstrakte Bezeichnungen mussten derart konkretisiert werden, dass diese operationalisiert werden konnten und auf deren Grundlage die Item-Entwicklung stattfinden konnte.
Auf Grundlage der definierten Kategoriendefinitionen erfolgt im folgenden Kapitel 6 die Entwicklung von Aufgaben und Items zur Überprüfung von objektorientierter Modellierungskompetenz.

6. Entwurf eines Messinstruments für informatische Modellierungskompetenz und Entwicklung eines Lehr-/Lernarrangements zur Erprobung

Dieses Kapitel fokussiert die Entwicklung von Aufgaben und darin enthaltener Items auf Grundlage der zuvor formulierten Kompetenzprofile (Kategoriendefinitionen). Hierbei werden zunächst theoretische Grundlagen zur Testentwicklung und zur Fragebogenkonstruktion aufgezeigt und beispielhaft der Entwicklungsprozess einzelner repräsentativer Aufgaben des Messinstruments dargestellt.

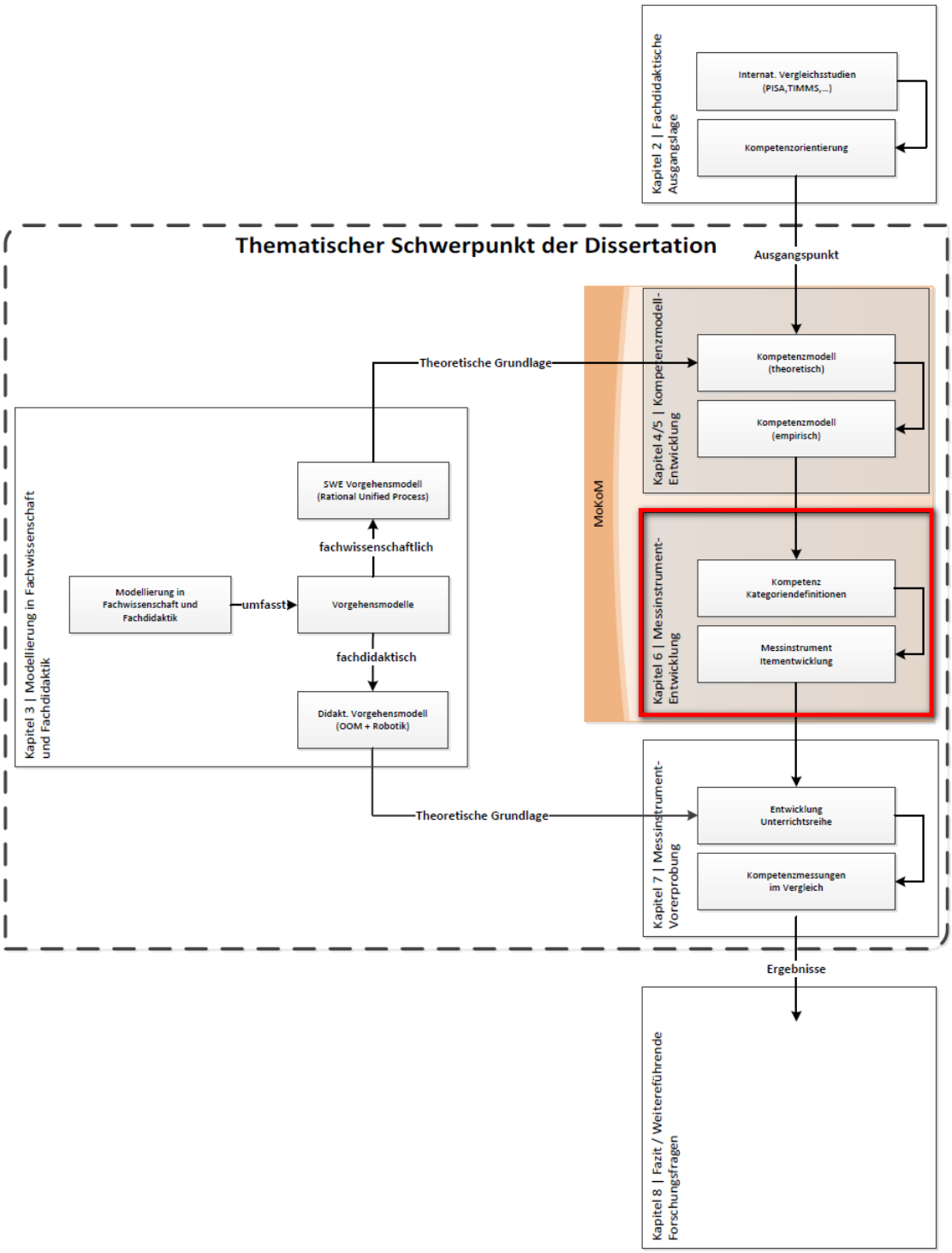


Abbildung 6.1.: Kapitel 6 im Gesamtkontext der Arbeit

Die folgende Abbildung 6.2 illustriert die Vorgehensweise zur Entwicklung der Aufgaben des Messinstruments. Zunächst werden auf Grundlage der zuvor formulierten Kompetenzkategorien zugehörige Items entwickelt. Die Items werden jeweils zu Aufgaben kombiniert und mit einem für die Probanden lebensweltnahen Stimulusmaterial kombiniert. Im Sinne unseres Kompetenzverständnisses nach Weinert werden neben den kognitiven Kompetenzbereichen auch die nicht kognitiven Kompetenzbereiche aus der Kompetenzdimension *K4* adressiert und explizit in den *Aufgaben 1 - 10* mit berücksichtigt. Somit soll sichergestellt werden, dass Kompetenzen ganzheitlich abgefragt werden. Eine Abfrage von isolierten Fähigkeiten ist im Sinne des festgelegten Kompetenzverständnisses nicht wünschenswert.

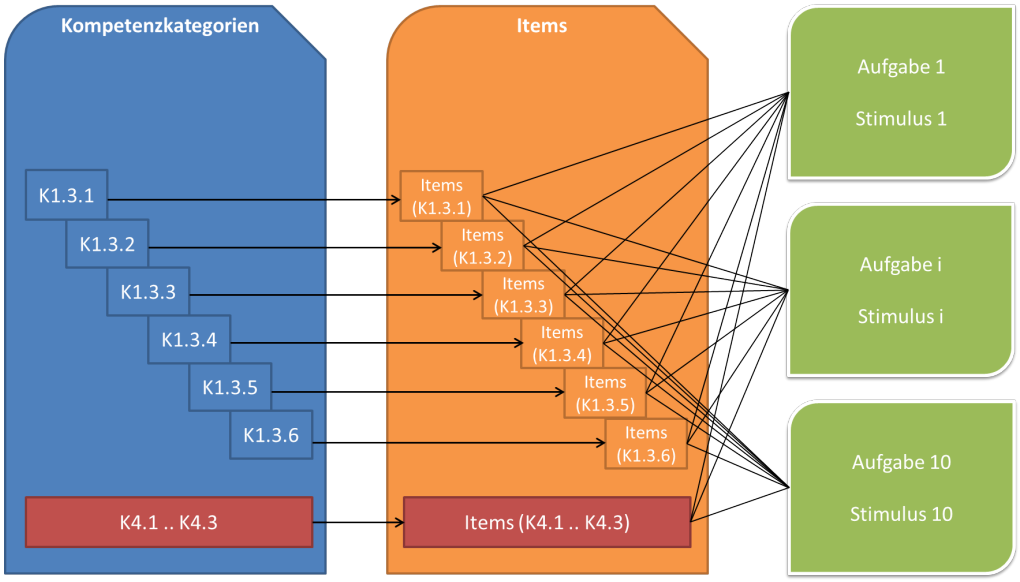


Abbildung 6.2.: Illustration der Aufgabenentwicklung

Zur Evaluation des entwickelten Messinstruments umfasst dieses Kapitel ferner die Entwicklung und Darstellung einer geeigneten Unterrichtsreihe. Diese basiert auf dem in Kapitel 3 vorgeschlagenen Vorgehensmodell *Modellierung & Robotik* und dem Paderborner *Informatik Lernlabor*. In diesem Zusammenhang wurde eine *LEGO Mindstorms* basierte Inhaltseinheit *Kommissionierstation* entwickelt, die sich im Rahmen unserer Erfahrungen in der Hochschullehre als kompetenzförderlich erwiesen hat.

6.1. Entwicklung von Aufgabenitems

Auf Basis des entwickelten Kompetenzmodells wurde in einem weiteren Arbeitsschritt ein erstes Instrumentarium zur Diagnose von Kompetenzständen zu ausgewählten Teilen des Modells entwickelt und im Rahmen eines Unterrichtsversuchs zur Förderung von Kompetenzen im Bereich des informatischen Modellierens erprobt. Bei der Testentwicklung wurden in erster Linie sog. objektive Testverfahren bzw. *Situational Judgement Tests (SJT)* entwickelt, die relevante Kompetenzen anhand konkreter Lösungen von repräsentativen Aufgabenstellungen der Unterrichtsdomäne ermitteln und bewertbar machen. Hierbei wurden Erfahrungen zur Testentwicklung bei der TIMMS-, PISA- und DESI-Studie [Baumert et al. 2000], [Prenzel und Deutschland 2004], [Beck und Klieme 2007] sowie von Testentwicklungen im Bereich der Arbeits- und Organisationspsychologie [Schaper und Horvath 2008] berücksichtigt. Unter Bezugnahme auf die Kompetenzbeschreibungen bzw. -profile des Kompetenzmodells wurden zunächst geeignete Aufgabenstellungen, die entsprechende Kompetenzanforderungen repräsentieren, generiert und in einem zweiten Schritt passende Antwortformate (in Form vorgegebener Antwortalternativen oder freier bzw. offener Antwortformate) entwickelt. Auf der Basis dieses Vorgehens wurden insgesamt 62 Items zur Erfassung der Kompetenzbereiche *K1.3 Systemgestaltung* und *K4 Nicht-kognitive Fähigkeiten* konstruiert.

Für den Kompetenzbereich *K1.3 Systemgestaltung* wurden für den Informatikunterricht geeignete Items entwickelt und zu Aufgaben kombiniert. Diese orientieren sich an dem PISA/TIMMS-Aufgabenmuster. Sie stellen ein jeweils für die Schüler lebensweltnahes Stimulus- bzw. Aufgabenmaterial voran und enthalten daran anschließend sowohl 23 offene als auch 11 Multiple-Choice(MC)-Antwortformate. In diesem Kontext wurden jeweils ein bis zwei Aufgaben zur Operationalisierung der im Kompetenzstrukturmodell enthaltenen Kompetenzbereiche *K1.3.1 Anforderungsanalyse*, *K1.3.2 Analysephase*, *K1.3.3 Designphase*, *K1.3.4 Implementierung*, *K1.3.5 Softwaretest* sowie *K1.3.6 Iteratives Vorgehen* erstellt.

Exemplarisch für eine Aufgabe mit lebensweltnahem Stimulusmaterial und ein enthaltenes Item im MC-Antwortformat sei die Aufgabe 5B (siehe Fragebogen im Anhang) genannt. Ausgehend von einer textuellen Szenariobeschreibung einer Schulbibliothek werden die Probanden aufgefordert, eines von zwei Analyse-Klassendiagrammen auszuwählen, welches die Schulbibliothek korrekt modelliert. Die Zielsetzung dieser Aufgabenstellung (bzw. des enthaltenen Items) besteht darin, die Teilkompetenz *K1.3.3.3 (Lernende sind in der Lage, relevante statische und dynamische UML-Diagramme ohne implementierungsspezifische Details zu entwickeln und zu beurteilen)* abzufragen.

Als Beispiel für SJT-Aufgabenstellung und einem Item mit offenem Antwortformat sei die Aufgabe 2 angeführt. Hierbei handelt es sich um eine Aufgabenstellung, bei der sich die Probanden in die Rolle eines IT-Projektmanagers versetzen sollen. Aus dieser Perspektive gilt es, die Auswahl einer geeigneten Programmiersprache für die Entwicklung eines plattformunabhängigen, verteilten Chat-Systems zu begründen. In diesem Zusammenhang werden sie u.a. gefragt, welche Eigenschaften der Programmiersprache *Java* die Entwicklung eines plattformunabhängigen Softwaresystems begünstigen. Hierbei enthält die Aufgabe ein Item mit offenem Antwortformat. Mit dieser Aufgabe und dem enthaltenen Item soll die Teilkompetenz *K1.3.4.1 (Lernende sind in der Lage, die Architektur der zu entwickelnden Software zu bestimmen und eine geeignete Programmiersprache auszuwählen)* abgefragt werden.

6.1.1. Zuordnung von Aufgaben zu Kompetenzkategorien

Die Items wurden inhaltlich entsprechend der einzelnen Kompetenzkategorien zur informatischen Modellierung (wie im vorherigen Kapitel erarbeitet) ausgerichtet. Hierbei wurden spezielle Items für Kompetenzkategorien entwickelt und zu Aufgaben kombiniert die möglichst mehrere im Strukturmodell abgebildete Kompetenzbereiche abfragen. Die folgende Tabelle stellt die Kompetenzkategorien mit der Zuordnung zu den einzelnen Aufgaben des Messinstruments dar.

Zuordnung Kompetenzkategorien zu Aufgabe 5

Kompetenzkategorien zu Aufgabe 5	
K1.3.1.1	**Analyse**
K1.3.3.3	Die Lernenden sind in der Lage, relevante statische und dynamische UML-Diagramme ohne implementierungsspezifische Details zu entwickeln (z.B. *CRC-Karten*). Durch diese formale konzeptionelle Modellierung erhalten die Lernenden einen vertieften Einblick in die Problemdomäne.
K1.3.3.3.1	**CRC-Karten:** Die Lernenden sind in der Lage, ein (textuelles) Szenario in Form von *CRC-Karten* darzustellen.
K1.3.3.3.4	**Klassendiagramm:** Die Lernenden sind in der Lage, Analyse-Klassendiagramme zu entwickeln; diese können sie ggf. ausgehend von textuellen Beschreibungen, *CRC-Karten*, Use Case Diagrammen oder Objektdiagrammen überführen; sie können Klassen inklusive Attributen und Methoden definieren, Assoziationen festlegen und sinnvolle Vererbungsstrukturen entwickeln.

Zuordnung Kompetenzkategorien zu Aufgabe 6

Kompetenzkategorien zu Aufgabe 6	
K1.3.2	**Anforderungsanalyse**
K1.3.2.2	Die Lernenden sind in der Lage, Anwendungsfälle (Use Cases) zu ermitteln und anzugeben (benennen), diese zu analysieren (durchzuspielen); sie sind diesbezüglich auch in der Lage, Use Case Diagramme zu entwickeln. Hierbei können sich die Lernenden einen Eindruck verschaffen, was die zu entwickelnde Software zu leisten hat.
K1.3.3	**Analyse**
K1.3.3.3	Die Lernenden sind in der Lage, relevante statische und dynamische UML-Diagramme ohne implementierungsspezifische Details zu entwickeln (z.B. *CRC-Karten*). Durch diese formale konzeptionelle Modellierung erhalten die Lernenden einen vertieften Einblick in die Problemdomäne.
K1.3.3.3.3	**Sequenzdiagramm:** Die Lernenden sind in der Lage, Sequenzdiagramme zu entwickeln; diese können sie ggf. ausgehend von Use Case Diagrammen überführen.

Zuordnung Kompetenzkategorien zu Aufgabe 10

Kompetenzkategorien zu Aufgabe 10	
K1.3.6	**Test**
K1.3.6.1	Die Lernenden sind in der Lage, ein bestehendes Softwaresystem systematisch zu testen. Hierbei besteht die Zielsetzung unter anderem darin, zu überprüfen, ob die zuvor spezifizierten funktionalen Anforderungen erfolgreich umgesetzt wurden.
K1.3.6.1.1	Die Lernenden sind in der Lage, zu Beginn der Testphase einen geeigneten Testplan zu entwickeln.
K1.3.6.1.3	Die Lernenden sind in der Lage, Testfälle zu ermitteln (Extremfälle und unerwartete Eingabedaten erzeugen) oder zu entwickeln; sie können diese zum Test verwenden und die daraus resultierenden Ausgaben protokollieren.
K1.3.7	**Iteratives Vorgehen**
K1.3.7.1	Die Lernenden sind in der Lage, abhängig von der jeweiligen Iteration des SWE-Prozesses, sinnvolle Modellierungstechniken auszuwählen, anzuwenden und zu beurteilen.
K1.3.7.2	Die Lernenden sind in der Lage zu beurteilen, ob ein erneutes Durchlaufen einer bereits absolvierten Phase des SWE-Prozesses erforderlich ist; sie können abhängig von den auftretenden Problemen in der aktuellen Phase eine sinnvolle vorherige Phase auswählen, die es erneut zu durchlaufen gilt.

6.1.2. Exemplarische Item-Entwicklung

In diesem Kapitel werden neben einer thematischen Übersicht aller Aufgaben, drei beispielhafte Aufgaben zur informatischen Modellierung vorgestellt.
Ausgehend von den zuvor definierten Kompetenzprofilen wurden die inhaltliche Strukturierung und die jeweiligen inhaltlichen Themen der Aufgaben festgelegt. Im Folgenden wird eine tabellarische Auflistung der Kompetenzprofile inkl. der jeweiligen Aufgabenitems im Messinstrument (siehe Anhang) aufgeführt.
Betrachten wir nun einen exemplarischen Ausschnitt aus dem Entstehungsprozess der

Aufgaben.

Aufgabe 5 - Item zur Analysephase

Die Zielsetzung der ersten Items (Teilaufgabe 5 a,b) war die Überprüfung des Kompetenzbereichs *K1.3 Analysephase*. Innerhalb der Items (a,b) werden gezielt die jeweiligen Teilkompetenzen abgefragt. Teilaufgabe a) wurde zur Messung des Kompetenzbereichs *K1.3.3.3* entwickelt.

> *K1.3.3.3 Die Lernenden sind in der Lage, relevante statische und dynamische UML-Diagramme ohne implementierungsspezifische Details zu entwickeln (z.B.* CRC-Karten*). Durch diese formale konzeptionelle Modellierung erhalten die Lernenden einen vertieften Einblick in die Problemdomäne.*

Insbesondere ging es darum den Diagrammtyp *CRC-Karten* zu thematisieren.

> *K1.3.3.3.1 CRC-Karten: Die Lernenden sind in der Lage, ein (textuelles) Szenario in Form von* CRC-Karten *darzustellen.*

Zunächst enthält die Aufgabe eine kurze Aufgabenbeschreibung die dem Probanden mithilfe standardisierter Operatoren die erforderlichen Tätigkeiten zur Lösung der Aufgabe vorstellt. Anschließend wird dem Probanden (hier Schüler der Sekundarstufe II an Gymnasien in NRW) ein Stimulus-Material aus seiner Erfahrungswelt präsentiert. Dies beschreibt den jeweiligen Kontext der Aufgabe und soll dem Probanden die Möglichkeit geben, sich mit der Aufgabe zu identifizieren. In diesem Aufgabenitem handelt es sich um eine vereinfachte Darstellung einer Schulbibliothek.

Aufgabe 5a
Sie wurden beauftragt, eine Software zur Verwaltung Ihrer Schulbibliothek zu entwickeln. In der Analyse-Phase sollen zunächst *CRC-Karten* für die wichtigsten Klassen erstellt werden. Ergänzen Sie hierzu die unten dargestellten *CRC-Karten* um die jeweiligen *Responsibilities* und *Collaborators*. Orientieren Sie sich hierbei an der Beschreibung der *Schulbibliothek I*. *Schulbibliothek I*: Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, ID-Nummer und Anzahl der Seiten. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter. (...)

Innerhalb der Aufgabe werden den Probanden rudimentäre *CRC-Karten* zur Verfügung gestellt, die es – wie oben beschrieben – zu ergänzen gilt. Hierbei müssen die jeweiligen *Responsibilities* und *Collaborators* eingetragen werden, sodass die *CRC-Karten* die „Schulbibliothek I“ korrekt modellieren. Ein Stiftsymbol symbolisiert zusätzlich die zu bearbeitenden Stellen innerhalb der Grafik.
Selbige soll Aufgaben mit lebensweltnahem Stimulusmaterial und einem Item mit offenem Antwortformat repräsentieren.
Teilaufgabe b) wurde zur Überprüfung des Kompetenzbereichs *K1.3.3.3.4* konzipiert.

> *K1.3.3.3.4 Klassendiagramm: Die Lernenden sind in der Lage, Analyse - Klassendiagramme zu entwickeln; diese können sie ggf. ausgehend von textuellen Beschreibungen,* CRC-Karten*, Use Case Diagrammen oder Objektdiagrammen überführen; sie können Klassen inklusive Attributen und Methoden definieren, Assoziationen festlegen und sinnvolle Vererbungsstrukturen entwickeln.*

Hier wird wiederum als erstes die Aufgabenbeschreibung dargelegt, gefolgt von einem weiteren Stimulusmaterial in Form einer Beschreibung *Schulbibliothek II*. Zur Abfrage des zugehörigen Kompetenzbereichs K1.3.3.3.4 enthält die Aufgabe ein Item mit MC-Antwortformat.
Bei der *Schulbibliothek II* handelt es sich um eine erweiterte Szenariobeschreibung mit deutlich erhöhter Komplexität, die im Gegensatz zur *Schulbibliothek I* dem Probanden Aspekte der Vererbung, die Unterscheidung Assoziation/Aggregation, etc. abverlangt.

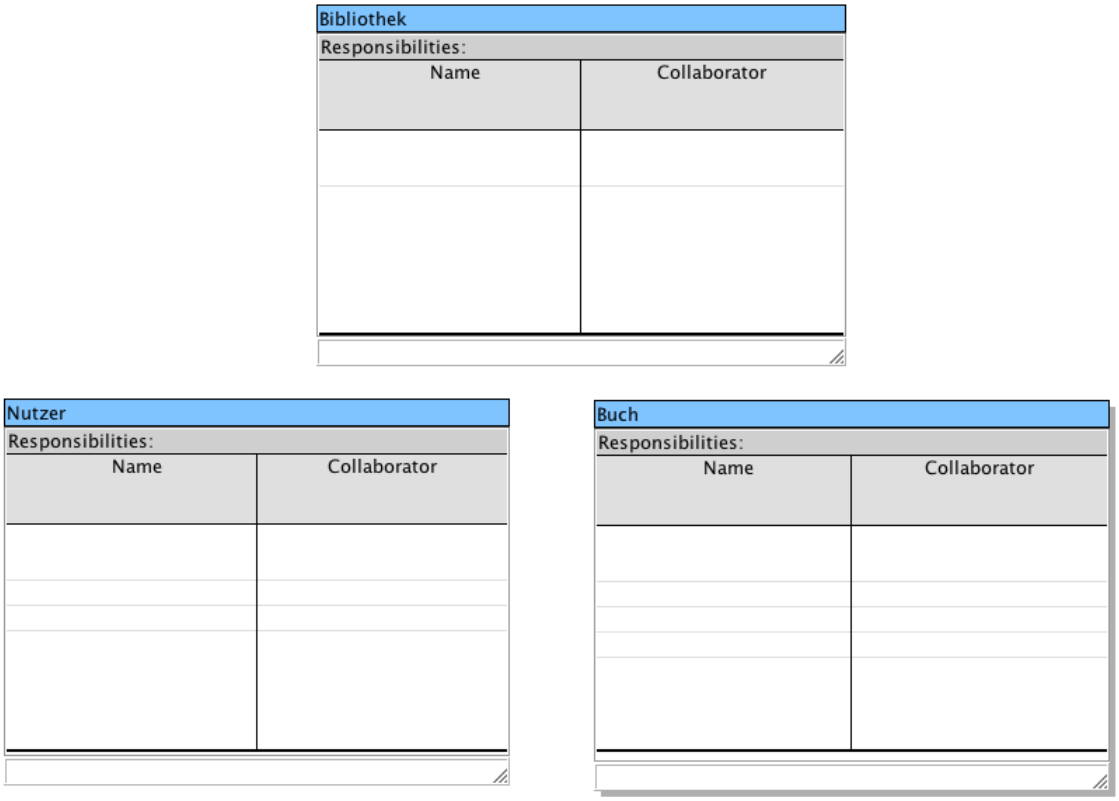


Abbildung 6.3.: *CRC-Karten* zur Schulbibliothek I

Aufgabe 5b
Wählen Sie das Klassendiagramm aus, das die unten beschriebene erweiterte Version der Schulbibliothek (Schulbibliothek II) korrekt modelliert. *Schulbibliothek II*: Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, ID-Nummer und Anzahl der Seiten. Es gibt Sachbücher, Lexika und Romane. Sachbücher sind zusätzlich gekennzeichnet durch ein Themengebiet, Lexika durch die Anzahl Bände sowie die jeweilige Bandnummer des Exemplars. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter. Unterschieden wird zwischen verschiedenen Benutzergruppen: Lehrer haben ein erstes und zweites Unterrichtsfach und dürfen höchstens vier Bücher gleichzeitig ausleihen. Zusätzlich stehen Sie als Berater für zwei bestimmte Sachgebiete der Fachbücher zur Verfügung. Schüler haben eine Jahrgangsstufe und dürfen höchstens zwei Bücher gleichzeitig ausleihen. (...)

Der Proband ist nun aufgefordert per Multiple Choice eins von zwei Klassendiagrammen auszuwählen, welches das in *Schulbibliothek II* beschriebene Szenario korrekt modelliert. Um die Entscheidung zu begründen und zu verhindern, dass die korrekte Antwort geraten wird, soll er im falschen Klassendiagramm einen logischen Fehler und eine Schwäche hinsichtlich redundanter Attribute (aufgrund fehlender Vererbungsstruktur) auffinden und kennzeichnen. Hierzu ist der Proband im weiteren Verlauf der Aufgabe angehalten, die entsprechenden Bereiche im falschen Klassendiagramm mit Kreismarkierungen hervorzuheben.

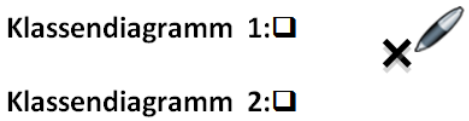


Abbildung 6.4.: Multiple Choice Auswahl eines von zwei Klassendiagrammen

Klassendiagramm 1:

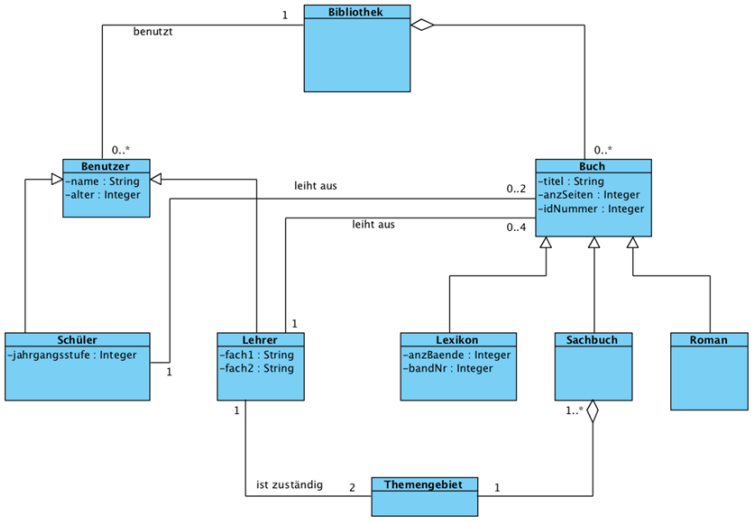


Abbildung 6.5.: Korrektes Klassendiagramm

Klassendiagramm 2:

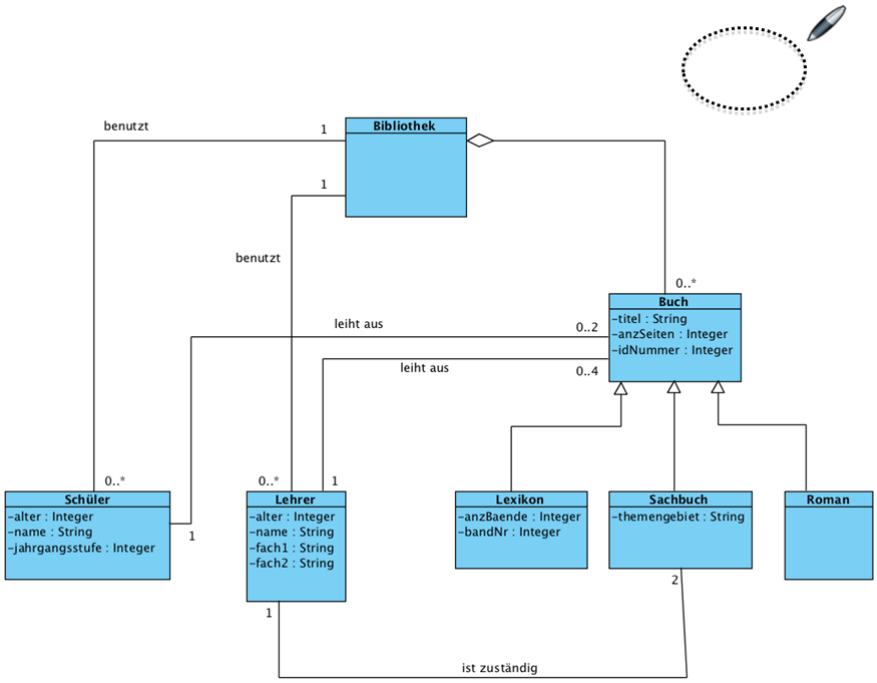


Abbildung 6.6.: Falsches Klassendiagramm

Aufgabe 6 - Aufgabe zur Designphase

Die Aufgabe 6 enthält Items zur Überprüfung der Kompetenzkomponente *K1.3.4 Design* und deren Teilbereiche *K1.3.4.3* am Beispiel des Zustandsdiagramms (*K1.3.4.3.2*).

> *K1.3.4.3 Die Lernenden sind in der Lage, relevante statische und dynamische UML-Diagramme mit implementierungsspezifischen Details zu entwickeln. Hierdurch entsteht ein entwurfsspezifisches Modell, das in Quellcode einer objektorientierten Hochsprache überführt werden kann.*
>
> *K1.3.4.3.2 Zustandsdiagramm: Die Lernenden sind in der Lage, Zustandsdiagramme zu entwickeln.*

Zu Beginn der Aufgabe findet der Proband wiederum eine Aufgabenbeschreibung (unter Verwendung der Operatorliste) gefolgt von dem Stimulusmaterial zur Verankerung des Kontexts der Aufgabe vor.

Aufgabe 6
Ergänzen Sie ausgehend von der unten aufgeführten Funktionsbeschreibung eines Festplatten-Rekorders das Zustandsdiagramm: Ergänzen Sie hierbei die fehlenden Zustandsübergänge. *Festplatten-Rekorder*: Das Gerät befindet sich nach dem Einschalten im Hauptmenü. Mittels der TV-Taste gelangt man in den TV-Modus, in dem das aktuelle Fernsehbild dargestellt wird. Betätigt man die Record-Taste, wechselt das Gerät in den Aufnahme-Modus und zeichnet das aktuelle Fernsehprogramm auf. Betätigt man in diesem Zustand die Stop-Taste wird die Aufnahme beendet und das Gerät wechselt wieder in den TV-Modus. Durch Betätigung der Pause-Taste innerhalb des TV-Modus wird der Timeshift-Modus aktiviert. Hierbei wird die aktuelle Fernsehsendung pausiert und ab diesem Zeitpunkt aufgenommen. Durch nochmaliges Drücken der Pause-Taste wird das Fernsehprogramm von der zuvor pausierten Position fortgesetzt. Drückt man die Stop-Taste wechselt der Festplatten-Rekorder wieder in den TV-Modus und spielt das TV-Programm ohne zeitlichen Versatz ab. Drückt man innerhalb des Hauptmenüs die Archiv-Taste, wechselt das Gerät in den Archiv-Modus. Hier kann durch Betätigung der Play-Taste eine ausgewählte (zuvor aufgenommene) Sendung abgespielt werden (das Gerät wechselt in den Abspielen-Modus). Mit Hilfe der Stop-Taste gelangt man wiederum in den Archiv-Modus. Sowohl im TV- als auch im Archiv-Modus gelangt man durch Drücken der Menü Taste ins Hauptmenü. (...)

Jetzt ist der Proband aufgefordert, innerhalb der folgenden Abbildung (unvollständiges Zustandsdiagramm) die fehlenden Zustandsübergänge zu ergänzen, sodass die in der Beschreibung *Festplatten-Rekorder* beschriebene Programmlogik korrekt modelliert wird. Hierbei handelt es sich wiederum um eine Aufgabe mit lebensweltnahem Stimulusmaterial und offenem Antwortformat.

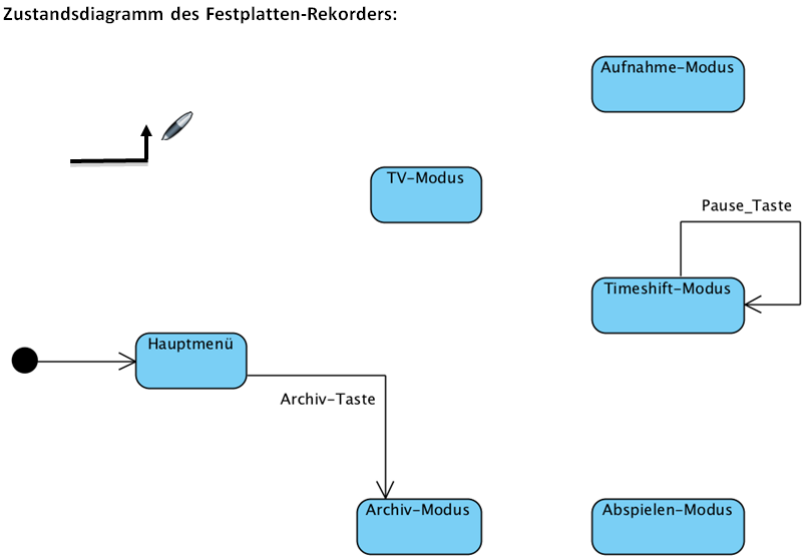


Abbildung 6.7.: Unvollständiges State-Chart

Aufgabe 10 - Aufgabe zur Testphase & zum iterativen Vorgehen

Die Aufgabe 10 des Kompetenzmessinstruments enthält mit dem dargebotenen Screenshot eines Reisebuchungssystem ein lebensweltnahes Stimulusmaterial und umfasst sowohl Items mit Multiple Choice Antwortformaten (Teilaufgabe 10a und 10c) als auch Items mit offenen Antwortformaten (Teilaufgabe 10b)). Die Zielsetzung der Aufgabe 10 a) besteht in der Überprüfung des von *K1.3.6 Test* und *K1.3.7 Iteratives Vorgehen*. Hierbei werden insbesondere die Kompetenzprofile *K1.3.6.1* und *K1.3.7.1* fokussiert.

> *K1.3.6.1 Die Lernenden sind in der Lage, ein bestehendes Softwaresystem systematisch zu testen. Hierbei besteht die Zielsetzung unter anderem darin, zu überprüfen, ob die zuvor spezifizierten funktionalen Anforderungen erfolgreich umgesetzt wurden.*
>
> *K1.3.7.1 Die Lernenden sind in der Lage zu beurteilen, ob ein erneutes Durchlaufen einer bereits absolvierten Phase des SWE-Prozesses erforderlich ist; sie*

können abhängig von den auftretenden Problemen in der aktuellen Phase eine sinnvolle vorherige Phase auswählen, die es erneut zu durchlaufen gilt.

Innerhalb der Teilaufgabe 10a) werden dem Probanden allgemeine Fragen zur Testphase unter Verwendung von Items im Multiple Choice Format gestellt (Items i) - iv)). Hierbei geht es darum, allgemeine Fakten zur Testphase, deren Bedeutung im Kontext des SWE-Prozesses und bestimmte Begriffe, wie z.B. der *Robustheit* abzufragen. Das Item v) thematisiert die Testphase im sozialen Kontext. Diesbezüglich sollen die Probanden Situationen aus Ihrer Erfahrungswelt nennen, bei denen ein sorgfältiger SW-Test von hoher Bedeutung ist. Die Formulierung dieses Items macht wiederum deutlich, dass die Kompetenzdimensionen *K1* und *K4* eng miteinander verflochten sind. Analog zur Testphase und dem Kompetenzprofil *K1.3.6.1* hat dieses auch die Zielsetzung den Kompetenzbereich *K4.2* abzudecken.

Aufgabe 10a		
Entscheiden Sie, ob die folgenden Aussagen wahr sind.	ja	nein
i) Im Rahmen der Testphase wird ausschließlich überprüft, ob der Auftraggeber mit dem für ihn entwickelten Softwaresystem zurechtkommt.	[ ]	[ ]
ii) In der Testphase wird überprüft, ob sämtliche funktionalen Anforderungen aus der Anforderungsanalyse innerhalb des Softwaresystems umgesetzt wurden.	[ ]	[ ]
iii) Es kann sinnvoll sein im Rahmen der Testphase einen Rückgriff auf die bereits abgeschlossene Anforderungsdefinition zu machen.	[ ]	[ ]
iv) Wenn man eine Software innerhalb der Testphase auf Robustheit überprüft, testet man wie zuverlässig das System über einen längeren Zeitraum läuft.	[ ]	[ ]
v) Es gibt bestimmte sicherheitskritische Bereiche, in denen Softwaresysteme zur Unterstützung eingesetzt werden. Hierbei ist es von enormer Wichtigkeit, dass die jeweilige Software auf Herz und Nieren getestet wird. Nennen Sie mindestens zwei solcher Bereiche, in denen ein sorgfältiger Softwaretest vor dem Einsatz der Software außerordentlich wichtig (vielleicht sogar lebenswichtig) ist. _____ _____ _____ _____ _____ _____		

Innerhalb der Teilaufgabe b) werden insbesondere die Kompetenzprofile *K1.3.6.1.1* und *K1.3.6.1.3* adressiert. Implizit wird außerdem der Kompetenzbereich *K1.3.7 Iteratives Vorgehen* angesprochen.

> *K1.3.6.1.1 Die Lernenden sind in der Lage, zu Beginn der* Testphase *einen geeigneten Testplan zu entwickeln.*
>
> *K1.3.6.1.3 Die Lernenden sind in der Lage, Testfälle zu ermitteln (Extremfälle und unerwartete Eingabedaten erzeugen) oder zu entwickeln; sie können diese zum Test verwenden und die daraus resultierenden Ausgaben protokollieren.*

K1.3.7.1 Die Lernenden sind in der Lage zu beurteilen, ob ein erneutes Durchlaufen einer bereits absolvierten Phase des SWE-Prozesses erforderlich ist; sie können abhängig von den auftretenden Problemen in der aktuellen Phase eine sinnvolle vorherige Phase auswählen, die es erneut zu durchlaufen gilt.

Zu Beginn der Aufgabe wird der Proband innerhalb der Aufgabenbeschreibung aufgefordert, anhand einer Illustration einer bestehenden Webapplikation zur Reisebuchung und einer Liste von funktionalen Anforderungen einen geeigneten Testplan zu entwickeln und konkrete Testfälle zu spezifizieren. Dem Probanden wird hier bewusst ein Rückgriff von der *Testphase* auf die *Anforderungsanalyse* vorgegeben, indem je funktionaler Anforderung ein konkreter Testfall entwickelt werden soll. Diesen gilt es in eine vorgefertigte Tabelle einzutragen. Um diesen geforderten Aufgabenschritt deutlich zu illustrieren, wird der entsprechende Testfall für die Anforderung 1 vorgegeben.

Aufgabe 10b
Entwickeln Sie anhand des unten dargestellten Screenshots einer Webapplikation zur Reisebuchung und anhand des Ausschnitts der Anforderungsdefinition einen geeigneten Testplan. Gehen Sie dabei folgendermaßen vor: i) Überprüfen Sie, ob sämtliche funktionalen Anforderungen an die Software umgesetzt wurden, indem Sie für jede Anforderung jeweils einen Testfall entwickeln. Tragen Sie diese Testfälle in Tabelle 1 ein. Anforderungsdefinition Reisebuchungssystem: • Anforderung 1: Benutzer kann Pauschalreisen suchen, indem er Reiseziel, Abflughafen, Abflugdatum, Rückflugdatum (muss mindestens zwei Tage hinter dem Abflugdatum terminiert sein), Anzahl Erwachsener (mindestens einer), Anzahl Kinder, Verpflegungsarten (mindestens eine) sowie einen Zimmertyp auswählt. • Anforderung 2: Benutzer kann optional die Hotelkategorie (Anzahl Sterne) mit in die Suche einbeziehen. • Anforderung 3: Benutzer kann auch nur den Hinflug buchen. Hierbei muss keine Eingabe in die Elemente der rechten Spalte gemacht werden. ii) Überprüfen Sie die *Robustheit* der Anwendung, indem Sie nochmals den Auszug der Anforderungsdefinition betrachten und drei unerwartete Testfälle entwickeln, die die Anwendung zum Absturz bringen könnten. Ergänzen Sie diese Testfälle in Tabelle 2.

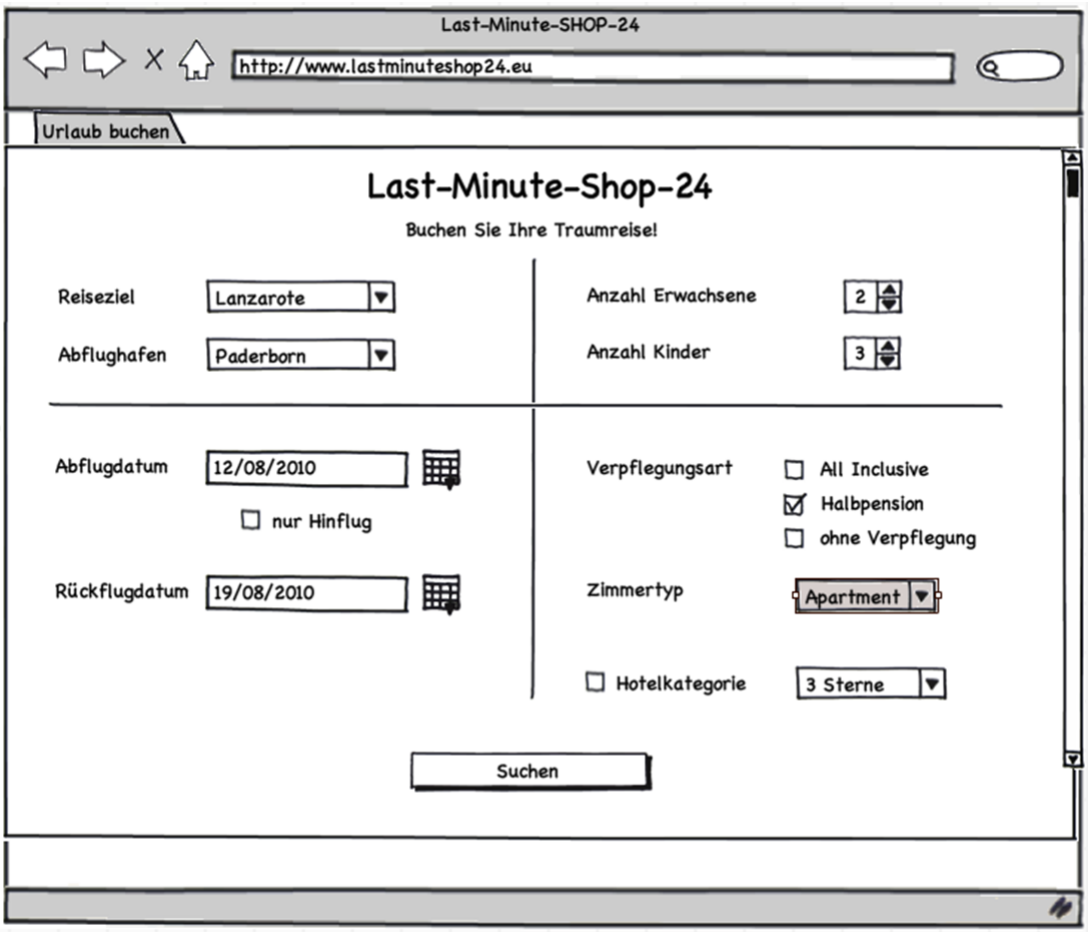


Abbildung 6.8.: Mockup Reisebuchungssystem

Testfall: Anforderung 1	Testfall: Anforderung 2	Testfall: Anforderung 3
Reiseziel: Lanzarote	Reiseziel:	Reiseziel:
Abflughafen: Paderborn	Abflughafen:	Abflughafen:
Abflugdatum: 01.08.2010	Abflugdatum:	Abflugdatum:
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ❑
Rückflugdatum: 08.08.2010	Rückflugdatum:	Rückflugdatum:
Anzahl Erwachsene: 2	Anzahl Erwachsene:	Anzahl Erwachsene:
Anzahl Kinder: 1	Anzahl Kinder:	Anzahl Kinder:
Verpflegungsart: AI ❑; VP ■; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑
Zimmertyp: Apartment	Zimmertyp:	Zimmertyp:
Hotelkategorie: ❑	Hotelkategorie: ❑	Hotelkategorie: ❑

Abbildung 6.9.: Testfälle zur Anforderungsdefinition

Die Teilaufgabe ii) zielt ebenfalls auf die Spezifikation von Testfällen ab. Hierbei soll der Proband die Robustheit der Webanwendung testen. Analog zu Teilaufgabe i) wird ein exemplarischer Testfall in Tabelle 2 vorgegeben.

Testfall: Fehleingabe 1	**Testfall: Fehleingabe 2**	**Testfall: Fehleingabe 3**
Reiseziel: Lanzarote	Reiseziel:	Reiseziel:
Abflughafen: Paderborn	Abflughafen:	Abflughafen:
Abflugdatum: 01.08.2010	Abflugdatum:	Abflugdatum:
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ❑
Rückflugdatum: 25.07.2010	Rückflugdatum:	Rückflugdatum:
Anzahl Erwachsene: 2	Anzahl Erwachsene:	Anzahl Erwachsene:
Anzahl Kinder: 1	Anzahl Kinder:	Anzahl Kinder:
Verpflegungsart: AI ■; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑
Zimmertyp: Apartment	Zimmertyp:	Zimmertyp:
Hotelkategorie: ■ (3 Sterne)	Hotelkategorie: ❑	Hotelkategorie: ❑

Abbildung 6.10.: Testfälle zur Robustheit

Die Teilaufgabe c) und die enthaltenen Items wurden mit der Zielsetzung entwickelt, Teilaspekte des Kompetenzbereichs *K1.3.6.1* und *K1.3.6.1.1* anzusprechen.

> *K1.3.6.1 Die Lernenden sind in der Lage, ein bestehendes Softwaresystem systematisch zu testen. Hierbei besteht die Zielsetzung unter anderem darin, zu überprüfen, ob die zuvor spezifizierten funktionalen Anforderungen erfolgreich umgesetzt wurden.*
>
> *K1.3.6.1.1 Die Lernenden sind in der Lage, zu Beginn der Testphase einen geeigneten Testplan zu entwickeln.*

Aufgabe 10c
i) Sie entwickeln eine Webseite für ein Reisebüro und befinden sich nach Abschluss der Implementierung in der Testphase. Um die *Robustheit* Ihres Softwareprodukts zu verbessern, sollen Betatester in den Prozess mit einbezogen werden. Welche Personen würden Sie in die Testphase mit einbeziehen? (Mehrfachnennungen möglich) [ ] Die Entwickler des Reisebuchungssystem [ ] Erfahrene Benutzer anderer Reisebuchungssysteme [ ] Benutzer, die Grundkenntnisse in der Benutzung des Internets haben [ ] Grundschüler, die gerade das Lesen gelernt haben
ii) Viele Betatester haben über Abstürze der Webseite berichtet. Wie gehen Sie vor, um die Eingaben in das System, die zum Absturz geführt haben, herauszufinden? Wie ermitteln Sie, was der Benutzer mit seinen Eingaben (die zum Absturz geführt haben) bezwecken wollte? _____ _____ _____ _____ _____

Zu Beginn der Teilaufgabe c) wird erneut ein Stimulusmaterial in Form einer textuellen Darstellung des oben illustrierten Aufgabenszenarios dargeboten. Dieses beschreibt das oben aufgeführte Aufgabenszenario zu einem späteren Zeitpunkt des SWE-Prozesses, nach Abschluss der Implementierungsphase. In Form eines Multiple Choice Aufgabenformats wird in Item i) abgefragt, welche Personen sinnvoll bei der Durchführung der Testphase als Betatester eingesetzt werden könnten. Dies stellt einen wichtigen Bereich der Testplanung dar und kann somit dem abgefragten Kompetenzprofil *K1.3.6.1.1* zugeordnet werden.
Innerhalb der Teilaufgabe ii) wird ein weiteres Stimulusmaterial angegeben. Hierbei wird ein Fehlerszenario beschrieben, in dem die Nutzer des Reisebuchungssystems über Abstürze berichten. In diesem Zusammenhang wird der Proband nach seiner Vorgehensweise gefragt, um die Fehlerursache zu reproduzieren, die zum Absturz geführt haben könnte. Die Teilaufgabe ii) steht demnach auch repräsentativ für eine Aufgabe mit einleitendem Stimulusmaterial mit lebensweltnaher Szenariobeschreibung und einem Item mit offenem Antwortformat.

Zusammenfassung

Innerhalb dieses Teilkapitels bestand die Zielsetzung, den Prozess der Aufgabenentwicklung zu beschreiben und zu veranschaulichen. Dementsprechend wurde auf Grundlage von Kompetenzprofilen, die im Kapitel 5.6 definiert wurden, zugehörige Items generiert und zu Aufgaben kombiniert. Die Testentwicklung wurde in Anlehnung an objektive Testverfahren (*Situational Judgement Tests (SJT)*) entwickelt. Es wurden sowohl Erfahrungen zur Testentwicklung bei internationalen Vergleichsstudien als auch von Testentwicklungen im Bereich der Arbeits- und Organisationspsychologie mit einbezogen. Im Verlauf des Kapitels wurden repräsentative Aufgaben vorgestellt und deren Aufbau und Entwicklungsprozess dargelegt. Diese Aufgaben enthalten jeweils eine Aufgabenbeschreibung, ein lebensweltnahes Stimulusmaterial und die entsprechenden Items zur Überprüfung der Kompetenzbereiche des Strukturmodells.
Um das entwickelte Messinstrument zu erproben wird im folgenden Teilkapitel die Konzeption eines Lehr-/Lernarrangements zur Förderung von Modellierungskompetenzen beschrieben. Mit dem Ziel, das Instrument im Rahmen eines Unterrichtsversuchs zur Förderung von Kompetenzen im Bereich des informatischen Modellierens zu erproben, wird im folgenden Unterkapitel die theoretische und praktische Entwicklung des Lehr-/Lernarrangements beschrieben.

6.2. Entwicklung eines Lehr/Lernarrangements zur Erprobung des Messinstruments

Zunächst soll die fachdidaktische Grundlage erläutert werden, auf der die zu entwickelnde Lerneinheit basiert. Dementsprechend werden verschiedene fachdidaktische Ansätze dargestellt. Hierbei soll angeführt werden, inwiefern die unterschiedlichen didaktischen Ansätze zur Entwicklung der Unterrichtsreihe und somit zur Evaluation des Messinstrumentariums beigetragen haben. Der Fokus liegt hierbei auf dem Ansatz des *Informatik Lernlabors* der systemorientierten Didaktik der Informatik nach Magenheim.
Im Folgenden wird die theoretische Grundlage zur Konzeption der Unterrichtsreihe für die Erprobung des Kompetenzmessinstruments erörtert. Hierbei werden der Ansatz der systemorientierten Didaktik der Informatik nach Magenheimh sowie die theoretische Einbettung der Unterrichtsreihe im Kontext des *Informatik Lernlabors* dargestellt.

6.2.1. Informatiksysteme in didaktischem Kontext

Die Vernetzung von Informatiksystemen und deren Wirkprinzipien werden seit Anfang der 90er Jahre als Orientierungspunkt für den Informatikunterricht gefordert [Stechert 2009, S. 25]. Im weiteren Verlauf sollen verschiedene Ansätze (ohne Anspruch auf Vollständigkeit) für Informatiksysteme im didaktischen Kontext dargestellt werden. Es sollen jeweils die Zusammenhänge und Bezugspunkte der verschiedenen didaktischen Ansätze zur systemorientierten Didaktik, die als theoretische Grundlage der Unterrichtsreihe fungiert, geknüpft werden.
Hubwieser und Broy entwickeln den *informationszentrierten Ansatz* als möglichen Ausgangspunkt zur Legitimation des Informatikunterrichts an Gymnasien [Hubwieser und Broy 1996].
Informatiksysteme weisen nach diesem Ansatz folgende Eigenschaften auf [Hubwieser 2007, S. 44]:

- automatische Verarbeitung von Daten,
- Vernetzung,
- Interaktion mit menschlichen Benutzern.

Für den Einsatz und die Auseinandersetzung mit Informatiksystemen in der Schule sieht Hubwieser die folgenden Schwerpunkte [Hubwieser 2007, S. 44]:
Zunächst betont er den Nutzen von Informatiksystemen bei der Unterstützung von Lernvorgängen jeglicher Art. Dies macht nach Hubwieser einen der wichtigsten und zeitaufwändigsten Teile der oben genannten Schwerpunkte aus. Als Beispiel wird in diesem Zusammenhang die Beschaffung von Informationen aus dem Internet oder die Verwendung von Lernsoftware jeglicher Art genannt.
Als weiterer Punkt wird die Schulung von Bedienerfertigkeiten für konkrete Systeme genannt. Hier werden beispielhaft Kurse zur Einführung in die Handhabung eines speziellen Textverarbeitungssystems genannt, die häufig sogar nach dem Produkt benannt sind. Solche Schulungen vermitteln allerdings häufig nur sehr spezielle Kenntnisse über die jeweilige Software und dessen Benutzeroberfläche. Ein positiver Lerntransfer ist mit solchem produktspezifischen Wissen kaum möglich.
Die Vermittlung allgemeiner und langlebiger Grundlagen der Informationstechnik wird als weitere wichtige Zielsetzung für den Informatikunterricht angeführt. Diese letzte Eigenschaft macht die eigentliche Charakteristik des Informatikunterrichts aus.

> „Neben der Unterstützung von Lernprozessen oder der Bedienerschulung können im Unterricht auch Prinzipien, Konzepte und Strategien zur Planung,

> Konstruktion, Beschreibung und Bewertung abstrakter Informatiksysteme thematisiert werden. Vor allem um diesen Informatikunterricht im eigentlichen Sinne geht es [...]. Dazu gehört auch die Behandlung von Anwendungen informatischer Prinzipien [...] [Hubwieser 2007, S. 48]."

Die drei genannten Schwerpunkte in der unterrichtlichen Auseinandersetzung mit Informatiksystemen sind eng miteinander verzahnt. Hubwieser sieht die Verknüpfung dieser Bereiche als wesentliche Eigenschaft eines guten Informatikunterrichts.

> "Die Vermittlung grundlegender Konzepte der Informatik ist ohne die Arbeit am Rechner nach Hubwieser eine abschreckend abstrakte Veranstaltung. Dies ist aus seiner Sicht zur Veranschaulichung der Lerninhalte und zur Motivierung der Lernenden absolut unerlässlich. Im Umkehrschluss setzt das wiederum einige Fertigkeiten in der Bedienung von Hard- und Software voraus, so dass man auch hier nicht ohne ein gewisses Maß an Bedienerschulung auskommt [Hubwieser 2007, S. 48]."

Hubwieser sieht folgende methodische Prinzipien bei der Gestaltung einer Unterrichtsreihe für den Informatikunterricht [Hubwieser 2007, S. 68]:

1. *Problemorientierung*
   Unterrichtsinhalte sollten anhand eines Problems aus der Erfahrungswelt der Schüler behandelt werden. Eine strikte Problemorientierung kann den Informatikunterricht davor bewahren, dass dieser den Charakter einer Produktschulung oder eines Programmierkurses hat.

2. *Modellbildung und Simulation*
   Die Modellbildung und die Simulation sollen als durchgängiges Prinzip in die Planung des Informatikunterrichts mit einfließen.

   > „Unser Informatikunterricht beschäftigt sich nicht nur mit Modellbildung und Simulation, unser Unterricht besteht im Wesentlichen aus Modellbildung und Simulation, [...] [Hubwieser 2007, S. 69ff]."

Wie zuvor beschrieben sieht *Hubwieser* analog zum systemorientierten Ansatz nach Magenheim die Modellierung als ein wichtiges Prinzip des Informatikunterrichts. Darüber hinaus können anhand des *informationszentrierten Ansatzes* Hinweise zur Gestaltung der Unterrichtsreihe abgeleitet werden. Diese sind zum einen die Problemorientierung, die es bei der Unterrichtsreihe zu bewerkstelligen gilt und zum anderen, dass die Modellierung und Simulation als Unterrichtsprinzip eine hohe Berücksichtigung finden sollte.

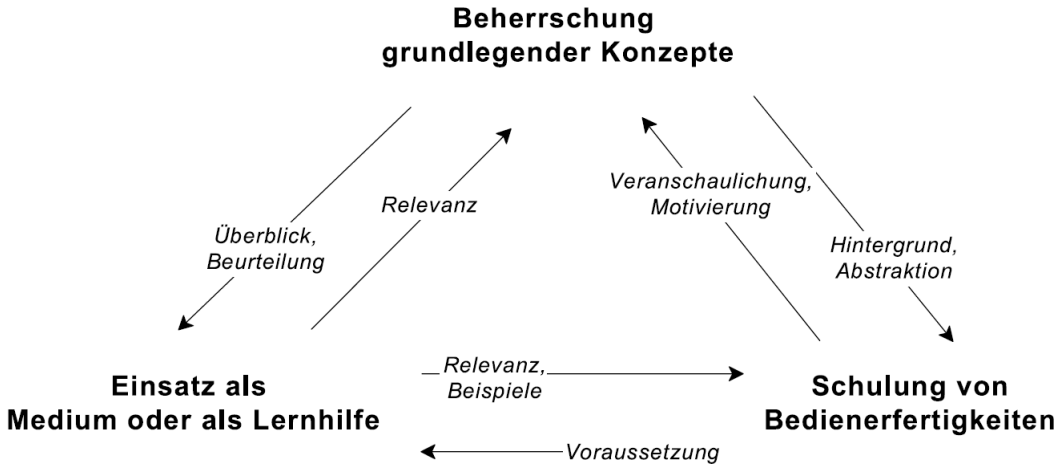


Abbildung 6.11.: Verflechtung von Unterrichtshilfen, Bedienerschulung und Vermittlung grundlegender Konzepte [Hubwieser 2007, S. 49]

Bezüglich der Unterrichtsreihe zur Evaluation des Messinstruments soll zur Problemorientierung der Unterrichtsreihe das Fallbeispiel *Kommissionierstation* verwendet werden, welches im weiteren Verlauf des Kapitels ausführlicher dargestellt wird. Ferner erhalten die Schüler die Gelegenheit die eigens entwickelten Modelle in Form von *LEGO Mindstorms* Robotern zu simulieren.
Schubert und Stechert charakterisieren Informatiksysteme durch mögliche Perspektiven auf ein solches System. Auf Grundlage von Erkenntnissen einer Expertengruppe der *Association for Computing Machinery (ACM)* zur Erstellung von Curricula für die Hochschulinformatik, werden sieben Kategorien der Informatik abgeleitet [Denning 2003]. Schubert und Stechert sehen ausgehend von diesen Kategorien unterschiedliche Sichten auf ein Informatiksystem als relevant an, die Lernende auf den Lerngegenstand Informatiksystem einnehmen können. Die Kategorien der Informatik nach Denning lassen sich sinnvoll mit dem Sichtenkonzept kombinieren [Stechert 2009, S. 29].

- computation
  Bedeutung von Berechnungen und Fragen der Berechenbarkeit:
  Automaten, formale Sprachen, Turingmaschinen, Universalität, komplexitätstheoretische Fragestellungen, Übersetzung, die technisch-physikalische Realisierung von Informatiksystemen

- communication
  Zuverlässige Datenübertragung: Informatiksystem als Medium, speziell als Nachrichtensystem im Shannon'schen Sinne eines Kanalmodells mit Kodierung, Kanal-

kapazität, Rauschunterdrückung, Datenkompression, Kryptographie, rekonfigurierbarer Paketvermittlung und Ende-zu-Ende Fehlerbehandlung

- coordination
  Kooperation zwischen vernetzten Entitäten: zwischen Menschen deren Arbeitsabläufe durch Informatiksysteme unterstützt werden (Workflows), Eingabe- und Ausgabeverhalten der Informatiksysteme sowie Antwortzeit. Synchronisation zwischen Informatiksystemen mit Race-conditions und Deadlocks, Serialisierbarkeit und atomaren Aktionen

- automation
  Fragen zu Grenzen der Simulation kognitiver Prozesse: philosophische Betrachtungen zur Automatisierung, zu Expertise und Expertensystemen, Verbesserung von (künstlicher) Intelligenz, Turingtest, Bedeutung des Maschinenlernens mittels evolutionärer Algorithmen, Bionik

- recollection
  Speicherhierarchien, Lokalität von Referenzen, Caching, Adressbereich und Abbildung, Namenskonventionen, Suche sowie Retrievaltechniken durch Name oder Inhalt

- evaluation
  Leistungsvoraussagen und Kapazitätsplanung: Sättigung und Flaschenhälse in Netzen

- design
  Entwicklung qualitativ hochwertiger Informatiksysteme: Ebenenmodell des Rechners, Schichtenarchitektur des Internets, Modularisierung, Geheimnisprinzip und Abstraktion

Diese Sichten auf ein Informatiksystem sollen lernförderlich miteinander kombiniert werden. Als Grundlage zur Verknüpfung der verschiedenen Sichten werden die folgenden drei Charakteristika in Anlehnung an Claus und Schwill vorgeschlagen [Claus und Schwill 2006].

1. nach außen sichtbares Verhalten
2. innere Struktur
3. Entwicklung einer konkreten Realisierung

Da sich die Sichten auf ein Informatiksystem stark überschneiden, ist es sinnvoll eine Bündelung der Lerninhalte und Fragestellungen zu den oben genannten Kategorien anhand dieser Charakteristika vorzunehmen.
Bei der Behandlung von Informatiksystemen (z.B. bei der Modellierung eines Softwareprojekts) ist es sinnvoll, den Gegenstandsbereich aus unterschiedlichen Perspektiven zu betrachten. Dementsprechend fordert Schubert in Übereinstimmung mit Brinda, dass der zu explorierende Lerngegenstand in unterschiedlichen, interaktiv erfahrbaren und synchronisierten Sichten darzustellen ist [Brinda 2004, S. 52].
Diese Forderung stellt einen besonderen Verknüpfungspunkt zwischen dem Siegener und dem Paderborner Ansatz dar. Wie im weiteren Verlauf der Arbeit genauer erläutert wird, sieht das Konzept des *Informatik Lernlabors* auch einen expliziten Perspektivwechsel durch unterschiedliche mediale Repräsentationsformen eines Informatiksystems vor. Beispielsweise kann ein Warenwirtschaftssystem eines Kiosks *real*, *auf Modellebene* oder *auf Quellcodeebene* medial repräsentiert werden. Hierbei macht es durchaus Sinn während der einzelnen SWE-Phasen unterschiedliche und sich wechselnde Perspektiven auf ein Informatiksystem einzunehmen. Diese medialen Repräsentationsformen können auch als unterschiedliche Sichten auf ein Informatiksystem interpretiert werden.
Ausgangspunkt der systemorientierten Didaktik nach Magenheim ist ein divergentes Verständnis von Informatiksystemen und deren Bedeutung für den Informatikunterricht. Um den Unterschied deutlich zu machen, betrachten wir die Definition des Begriffes Informatiksystem nach Rüdeger Baumann.

> „Unter einem Informatiksystem versteht man ein verteiltes, heterogenes, technisches System, das Wissen unterschiedlichster Art und Herkunft repräsentiert, diese Wissensrepräsentation in Gestalt von Daten und Programmen verarbeitet und den Benutzern in geeigneter Form zur Verfügung stellt [Baumann 1996, S. 164]."

Nach Magenheim beinhaltet dieses Verständnis von einem Informatiksystem einen zu starken Fokus auf das Produkt Software und dessen formale Dimension. Es ergibt sich somit eine zu theoretische Zugangsweise zu Informatiksystemen und deren Behandlung im Informatikunterricht. Inhaltlich zentrale Bestandteile eines solchen Unterrichts sind somit vorrangig Datenstrukturen und Algorithmen sowie Kommunikationsprotokolle und sukzessives Erlernen einer bestimmten Sprachsyntax.
Nach Magenheim vernachlässigt ein solcher Ansatz den Entwicklungsgang sowie die Anwendung von Software und die damit einhergehende soziale Interaktion mit Informatiksystemen. Eine von den Lehrplänen stets verlangte Behandlung von gesellschaftlichen

Auswirkungen von Informatiksystemen im Unterricht, kann infolgedessen nur in sehr geringem Umfang durchgeführt werden. Baumanns Ansatz bewirkt somit eine inhaltliche Benachteiligung dieses essentiellen Bestandteils des Informatikunterrichts.
Der Ansatz der systemorientierten Didaktik der Informatik setzt andere Maßstäbe bezüglich inhaltlicher und methodischer Gestaltung von Informatikunterricht. Ausgangspunkt des Unterrichts ist hierbei die Modellierung von Software und Softwaresystemen aus soziotechnischer Perspektive. Das Handlungssystem und die Dynamik zwischen Mensch und Maschine stehen demzufolge im Vordergrund. Formale Operationen und informatische Prinzipien werden im Gegensatz zu anderen didaktischen Ansätzen als Bestandteile des SWE-Prozesses verstanden. In direkter Beziehung dazu stehen Fragestellungen bezüglich der Anwendung und den Auswirkungen von Informatiksystemen. Gesellschaftliche Auswirkungen werden in diesem theoretischen Ansatz nicht mehr in Form von unterrichtlichen Abstechern behandelt, sondern als Ausgangspunkt für Design- und Entwurfsentscheidungen im Softwareentwurf. Es soll dem Lernenden somit deutlich gemacht werden, dass der Erwerb von sozialen Handlungs- und Kommunikationskompetenzen von fundamentaler Bedeutung für die Entwicklung von Software ist.
Magenheims Ansatz hat somit die Intention, dass Modellierung und Systemgestaltung als zentraler Bezugspunkt für Anwendungsfragen und Fragestellungen der Auswirkung von Informatiksystemen angesehen werden und somit den inhaltlichen Schwerpunkt von Informatikunterricht ausmachen.
Innerhalb der systemorientierten Didaktik der Informatik Paderborn wurde in mehreren Lehrveranstaltungen mit unterrichtspraktischem Bezug die Erfahrung gemacht, dass die Verwendung von überwiegend selbstgesteuerten und konstruktivistisch orientierten Lehr-/Lernarrangements [Cognition and Technology Group at Vanderbilt 1994], [Collins 1989], [Spiro und Feltovich 1992] Modellierungskompetenz wirkungsvoller fördern als solche Arrangements, bei denen rezeptiv vermittelnde Instruktionsstrategien zum Einsatz kommen. Auf dieser Grundlage wurde innerhalb des *Informatik Lernlabors* [Magenheim 2003a] die Lernumgebung *Kommissionierstation* entwickelt. Diese wurde für das Projekt *MoKoM* weiterentwickelt und didaktisch aufbereitet. Die Besonderheit dieser Lernumgebung besteht darin, dass die Schüler zunächst eine Explorationsphase durchlaufen, in der sie mit einem bestehenden komplexen Informatiksystem (*Kommissionierstation*) konfrontiert werden (in diesem Zusammenhang werden Systemanwendung und Systemverständnis gefördert). Anschließend gilt es, das Informatiksystem im Rahmen einer Re-Engineering-Phase zu dekonstruieren, zu analysieren und weiterzuentwickeln. In diesem Beispiel hat es sich als hilfreich erwiesen, dass sich die Schüler zunächst anhand der Entwurfsmuster *Model-View-Controller* und des *Observer-Patterns* die Systemfunktio-

nalität des *LEGO Mindstorms* Modells erschließen, um in einem nächsten Schritt die jeweiligen *UML-Diagrammtypen* und den Quellcode des Systems in strukturierter Form analysieren zu können. Um dieses lernerzentrierte explorative Vorgehen zu unterstützen, enthält die Lernumgebung *Kommissionierstation* spezifische Medien, die das Informatiksystem in Form von Learning Objects [Standards 2001] mit unterschiedlichen medialen Codierungsarten und Abstraktionsebenen repräsentieren [Tulodziecki und Herzig 2002], [Magenheim und Scheel 2004]. Diese Kategorisierung bot Hilfestellung bei der Auswahl von *Lernobjekten*, die im Rahmen der Unterrichtsreihe zum Einsatz kamen. Mit Hilfe der bereitgestellten medialen Repräsentationen der Lernumgebung wurden lernerzentrierte, explorative Lernprozesse zum Modellieren von Informatiksystemen ermöglicht. Hierbei bestand die Zielsetzung, etablierte Vorgehensmodelle des Softwareentwurfs zu vermitteln und mit Hilfe eines Beispiels aus der Lebenswelt der Schüler die Anwendung, den Transfer und die Bewertung von relevanten Modellierungstechniken zu vermitteln. Da neben der Förderung von kognitiven Fähigkeiten und Fertigkeiten zur informatischen Modellierung auch die Kooperation der Schüler sowie deren Kommunikation untereinander angeregt wurden, bot sich die Gelegenheit, die Messinstrumente im Rahmen dieser Lernumgebung (April bis Juni 2010) in einem Grundkurs der Jgst. 12 am Paderborner Pelizaeus Gymnasium einzusetzen. Auf Grundlage dieser Unterrichtsreihe konnte das entwickelte Teilinstrument für informatische Modellierungskompetenz und nicht-kognitive Kompetenzen erprobt werden.

Dekonstruktion als Konzept der systemorientierten Didaktik

Vom Standpunkt der systemorientierten Didaktik, kann Software als Text zur Beschreibung und Steuerung maschineller Betriebsamkeit und Interaktion zwischen Mensch und Maschine verstanden werden. Diese Repräsentationsform beinhaltet Entwurfs- und Designentscheidungen des Entwicklers.
In diesem Kontext versteht man den Begriff Dekonstruktion als eine interpretierende Annäherung an Software. Hierbei soll eine Sensibilisierung für Gestaltungsmöglichkeiten von Software und zudem eine erhöhte Aufmerksamkeit im Hinblick auf die Phasen der Softwareentwicklung geschaffen werden. Dekonstruktion versteht sich demzufolge nicht nur als Lesen von Software um die Syntax einer Programmiersprache zu erlernen, sondern als Vorgehensweise um Annahmen über Modell-, Entwurfs- und Designentscheidungen einer Softwareentwicklung zu erlangen. Die somit gewonnenen Annahmen sind allerdings nur hypothetischer Ausprägung und können nicht eindeutig belegt werden [Magenheim 2000].
Die Softwareentwicklung wird bei dem Vorgang der Dekonstruktion als Entscheidungs-

prozess angesehen, der von den Rahmenbedingungen abhängt unter denen die Software entstanden ist. Zu diesen Bedingungen zählen insbesondere die Interessen und Wünsche des Auftraggebers, die unterschiedliche gesellschaftliche Folgen nach sich ziehen können. Die Unterstreichung dieser Zusammenhänge sollte im Sinne des systemorientierten Ansatzes von fundamentaler Bedeutung für den Informatikunterricht sein.
Dekonstruktion kann zudem Einsicht in vielschichtige Betrachtungsweisen und Abstraktionsebenen eines Informatiksystems geben. Sie verschafft Einblicke in die Benutzeroberfläche und die dort vorhandenen interaktiven grafischen Komponenten, als Repräsentanten für Softwarefunktionalität.
Der Assemblercode ist eine weitere Abstraktionsebene des Informatiksystems. Er kann als Verbindung zur Hardwarearchitektur eines Rechners gesehen werden. Informatiksysteme erscheinen in dieser Betrachtungsweise als zeichenverarbeitende Maschinen, die Daten und Programme in Form von Zeichenketten verarbeiten. In diesem Zusammenhang versteht sich die Turingmaschine als theoretischer Kern des Informatiksystems, die formale, in Zeichenketten codierte Regeln befolgt [Magenheim 2000].
Die Dekonstruktion auf dieser Abstraktionsebene durchdringt nach Magenheim grundlegende Fragestellungen der theoretischen Informatik wie Berechenbarkeit, formale Sprachen und Grundelemente der Komplexitätstheorie [Magenheim 2000].

Dekonstruktion als Unterrichtsmethode im Informatikunterricht

Häufig stellt sich Informatikunterricht als monotoner Programmierkurs einer vorher festgelegten Programmiersprache dar. Die Aufgabenstellungen haben zumeist einen geringfügigen Grad an Komplexität und beziehen sich vorwiegend auf fundamentale Algorithmen. Der Bezug zur Mathematik ist hierbei charakteristisch.
Nach Magenheim ergeben sich in diesem Zusammenhang Defizite beim Einüben und Erlernen von Problemlösungsstrategien. Hierzu gehört die Fähigkeit, Probleme in Teilprobleme aufzuteilen und ein analytisches Abstraktionsdenken zu entwickeln.

> „Der für ein Unterrichtsfach an allgemein bildenden Schulen übliche Anspruch auf Vermittlung von Allgemeinbildung ist somit kaum zu leisten [Hampel et al. 1999, S. 1]."

Da die – in sich dynamische – Bezugswissenschaft des Informatikunterrichts eine ständige Weiterentwicklung erfährt, ist ein solches Unterrichtskonzept für einen wissenschaftspropädeutischen oder allgemein bildenden Unterricht nur bedingt einsatzfähig.
Der Ausgangspunkt der systemorientierten Didaktik hingegen, ist die Modellierung und Analyse von Informatiksystemen mittels Dekonstruktion von objektorientiertem *Java-*

Code. Dieser ist von hinreichender und dennoch unter didaktischen Gesichtspunkten reduzierter Komplexität.
Operationen, die die Dekonstruktion in diesem Zusammenhang charakterisieren, sind die Erkundung der Funktionalität des Informatiksystems und die Herleitung des Modells des Problembereichs. Darüber hinaus ist der Vergleich des Informatiksystems zum realen System für die Dekonstruktion kennzeichnend.
Auf diese Weise werden am Beispiel des Informatiksystems objektorientierte Verfahren erprobt und im Verlauf der Dekonstruktion auch einzelne Klassen und Objekte der Software analysiert. Die Wechselwirkung von Datenstrukturen, Methoden und Ereignissen wird somit erkennbar und zusätzlich deren Codierung in *Java* sukzessive erschlossen.
Die zuvor beschriebenen Teilziele können nach Magenheim insofern erreicht werden, als dass zu jedem Dekonstruktionsschritt Beispiele und Übungsaufgaben präsentiert und durchgeführt werden. Dies bewirkt zudem eine verbesserte Auffassungsgabe durch Transfer in einen anderen Wahrnehmungszusammenhang. Weiterhin wird eine verbesserte Einsicht in objektorientierte Sichtweisen und Modellbildung gewährleistet. Im weiteren Verlauf der Dekonstruktion sollten die Lernenden zudem einzelne Zusatzmodule entwickeln und in geeigneter Form in die Software integrieren.
Die Hauptzielsetzung der Dekonstruktion im Sinne der systemorientierten Didaktik besteht in den Kompetenzen, kleine Softwareprojekte zu modellieren und nach objektorientierten Paradigmen selbst zu entwickeln. Von essentieller Bedeutung ist ferner die Fähigkeit, dass sich die Lernenden über das reale sozio-technische Informatiksystem und den damit interagierenden Menschen bewusst werden [Hampel et al. 1999].

6.2.2. Theoretische Konzeption des Informatik Lernlabors

Das Informatik Lernlabor aus didaktisch-methodischer Perspektive

Der Entwurf und die Umsetzung des *Informatik Lernlabors* stellen den Versuch dar, die zuvor beschriebenen unterrichtsmethodischen Konzepte unter wissenschaftlicher Begleitung in die Praxis umzusetzen [Magenheim 2003a].
In der herkömmlichen informatischen Bildung kommen Tools, wie *Editoren*, *Debugger* und *Compiler* zum Einsatz. Die Durchführung des *Informatik Lernlabors* erfordert zudem weitere interaktive computerbasierte Medien. Im Technisch-organisatorischem Sinne lassen sich diese in Anlehnung an Keils Ansatz zu *primären*, *sekundären* und *tertiären Medienfunktionen* folgendermaßen unterscheiden [Keil-Slawik 2002] [Magenheim 2003b, S. 36]:

- *Cognitive Tools*

Diese ermöglichen die interaktive Gestaltung und Strukturierung von Dokumenten, Software und computerbasierten Medien. Sie stellen zudem Prozesse der Bearbeitung, Sicherung und Übertragung sicher.

- *Lernsoftware*
  Diese stellt dem Nutzer zusätzlich zur Inhaltspräsentation eine lerntheoretisch motivierte Schnittstelle zur Interaktion mit dem Informatiksystem zur Verfügung. Folglich bildet das Medium Lernsoftware Formen der Mediennutzung in sich ab.

- *Agents*
  Sie repräsentieren einen Medientyp, der Lernende und Nutzungsverhalten analysiert und sich infolgedessen dem Nutzer und seinen individuellen Bedürfnissen anpasst. *Agents* besitzen somit eine regelbasierte oder algorithmische Lernfähigkeit und werden auch als intelligente tutorielle Systeme bezeichnet. Sie spielen in didaktischem Kontext eine untergeordnete Rolle und sind auch bisher in der Konzeption des *Informatik Lernlabors* nicht zu finden.

Die Medientypen *Cognitive-Tools* und *Lernsoftware* sind dagegen wesentliche Bestandteile des *Informatik Lernlabors*. *Cognitive Tools* kommen in Form von grafischen Debuggern und integrierten Entwicklungsumgebungen zum Einsatz. Weiterhin finden sie als grafische Editoren Anwendung, die beispielsweise die interaktive Entwicklung von *Java-Quellcode* unterstützen. Die einzelnen Phasen der Systemmodellierung können infolgedessen interaktiv am Computer gestaltet und der Lerngruppe vorgestellt werden. Weiterhin ist eine Modifikation oder Revidierung nach Gruppendiskussionen unter geringem Aufwand möglich.
Der Einsatz von *Lernsoftware* findet in Form von sog. *Lernobjekten* (learning objects) statt [Standards 2001]. Diese stellen in sich abgeschlossene, multimediale Lerneinheiten dar, die sich dem Nutzer auf interaktive Weise medial präsentieren. *Lernobjekte* beschränken sich auf eine begrenzte Anzahl von zu realisierenden Lernzielen und bieten Nutzern die Möglichkeit, den Gegenstandsbereich explorativ zu erkunden.
Die Integration von *Lernobjekten* in das *Informatik Lernlabor* geschieht durch Kombination von Phasen des Präsenzlernens und Phasen des computerbasierten E-Learning. Diese sich abwechselnden Lernformen werden auch als *Blendet Learning* bezeichnet.
Lernobjekte sollen im *Informatik Lernlabor* möglichst interaktiv und für kollaboratives Arbeiten zugänglich angeboten werden. Diese Aufgabe kann beispielsweise von Lernplattformen und spezieller Groupware mit spezifischen E-Learning Funktionen übernommen werden. In diesem Zusammenhang können den Lernenden Videosequenzen von realen Arbeitsabläufen in Informatiksystemen offeriert werden. Zusätzlich sind Lehrende in der

Lage, Interviews mit Nutzern und Auftraggebern, Entwicklungsgespräche über Entwurfsentscheidungen und Animationen zu Arbeitsabläufen und Informationsflüssen (webbasiert) zur Verfügung zu stellen.
Das interaktive Medium Lernsoftware ermöglicht somit vielschichtige Sichten auf das Produkt Software und den SWE-Prozess. Diesbezüglich kann ein Anwendungs- und Realitätsbezug der Lernenden sichergestellt werden und eine Förderung von vernetztem Wissen seitens der Nutzer des *Informatik Lernlabors* stattfinden.
Die Arbeitsformen des *Informatik Lernlabors* lassen sich in zwei wesentliche Phasen unterteilen [Magenheim 2003a, S. 74]:
Dekonstruktionsphase

- *Exploration*
  Die Lernenden sollen hierbei mittels der oben angesprochenen interaktiven Medien den Gegenstandsbereich erkunden und folglich ein erhöhtes Verständnis des Informatiksystems erlangen. Die Durchführung der Explorationsphase kann sowohl arbeitsteilig als auch in Gruppenarbeit erfolgen.

- *Re-engineering*
  In dieser Phase werden die Lernenden angewiesen, das vorhandene Informatiksystem zu verändern und zusätzlich neue Module zu entwickeln und einzubetten. Diesbezüglich sollen Designentwürfe verglichen und hinsichtlich ihrer technischen Funktionalität bewertet werden. Zusätzlich können in dieser Dekonstruktionsphase auch mögliche soziale Folgen diskutiert und erörtert werden. Die Zielsetzung dieser Phase ist es, eine fundierte Wissensbasis seitens der Lernenden zu schaffen.

Konstruktionsphase

- *Transfer*
  In der Transferphase werden die Lernenden aufgefordert, einen komplexen Auftrag zur Entwicklung eines Informatiksystems arbeitsteilig zu realisieren. Erforderlich hierfür ist der Transfer des bei der Dekonstruktionsphase erworbenen Wissens auf die neue Anforderungssituation.

- *Softwareentwicklung*
  Dieser zeitlich umfassende Abschnitt der Softwarekonstruktion beinhaltet die für den Softwareentwurf üblichen Teilphasen, wie Anforderungsdefinition, Spezifikation, Entwurf und Implementierung. Die Softwareentwicklungsphasen können mit Hilfe handlungsorientierter Modellierungskonzepte wie CRC Karten oder dem *Object Game* gestaltet werden.

- *Evaluation*
  Diese abschließende Phase der Konstruktion beinhaltet eine Bewertung hinsichtlich der erreichten Lernziele seitens der Schüler. Weiterhin können die Lernenden die Qualität des erstellten Produkts anhand der zuvor formulierten Anforderungsdefinition einschätzen. Die Evaluation von erbrachten Eigenleistungen schafft somit einen qualitativen Fortschritt bezüglich der stattfindenden Lernprozesse seitens der Schüler.

Inhaltsmodule des Informatik Lernlabors

Das *Informatik Lernlabor* verfügt zurzeit über drei technisch ausgereifte Inhaltsmodule, die derzeit in Schul- und Hochschullehre ihren Einsatz finden. Das Modul Onlineredaktion und die von mir entwickelte Software zur visuellen Programmierung sollen ebenfalls im *Informatik Lernlabor* eingesetzt werden. Die bisherigen Inhaltsmodule strukturieren sich folgendermaßen:

- Das Modul Hochregallager hat die Steuerung von Transport- und Lagerungsprozessen in einem Hochregallager zu Gegenstand. [...]
- Im Modul Schulkiosk werden elementarste Konzepte eines Warenwirtschaftssystems thematisiert. Die zu dekonstruierende Software bezieht sich auf die Ein- und Verkaufsvorgänge eines Schulkiosks. [...]
- Das Computerspiel Ursuppe bildet den Ausgangspunkt für das Modul Computerspiel. Die zugehörige Software ist in der Lage, nach entsprechenden Benutzereingaben die Spielverwaltung zu übernehmen und Spielstände grafisch anzuzeigen. [...]
- Im Modul Onlineredaktion sollen vor allem webbasierte Transaktionen, die Gestaltung von interaktiven Webseiten und die Speicherung von deren Inhalten in einer Datenbank thematisiert werden.

6.2.3. Fallbeispiel *Kommissionierstation* in der Hochschullehre

Die Unterrichtsreihe zur Evaluation des Kompetenzmessinstruments für informatisches Modellieren basiert auf dem *LEGO Mindstorms* System. Dieses setzt sich aus unterschiedlichen LEGO-Komponenten zusammen. Zum Bau eines *LEGO Mindstorms* Systems nutzt man vorwiegend die handelsüblichen LEGO-Steine sowie Bauteile aus dem Bereich LEGO-Technik. Darüber hinaus kommen auch Motoren, Sensoren und Steuereinheiten (*NXTs*) zum Einsatz. Diese Steuereinheiten lassen sich beliebig konfigurieren und

programmieren. Dementsprechend hat man die Möglichkeit, Motoren und andere Peripheriegeräte beliebig auszulesen und anzusteuern. Die *NXT-Bausteine* unterstützen zusätzlich die Kommunikation untereinander via Bluetooth. Somit lassen sich unterschiedliche *NXT-Systeme* zu einem komplexen System kombinieren und unterschiedliche reale Prozesse (z.B.: Hochregallager, Kommissionierstationen, etc.) simulieren.
Wie oben beschrieben besteht im Rahmen der Vorlesung *Informatik Lernlabor* die Zielsetzung, die theoretischen Ansätze der systemorientierten Didaktik der Informatik nach Magenheim unter wissenschaftlicher Begleitung praktisch zu erproben. Innerhalb der Vorlesung *ILL '08* bestand die Aufgabe ein Fallbeispiel *Kommissionierstation* zu implementieren. Hierbei wurde den Studierenden die folgende Zielsetzung mitgeteilt:

Zielsetzung

Die Aufgabe besteht darin, mit Hilfe des *LEGO Mindstorms* Systems eine *Kommissionierstation* zu konstruieren. Dies umfasst die technische und software-technische Konstruktion, Verarbeitung und Abarbeitung von Aufträgen. Bei der Realisierung wurde großen Wert darauf gelegt, dass eine realitätsnahe Abbildung einer *Kommissionierstation* entsteht. Hierfür wurden Videoaufzeichnungen und bereits bestehende Systeme betrachtet und als Vorlage verwendet.

Anforderungen

Eine *Kommissionierungsstraße* besteht aus mehreren *Kommissionierungsstationen*, die durch ein Palettenband bedient werden. Auf einem Palettenband sollen sich verschiedene Paletten bewegen, die Aufträge entgegennehmen. Umgesetzt werden soll jedoch nur eine *Kommissionierungsstation*. Im Folgenden seien die Anforderungen für die Bestandteile der *Kommissionierungsstraße* aufgeführt:

- *Palettenband*
  Das Palettenband soll Paletten zu einer *Kommissionierungsstation* transportieren können.
  - *Palette*
    Eine Palette soll einen Code, der aus verschiedenfarbigen LEGO-Steinen besteht, enthalten. Eine Palette beinhaltet einen Auftrag.
- *Kommissionierungsstation*
  Eine *Kommissionierungsstation* ist Bestandteil einer *Kommissionierungsstraße*. Diese soll ein Förderband sowie drei *Kommisionierungstürme* enthalten.

  - *Förderband*
    Eine *Kommissionierungsstation* soll ein Förderband enthalten, welches die zu verwaltenden Steine aus dem *Kommissionierungstürmen*, zu der Palette befördert.
  - *Kommissionierungsturm*
    Ein *Kommissionierungsturm* ist Bestandteil der *Kommissionierungsstation*. Dieser enthält die zu verwaltenden LEGO-Steine einer bestimmten Kategorie. Eine *Kommissionierungsstation* soll drei *Kommissionierungstürme* enthalten, welche jeweils grüne, rote und graue Steine lagern.
  - *Farbsensor*
    Mit Hilfe des Farbsensors soll ein Farbcode, der sich auf der Palette befindet, ausgelesen werden.

- *Auftrag*
  Ein Auftrag soll die jeweilige Anzahl von grünen, roten und grauen LEGO-Steinen beinhalten.

Dekonstruktionsphase

- *Exploration*
  Anhand von Videomaterialien und eines bestehenden *LEGO Mindstorms Modells* waren die Studierenden angehalten, den Gegenstandsbereich des Hochregallagers zu erkunden und dessen Funktionsweise zu verstehen.

- *Re-engineering*
  Innerhalb der Re-Engineering-Phase erhielten die Studierenden den Auftrag ein weiteres Hochregal mit entsprechendem Lift zur Ein- und Auslagerung von Paletten in das Hochregallager zu integrieren. Hierzu waren sie aufgefordert, sich in den bestehenden Quellcode des Systems einzuarbeiten und sowohl die technische als auch die software-technische Umsetzung vorzunehmen. Hierdurch sollte eine fundierte Wissensbasis seitens der Studierenden aufgebaut werden.

Konstruktionsphase

- *Transfer*
  In der Transferphase werden die Lernenden aufgefordert, einen komplexen Auftrag zur Entwicklung eines Informatiksystems arbeitsteilig zu realisieren. Erforderlich hierfür ist der Transfer des bei der Dekonstruktionsphase erworbenen Wissens auf die neue Anforderungssituation.

- *Softwareentwicklung*
  Die Zielsetzung der Phase bestand in der Konstruktion einer *Kommissionierstation*. Innerhalb dieser umfangreichen Softwareentwicklungsphase wurden alle für den Softwareentwurf üblichen Teilphasen durchlaufen. Dementsprechend haben die Studierenden – wie oben beschrieben – die Anforderungen an die *Kommissionierstation* aufgenommen, die Analyse- und Designphasen durchlaufen und deren Implementierung vorgenommen. Während jener Softwareentwicklungsphasen kamen handlungsorientierte Modellierungskonzepte wie CRC Karten, Klassendiagramme oder das *Object Game* zum Einsatz.

- *Evaluation*
  Die abschließende Evaluationsphase beinhaltet eine Bewertung in Bezug auf erreichte Lernziele der Studierenden. Weiterhin ist hierbei die Qualität des entwickelten Produkts anhand der zuvor formulierten Anforderungen bewertet worden.

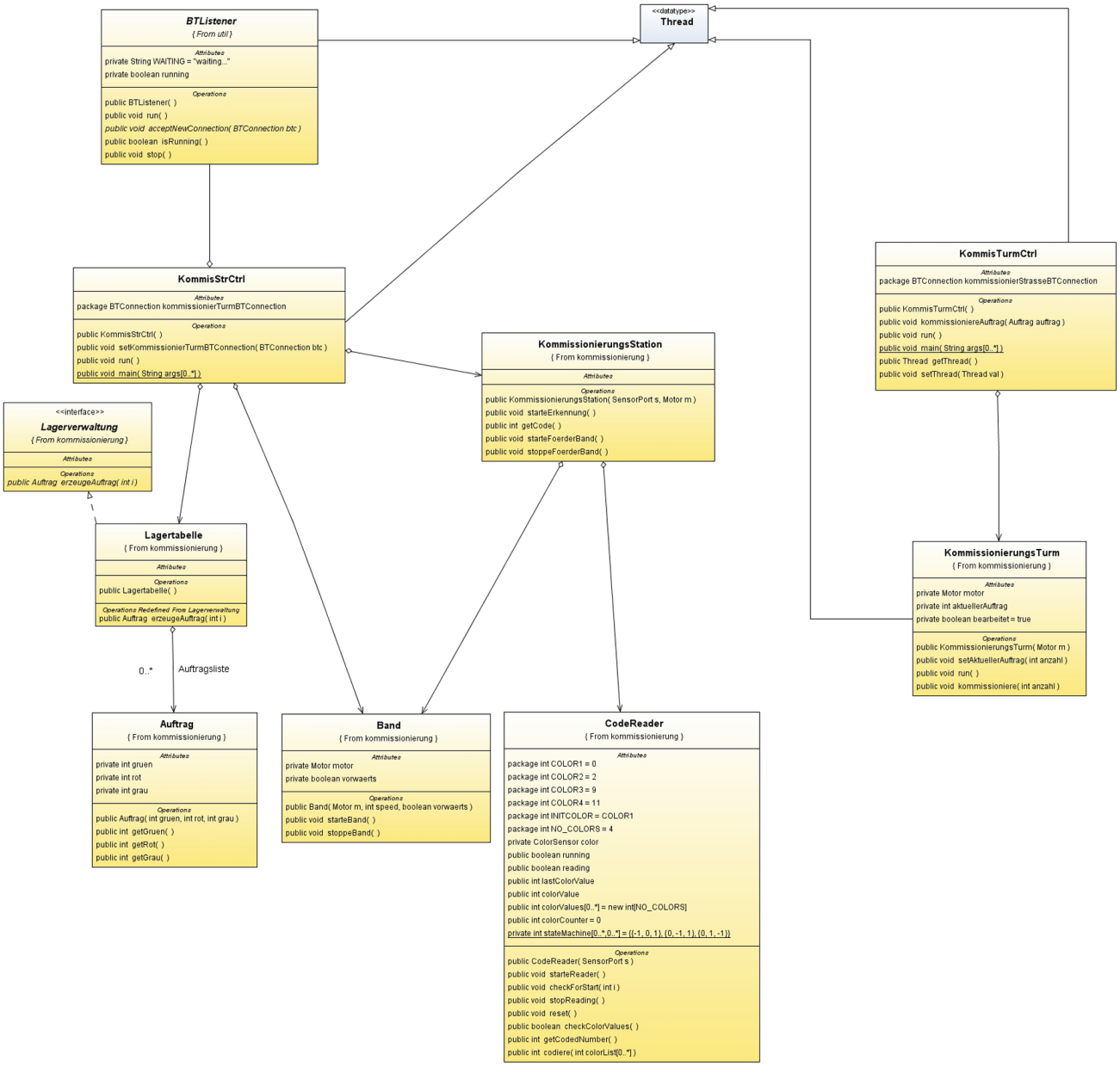


Abbildung 6.12.: Klassendiagramm der ILL-Kommissionierstation

Das neue ILL-Inhaltsmodul Kommissionierstation

Die *Kommissionierstation* besteht aus vier Hauptbestandteilen:

1. Kontrolleinheiten (bestehend aus zwei *NXTs*)
2. Transportband der einzelnen Warenstücke
3. Kommissioniertürme
4. Transportband für die zu befüllenden Paletten

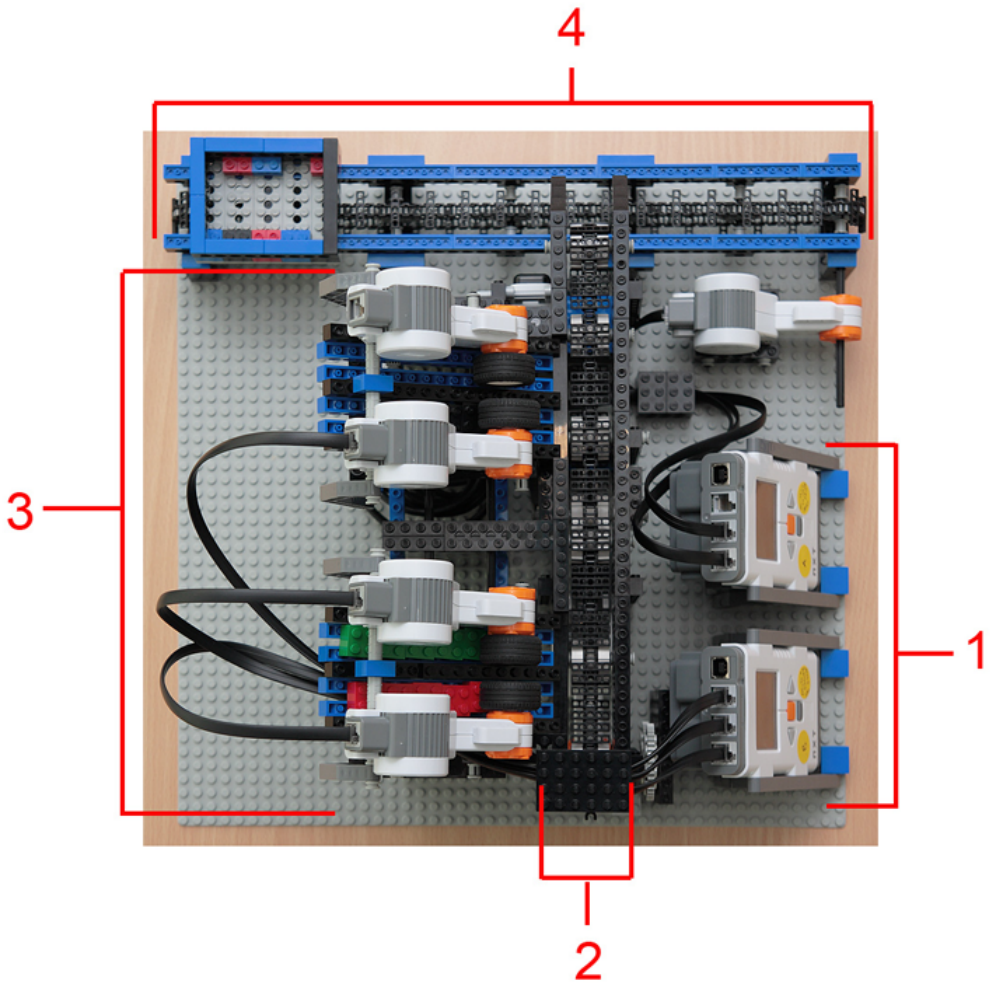


Abbildung 6.13.: Technische Bestandteile der Kommissionierstation

Folgende Schritte sind erforderlich, um die *Kommissionierstation* in Betrieb zu nehmen:

Zunächst muss die *Kommissionierstation* auf einen ebenen Untergrund gestellt werden. Nun gilt es, die *Kommissioniertürme* mit den Waren (hier 2x2 breite LEGO-Steine) zu bestücken.
Zum Einschalten der Kontrolleinheiten (*NXTs*) muss die orangefarbene Taste gedrückt werden. Nach Betätigung der Taste erscheint ein Startbildschirm und man gelangt in das Hauptmenü. Im nächsten Schritt muss eine Palette auf das Palettentransportband gelegt werden. Wenn man sich nun wieder der Kontrolleinheit zuwendet, sieht man den folgenden Startbildschirm.

Abbildung 6.14.: Startbildschirm eines *NXT-Bausteins*

Jetzt muss der Reiter im Menü auf *Run Default* eingestellt werden und die orangenen Tasten der *NXTs A* und *B* betätigt werden (die einzelnen Kontrolleinheiten sind entsprechend beschriftet). Nun verbinden sich die beiden *NXTs* über die *Bluetooth-Schnittstelle* miteinander und das Transportband für die Paletten beginnt zu laufen. Das Band sorgt für den Transport der Palette und stoppt erst nach dem die Palette den Farbsensor passiert hat und die korrekte Kodierung gelesen hat. Durch das Auslesen der Palettenkodierung wird die Bestückung der Palette bestimmt und ein Signal an den jeweiligen

Kommissionierturm weitergegeben. Zeitgleich beginnt sich das Transportband für die Warenstücke in Bewegung zu setzen. Die *Kommissioniertürme* lassen die passende Anzahl an Warenstücken auf dieses Band fallen, welches die Warenstücke dann zu der Palette befördern. Anschließend fährt das Palettenband weiter und die nächste Palette kann bearbeitet werden.
Für die fehlerfreie Abarbeitung einer Palettenreihe, sollte zwischen den Paletten ein Mindestabstand von 2-3 cm eingehalten werden. Um die *NXTs* abzuschalten muss zweimal die Taste, die sich unter der orangefarbenen Taste befindet, betätigt werden.

Farbcodierung

Die Farben schwarz, rot und blau wurden aus dem Grunde gewählt, da diese die geringste Fehleranfälligkeit hatten. In verschiedenen Tests wurden Kombinationen von Farben gewählt und auf deren Tauglichkeit überprüft. So stellte sich heraus, dass besonders helle Farben neben dunklen Farben, und umgekehrt, nicht korrekt erkannt wurden. Als Beispiel wurde weiß-schwarz-weiß als weiß-blau-weiß erkannt oder schwarz-gelb-schwarz als schwarz-grün-schwarz. Somit schloss sich der Gebrauch dieser Farben als Kombinationen für die Farbcodierung aus. Die Fehleranfälligkeit des Farbsensors lässt sich daraus ersehen, dass der Sensor immer einen Farbbereich erkennt. Besonders an Übergängen von einer Farbe zu einer Anderen traten Farbfehler auf, die herausgefiltert werden mussten. Für diese Implementierung gelten folgende Farbwerte:

- COLOR1 = 0 // schwarz
- COLOR2 = 3 // blau
- COLOR3 = 5 // rot
- INITCOLOR = COLOR1 = 0 // schwarz

Die Farberkennung liest einen Farbwert ein und vergleicht diesen mit dem zuletzt gelesenen Wert. Unterscheiden sich diese Werte, muss ein Farbwechsel stattgefunden haben. Die Folge von aufeinanderfolgen Farben wird als Binärstring codiert, sodass eine eindeutige Zuordnung von Farbfolge und Aufträgen erfolgen kann.

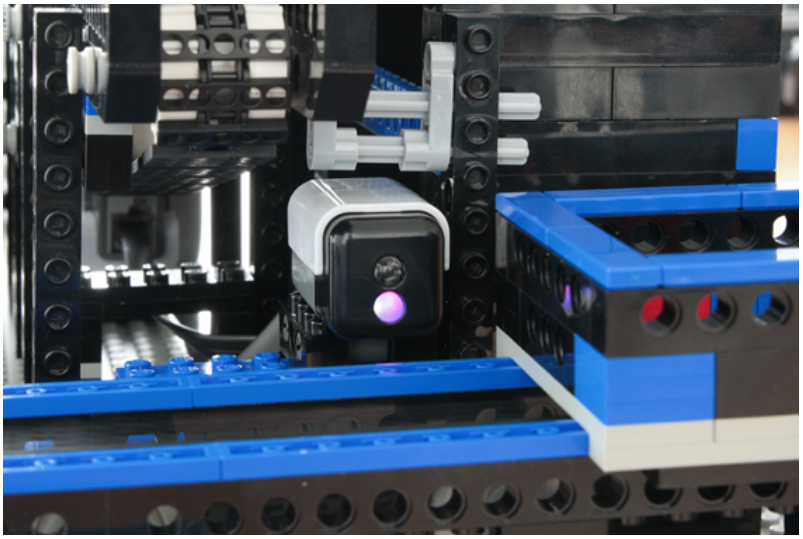


Abbildung 6.15.: Farbsensor

6.2.4. Fallbeispiel *Kommissionierstation* als Unterrichtsreihe zur Kompetenzmessung

Reduktion unter didaktischen Gesichtspunkten

Das im Rahmen des Seminars aufgenommene Feedback der Lehramts-Studierenden war überaus positiv. Der Einsatz der *LEGO Mindstorms* hat sich als motivierend herausgestellt. Darüber hinaus stand nach Abschluss des Seminars *Informatik Lernlabor SoSe 2008* ein funktionstüchtiger Prototyp der *Kommissionierstation* zur Verfügung.
In diesem Zusammenhang wurde auch die Tauglichkeit der Inhaltseinheit *Kommissionierstation* für den Einsatz im Informatikunterricht in der Sekundarstufe II thematisiert und bewertet. Hier verfestigte sich nach zahlreichen Diskussionen die Annahme, dass die Thematik grundsätzlich geeignet sei und in der Erfahrungswelt der Schüler verankert ist. Es gilt jedoch eine didaktische Reduktion der Einheit vorzunehmen um technische Hürden auszuschließen und das Re-Engineering sowie die Phase der Softwareentwicklung handhabbarer und für Schüler leistbar zu machen.

Technische Vereinfachungen

Aufgrund der Erfahrungen an der Hochschule und den praktischen Erfahrungen zur didaktischen Tauglichkeit des *ILL-Moduls Kommissionierstation* für den schulischen Einsatz der Inhaltseinheit, wurden technische Vereinfachungen und Vereinfachungen im Quellcode vorgesehen. Zunächst ist die Identifizierung der einzelnen Paletten mittels

eines RFID-Sensors realisiert worden. Hierzu wurde an jeder Palette ein eindeutiger RFID-Transponder angebracht. Hiermit wurde die störanfällige Farberkennung ersetzt und der Programmcode deutlich vereinfacht. Im Gegensatz zur Codierung der einzelnen Paletten durch farbige Bausteine erfolgt ein simples Auslesen des jeweiligen Byte-Werts, der die eindeutige Kennung der jeweiligen Palette (durch den eindeutigen Transponder) darstellt.

Reduzierte Ausbaustufen und Meilensteine

Es ergeben sich die folgenden Ausbaustufen bei der Verwendung der technisch reduzierten *Kommissionierstation*:

1. Ausbaustufe I *Ein Kommissionierturm*
   In der Ausbaustufe I ist die rudimentäre Funktionalität der *Kommissionierstation* umgesetzt. Mit einem *Kommissionierturm* kann eine fest im Quellcode hinterlegte Anzahl von Steinen in einer Farbe auf das Transportband geschoben und in die jeweilige Palette befördert werden.

2. Ausbaustufe II *Drei Kommissioniertürme*
   In der Ausbaustufe II werden drei *Kommissioniertürme* verwendet. Diese müssen entsprechend in das technische LEGO-Modell integriert werden. Auf Modellierungsebene gilt es, die *Kommissioniertürme* adäquat in das entsprechende Entwurfsmodell (z.B. Klassendiagramm) mit aufzunehmen. Letztendlich müssen die zwei zusätzlichen Türme auch im Quellcode mit angepasst werden. Sofern die jeweilige Iteration der Modellierungsphase abgeschlossen ist, lässt sich das Modell eindeutig in objektorientierten Quellcode übersetzten.

3. Ausbaustufe III *RFID Steuerung*
   In der Ausbaustufe III müssen weitreichende Änderungen an der *Kommissionierstation* vorgenommen werden. Es gilt aus technischer Sicht einen RFID-Sensor in die Station zu integrieren und die einzelnen Paletten mit RFID-Transpondern auszustatten. Auf Modellierungsebene müssen die entsprechenden Komponenten (z.B. der RFID-Sensor) mit aufgenommen werden. Auf Quellcodeebene gilt es neben der Übernahme der Änderungen aus dem Entwurfsmodell ebenso die feste Hinterlegung der *Kommissionieraufträge* im Quellcode aufzulösen. Hierzu musste eine Datenstruktur hinterlegt werden, die eine Zuordnung von RFID-IDs zu den entsprechenden *Kommissionieraufträgen* ermöglicht.

Neben der technischen Realisierung und Erprobung der drei oben genannten Ausbaustufen der *Kommissionierstation* wurden die Änderungen im Modell sowie die Auswirkungen im Quellcode aufbereitet. In diesem Zusammenhang finden sich im Anhang dieser Arbeit die einzelnen Schritte zur Erweiterung des Quellcodes zwischen den einzelnen Ausbaustufen.

Grobplanung einer Unterrichtsreihe

Auf Grundlage der technisch und didaktisch reduzierten Inhaltseinheit *Kommissionierstation* des *ILL* wurde die Grobplanung der folgenden Unterrichtsreihe zur Evaluation des entwickelten Kompetenzmessinstruments für informatisches Modellieren vorgenommen. Auf theoretischer Grundlage der Phasen des *ILL* Magenheim [2003a] und der oben genannten Ausbaustufen wurde die folgende Grobplanung vorgeschlagen:

Dekonstruktion	
Exploration (1./2.Woche)	Die Lernenden erkunden den Gegenstandsbereich anhand von drei medialen Repräsentationsformen des Gegenstandsbereichs: 1. Die Lernenden erkunden die reale *Kommissionierstation* anhand von Videomaterial eines realen Systems. 2. Die Lernenden erkunden das bestehende *LEGO Mindstorms* Modell der *Kommissionierstation* in der Ausbaustufe I. Wie oben beschrieben handelt es sich hierbei um eine Minimalversion mit nur einem *Kommissionierturm* und einem im Quellcode hinterlegten *Kommissionierauftrag*. 3. Die Lernenden erkunden das Software-Modell anhand bereits modellierter Entwurfs-Klassendiagramme und anhand des bestehenden Quellcodes mit Hilfe von IDEs.
Re-Engineering (3.-4. Woche)	Die Lernenden sind angehalten die Integration eines weiteren *Kommissionierturms* am *LEGO Mindstorms* Modell (Ausbaustufe II) vorzunehmen. Darüber hinaus gilt es, die Erweiterung des Systems in den entsprechenden Entwurfsmodellen abzubilden und diese anschließend in den Quellcode zu übernehmen.

Konstruktion	
Transfer	Im Übergang zwischen Dekonstruktions- und Konstruktionsphase erfolgt der Lerntransfer der bei der Dekonstruktion erworbenen Fähigkeiten auf die neue Anforderungssituation
Softwareentwicklung (5.-8. Woche)	Die Lernenden sollen einen komplexen Auftrag zur Erweiterung der *Kommissionierstation* ausführen. Hierbei gilt es, die Ausbaustufe III zu realisieren. Dementsprechend muss ein RFID integriert werden, um eine automatische *Kommissionierung* von Waren zu ermöglichen. Hierzu muss die jeweilige Palette vom System erkannt und der jeweilige zugehörige *Kommissionierauftrag* ausgeführt werden.
Evaluation (8. Woche)	Die Lernenden bewerten die erreichten Lernziele und die Qualität des Produkts

Die Planung der Unterrichtsreihe zur ersten Evaluation des Kompetenzmessinstruments für informatisches Modellieren wurde bewusst als Grobplanung ausgelegt. Hier sollte die jeweilige Lehrperson möglichst viel Freiraum zur Gestaltung der Unterrichtsreihe und zur Erreichung der einzelnen vordefinierten Meilensteine (Ausbaustufen der *Kommissionierstation*) haben. Sowohl die Herangehensweise zur Vermittlung von Modellierungskompetenz als auch die Auswahl geeigneter Modellierungstechniken (z.B. *CRC-Karten*, Klassendiagramme, Objektdiagramme, etc.) liegt in der Kompetenz und Zuständigkeit der Lehrperson, die die Unterrichtsreihe begleitet.

6.3. Hypothesen für die Erprobung des Messinstruments

Unter der Annahme, dass das entwickelte Lehr-/Lernarrangement Kompetenzen in der informatischen Modellierung fördert, sollen im Folgenden Hypothesen zur Tauglichkeit des Instruments, Kompetenzzuwächse zu messen, angeführt werden.

Hypothese H1:

Das Messinstrument zeigt einen systematischen Kompetenzzuwachs der Lernenden beim Nachtest im Vergleich zum Vortest auf.

Zunächst soll dementsprechend geprüft werden, ob sich im Vergleich von Vor- und Nachtest ein Kompetenzzuwachs messen lässt. Hierbei soll das Gesamtergebnis der einzelnen Probanden beim Nachtest im Vergleich zum Vortest ermittelt und verglichen werden.

In Anlehnung an die Phasen des *ILL* macht es Sinn eine phasenabhängige Bündelung der Aufgaben in sog. Aufgabencluster vorzunehmen. Anhand der Inhalte und der zu fördernden Kompetenzen der Unterrichtsreihe werden drei Aufgabencluster gebildet:

1. *Cluster 1: Allgemeine Kompetenzen zu Vorgehensmodellen in der Softwaretechnik (Bündelung der Aufgaben 1,2,3)*
   Dieser Aufgabenbereich umfasst Kompetenzen zu den unterschiedlichen Vorgehensmodellen der Softwaretechnik. Hierbei können die Lernenden verschiedene lineare und iterative Vorgehensmodelle des Software-Engineerings (linear: z.B. vereinfachtes Wasserfallmodell, ...; iteratives Vorgehen: z.B. *RUP*) und deren Phasen benennen, für die Lösung eines softwaretechnischen Problems verwenden und beurteilen welche Phasen es zur Lösung des Problems (erneut) zu durchlaufen gilt.
   - **K1.3.6 Iteratives Vorgehen**
   - **K1.3.6.1** Die Lernenden sind in der Lage, abhängig von der jeweiligen Iteration des SWE-Prozesses, sinnvolle Modellierungstechniken auszuwählen, anzuwenden und zu beurteilen.
   - **K1.3.6.2** Die Lernenden sind in der Lage zu beurteilen, ob ein erneutes Durchlaufen einer bereits absolvierten Phase des SWE-Prozesses erforderlich ist; sie können abhängig von den auftretenden Problemen in der aktuellen Phase eine sinnvolle vorherige Phase auswählen, die es erneut zu durchlaufen gilt.

2. *Cluster 2: Kompetenzen für die Dekonstruktion und Analyse von Informatiksystemen (Bündelung der Aufgaben 4,5,6)*
   Dieser Aufgabenbereich fokussiert Kompetenzen zur Analyse eines (bestehenden) Informatiksystems. Dies umfasst die Ableitung von funktionalen Anforderungen bis hin zur Feinanalyse des Systems mit Hilfe der entsprechenden *UML-Notation*. Hierbei spielen implementierungsspezifische Details bei der Modellierung eine untergeordnete Rolle und sind lediglich für das Re-Engineering von Quellcode eines bestehenden Informatiksystems relevant.
   - **K1.3.1.1** Die Lernenden können eine geeignete (Software) Plattform/Basistechnologie auswählen, um das zu erstellende SW-Projekt zu entwickeln.
   - **K1.3.1.2** Die Lernenden sind in der Lage, Anwendungsfälle (Use Cases) zu ermitteln und anzugeben (benennen), diese zu analysieren (durchzuspielen); sie sind diesbezüglich auch in der Lage, Use Case Diagramme zu entwickeln. Hierbei können sich die Lernenden einen Eindruck verschaffen, was die zu entwickelnde Software zu leisten hat.
   - **K1.3.1.3** Die Lernenden sind in der Lage, funktionale Anforderungen an die zu entwickelnde Software zu ermitteln; dabei sind sie befähigt, die Ziele (z.B. funktionale Anforderungen), Grenzen (z.B. Abgrenzung zu bestehenden Softwaresystemen) und

Stakeholder innerhalb der Problemdomäne zu ermitteln. Hierbei besteht wiederum die Zielsetzung herauszufinden, was das zu entwickelnde System leisten soll.

- **K1.3.1.4** Die Lernenden sind in der Lage, eine tabellarische Use Case Beschreibung in ein Aktivitätendiagramm zu überführen. Hierdurch können mögliche Anwendungsszenarien genauer analysiert werden.
- **K1.3.1.5** Die Lernenden sind in der Lage, die zuvor ermittelten funktionalen Anforderungen für andere verständlich und nachvollziehbar darzustellen (dokumentieren). Hierbei besteht die Zielsetzung, ein gemeinsames Dokument (im Sinne eines Pflichtenhefts) für die SWE-Teams im Hinblick auf die weiteren Phasen des SWE-Prozess zu entwickeln.
- **K1.3.2 Analyse**
- **K1.3.2.1** Die Lernenden können objektorientierte Begrifflichkeiten angeben und erläutern. Dies ist Grundvoraussetzung, um eine objektorientierte Dekomposition durchführen zu können.
- **K1.3.2.2** Die Lernenden sind in der Lage, eine objektorientierte Dekomposition durchzuführen; d.h., sie können anhand einer textuellen Beschreibung des Problembereichs mögliche Klassenkandidaten, Attribute und Methoden ermitteln (auffinden) und diese in eine formale Darstellungsform überführen. Hierbei besteht die Zielsetzung ein Modell des Problembereichs zu erstellen.
- **K1.3.2.3** Die Lernenden sind in der Lage, relevante statische und dynamische *UML-Diagramme* ohne implementierungsspezifische Details zu entwickeln (z.B. *CRC-Karten*). Durch diese formale konzeptionelle Modellierung erhalten die Lernenden einen vertieften Einblick in die Problemdomäne.

3. *Cluster 3: Kompetenzen für die Konstruktion von Informatiksystemen (Bündelung der Aufgaben 7,8,9,10)*
   Dieser Aufgabenbereich fokussiert Kompetenzen zum Design und zur Gestaltung eines neuen Informatiksystems. Dies umfasst im Gegensatz zu den Aufgaben in *Cluster 2* vorrangig das konkrete Lösungsdesign eines Informatiksystems bzw. dessen Bestandteilen und die konkrete Implementierung auf Grundlage von Design-Modellen in *UML-Notation*. Hierbei liegt der Fokus bei der Modellierung auf Modellen mit implementierungsspezifischen Details. Neben der Konstruktion von Informatiksystemen adressieren die Aufgaben aus *Cluster 3* ebenso den Test eines Informatiksystems.
   - **K1.3.3 Design**
   - **K1.3.3.1** Die Lernenden sind in der Lage, die Architektur der zu entwickelnden Software zu bestimmen; dabei wählen sie eine geeignete Programmiersprache aus, berücksichtigen Aspekte der Verteilung, Nebenläufigkeit/Parallelität und möglicher

Entwurfsmuster. Dies ist eine wichtige Voraussetzung für die Entwicklung von entwurfsspezifischen *UML-Diagrammarten*.

- **K1.3.3.2** Die Lernenden sind in der Lage, sinnvolle Schnittstellen zu bestimmen um eine spätere erfolgreiche Integration von Programmmodulen zu ermöglichen.
- **K1.3.3.3** Die Lernenden sind in der Lage, relevante **statische und dynamische** *UML-Diagramme* mit implementierungsspezifischen Details zu entwickeln. Hierdurch entsteht ein entwurfsspezifisches Modell, welches in Quellcode einer objektorientierten Hochsprache überführt werden kann.
- **K1.3.4 Implementierung**
- **K1.3.4.1** Die Lernenden sind in der Lage, die Architektur der zu entwickelnden Software zu bestimmen; dabei wählen sie eine geeignete Programmiersprache aus, berücksichtigen Aspekte der Verteilung, Nebenläufigkeit/Parallelität und möglicher Entwurfsmuster. Dies ist eine wichtige Voraussetzung für die Entwicklung von entwurfsspezifischen *UML-Diagrammarten*.
- **K1.3.4.1.1** Die Lernenden sind in der Lage, Programmierkonzepte, wie z.B. das Variablenkonzept und Kontrollstrukturen (Bedingte Anweisung, Schleifenkonstruktion) in der Programmiersprache zu implementieren.
- **K1.3.4.1.2** Die Lernenden sind in der Lage, ein Klassendiagramm in objektorientierten *Java-Code* zu überführen; Sie können Klassen, Attribute und Methoden sowie Assoziationen und Vererbungsstrukturen in *Java-Code* implementieren.
- **K1.3.4.1.3** Die Lernenden sind in der Lage, Programmbibliotheken (z.B. Java-Swing) erfolgreich in eigene Programmmodule einzubinden.
- **K1.3.4.2** Die Lernenden sind in der Lage, mit Hilfe von integrierten Entwicklungsumgebungen (IDEs) Programmmodule zu implementieren und zu integrieren.
- **K1.3.4.3** Die Lernenden sind in der Lage, mit Hilfe einer Versionsverwaltungssoftware (z.B. Subversion) Programmmodule und deren Versionierung zu verwalten.
- **K1.3.4.4** Die Lernenden sind in der Lage, die selbst implementierten Programmmodule nachvollziehbar (im Hinblick auf gute Wartbarkeit) zu dokumentieren (z.B. mit Java-Doc).
- **K1.3.4.5** Die Lernenden sind in der Lage, Programmmodule sinnvoll in ein bestehendes Softwaresystem zu integrieren. Somit können Teile der zu entwickelnden Software zu einem lauffähigen System aggregiert werden.
- **K1.3.5 Test**
- **K1.3.5.1** Die Lernenden sind in der Lage, ein bestehendes Softwaresystem systematisch zu testen. Hierbei besteht die Zielsetzung unter anderem darin, zu überprüfen, ob die zuvor spezifizierten funktionalen Anforderungen erfolgreich umgesetzt wurden.

- **K1.3.5.1.1** Die Lernenden sind in der Lage, zu Beginn der Testphase einen geeigneten Testplan zu entwickeln.
- **K1.3.5.1.3** Die Lernenden sind in der Lage, Testfälle zu ermitteln (Extremfälle und unerwartete Eingabedaten erzeugen) oder zu entwickeln; sie können diese zum Test verwenden und die daraus resultierenden Ausgaben protokollieren.

Aufgrund der besonders (zeit-)intensiven Behandlung der Konstruktionsphase ist ein Kompetenzzuwachs insbesondere bei *Cluster 3* im Vergleich zu *Cluster 1* und *Cluster 2* zu erwarten. Demzufolge sollte das Instrument insbesondere für diesen Kompetenzbereich einen deutlichen Kompetenzzuwachs messen können. Dieser sollte merklich über dem ermittelten Kompetenzzuwachs bei den *Aufgabenclustern 2* und *3* liegen. Die Hypothese *H2* fasst den Sachverhalt in eine Aussage, die es zu überprüfen gilt, zusammen:

Hypothese H2:

Das Messinstrument zeigt einen systematischen Kompetenzzuwachs bei dem* Aufgabencluster 3 *(Konstruktion von IS) auf. Dieser ist größer als der Kompetenzzuwachs bei den* Aufgabenclustern 1 *(allgemeine Aufgaben zu Vorgehensmodellen in der Softwaretechnik) und 2 (Dekonstruktion von IS).

Diese Hypothesen gilt es bei der Auswertung der Ergebnisse der Erprobung im folgenden Kapitel zu überprüfen.

6.4. Zusammenfassung

Dieses Kapitel hatte die Zielsetzung, den Prozess zur Entwicklung von Aufgaben und darin enthaltener Items auf Grundlage der im Kapitel 5 formulierten Kompetenzprofile (Kategoriendefinitionen) aufzuzeigen. In diesem Kontext wurden zunächst theoretische Grundlagen zur Testentwicklung und zur Fragebogenkonstruktion aufgezeigt und beispielhaft der Entwicklungsprozess einzelner repräsentativer Aufgaben des Messinstruments dargestellt. Hierbei wurden spezielle Items für Kompetenzkategorien entwickelt und zu Aufgaben kombiniert, die möglichst mehrere im Strukturmodell abgebildete Kompetenzbereiche abfragen.
Eine weitere Zielsetzung war es, die Konzeption eines Lehr-/Lernarrangements zur Förderung von Modellierungskompetenzen zu beschreiben. Dieses soll als thematische und unterrichtsmethodische Grundlage für die Erprobung des Messinstruments fungieren. Zur theoretischen Fundierung wurden verschiedene didaktische Ansätze zum Verständnis von

Informatiksystemen aufgeführt und deren Einfluss auf die Gestaltung der Unterrichtsreihe dargestellt. Ebenso wurde die Verknüpfung der einzelnen Ansätze mit dem Fokus auf der systemorientierten Didaktik der Informatik dargelegt. Neben der theoretischen Fundierung wurde die praktische Entwicklung der *LEGO Mindstorms* basierten Inhaltseinheit *Kommissionierstation* des *Informatik Lernlabors* dargestellt.
Abschließend wurden als Ergebnis dieses Kapitels im Hinblick auf die Lerneinheit zur Evaluation Hypothesen zur Wirksamkeit des Messinstruments formuliert, die es im folgenden Kapitel zu überprüfen gilt.
Das nächste Kapitel beschreibt die Erprobung des Kompetenzmessinstruments im Rahmen einer Unterrichtseinheit innerhalb der gymnasialen Oberstufe. Diese Erprobung wurde auf theoretischer und unterrichtspraktischer Grundlage des oben beschriebenen Lehr-/Lernarrangements geplant und im Frühjahr 2011 am Paderborner Pelizaeus Gymnasium durchgeführt.

7. Erprobung des Messinstruments für informatische Modellierungskompetenz

Dieses Kapitel beschreibt die Erprobung des Messinstruments für Modellierungskompetenz. Dementsprechend werden auf Grundlage des dafür konzipierten Lehr-/Lernarrangements das Untersuchungssetting und -design mit den einzelnen Messzeitpunkten dargestellt. Aufgrund vielfältiger positiver Erfahrungen zum Einsatz der Lerneinheit in der Hochschullehre gehen wir davon aus, dass diese die Entwicklung von Modellierungskompetenz fördert. Der Fokus der Einheit liegt insbesondere auf der Konstruktion von Informatiksystemen und den dafür erforderlichen Kompetenzen.
Die Ergebnisse der beiden Kompetenzmessungen sollen im Vergleich vorgestellt und differenziert interpretiert werden. Ziel soll in diesem Zusammenhang die Prüfung der zuvor aufgestellten Hypothesen *H1* und *H2* sein. Bei der Prüfung der Hypothese *H1* werden die Gesamtergebnisse der Erprobung betrachtet, wo hingegen bei *H2* die Ergebnisse unter Berücksichtigung der definierten *Aufgabencluster 1-3* analysiert werden.
Infolgedessen soll zunächst eine deskriptive statistische Analyse der Ergebnisse von Vor- und Nachtest und der Verteilung der Probanden im Hinblick auf die erreichte Punktzahl durchgeführt werden. In diesem Zusammenhang kommt in einem zweiten Schritt ein geeignetes induktives Verfahren zum Einsatz, um die statistische Signifikanz der Ergebnisse des Nachtests im Vergleich zum Vortest für *H1* und *H2* zu untersuchen.
Hierbei stellt sich die zentrale Frage, ob das Messinstrument in der Lage ist, messtechnisch zu differenzieren und Kompetenzzuwächse aufzuzeigen.
Wie im vorherigen Kapitel zur Planung der Unterrichtsreihe beschrieben, sieht das verwendete Untersuchungssetting zwei Messzeitpunkte vor. Einen zu Beginn der Unterrichtsreihe und einen zum Abschluss der Einheit. Im Folgenden wird nochmals der Verlauf der Unterrichtsreihe skizziert und die Messzeitpunkte aufgeführt. Ferner werden die Rahmenbedingungen für die Durchführung der Kompetenzmessung festgelegt.

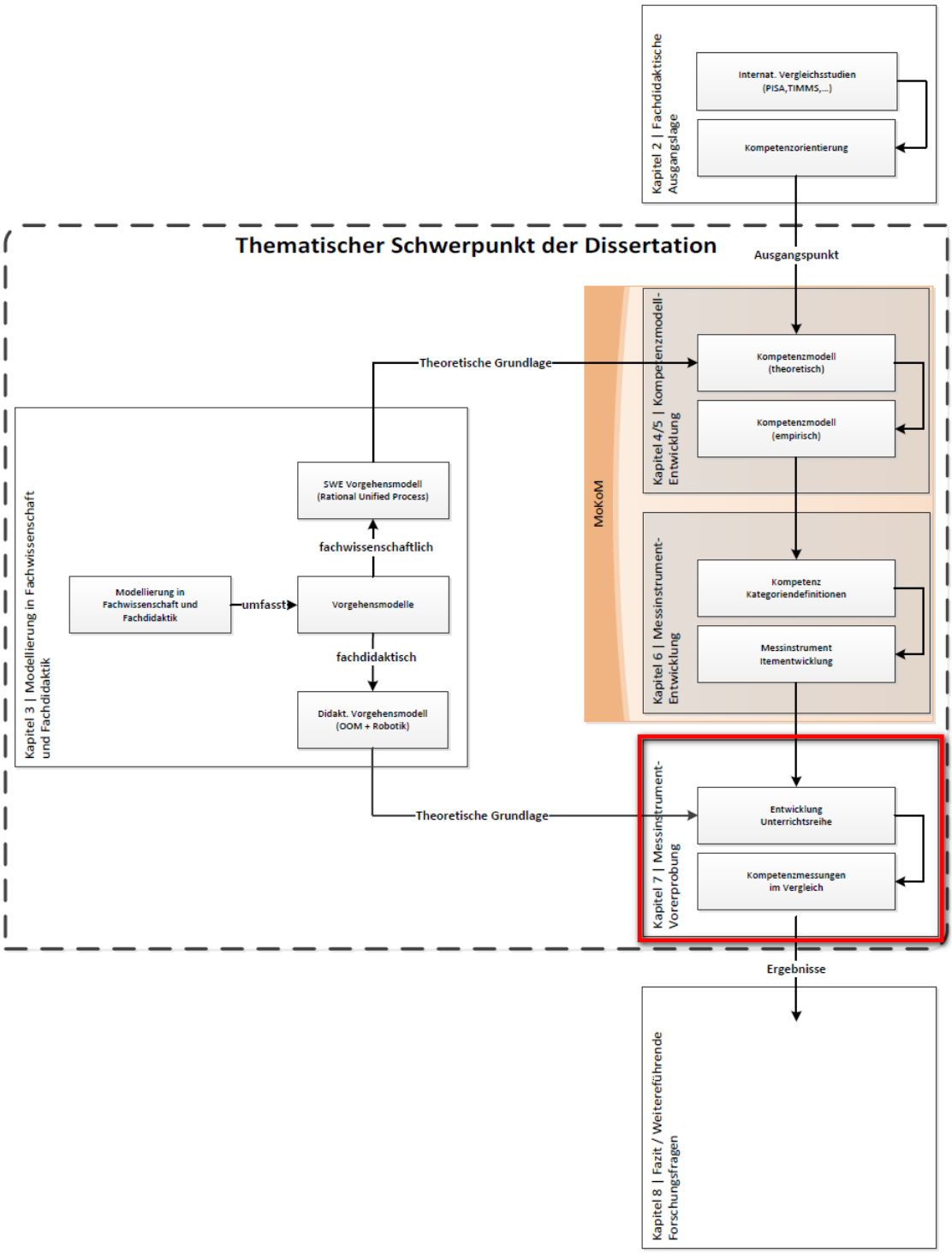


Abbildung 7.1.: Kapitel 7 im Gesamtkontext der Arbeit

7.1. Untersuchungssetting- und Design

7.1.1. Lerngruppe und zeitlicher Rahmen

Im Mai und Juni 2011 wurde die Erprobung des Kompetenzmessinstruments über acht Wochen am Paderborner Pelizaeus Gymnasium durchgeführt. Beim Kurs handelte es sich um einen Informatik Leistungskurs der Jahrgangsstufe 13. Der Kurs umfasste 28 Schülerinnen und Schüler. Es handelte sich hierbei um einen sog. Koorperations-Kurs der gemeinschaftlich von den Gymnasien im Paderborner Innenstadtbereich, nämlich Pelizaeus Gymnasium, Reismann Gymnasium und Theodorianum organisiert wurde. Durchschnittlich hatte der Kurs jeweils fünf Wochenstunden, wobei dieser jeweils abwechselnd in den geraden Kalenderwochen sechs Wochenstunden und in den ungeraden Kalenderwochen vier Wochenstunden umfasste.

7.1.2. Messzeitpunkte

Die Messungen hatten einen zeitlichen Umfang von jeweils 90 Minuten. Hierbei wurden die Probanden aufgefordert, alle zehn Aufgaben des Messinstruments zu bearbeiten. Im Verlauf der Unterrichtseinheit waren die folgenden Messzeitpunkte vorgesehen:

Dekonstruktion

1. Kompetenzmessung

Exploration
(1./2.Woche)

Re-Engineering
(3.-4. Woche)

Konstruktion

Transfer

Softwareentwicklung
(5.-10. Woche)

Evaluation
(10. Woche)

2. Kompetenzmessung

Zu den beiden Messzeitpunkten kam jeweils dasselbe Instrument (siehe Anhang) und das gleiche Bewertungsschema zum Einsatz. Um die Ergebnisse der Probanden bei Vor- und Nachtest vergleichen zu können und trotzdem die Anonymität der Schülerinnen und Schüler zu wahren, wurde ein eindeutiger Code zur Identifizierung vorgegeben.

```
1. Erster Buchstabe Vorname der Mutter
2. Zweiter Buchstabe Vorname des Vaters
3. Erste Ziffer des Geburtstages (inkl. 0)
4. Zweite Ziffer des Geburtstages

z.B. HI02 (Daten des Autors)
```

Bei den Kompetenzmessungen zu Beginn der Unterrichtseinheit (Vortest) und zu deren Abschluss (Nachtest) konnten die Ergebnisse von einer Grundgesamtheit von $N = 20$ Probanden ausgewertet werden. Die Abweichung der Grundgesamtheit zur Kursstärke hängt mit der Abwesenheit bestimmter Schüler bei Vor- und Nachtest zusammen.

7.2. H1 - Gesamtergebnis im Vergleich

Mit dem Ziel, zunächst einen Überblick über die Datenlage zu erlangen, werden in einem ersten Schritt ausschließlich deskriptive statistische Verfahren verwendet. Im weiteren Verlauf des Kapitels soll mit Hilfe eines induktiven Verfahrens überprüft werden, ob sich tendenziell abzeichnende Kompetenzzuwächse überzufällig (also statistisch signifikant) sind.

7.2.1. Deskriptive statistische Analyse

Dieses Teilkapitel hat die Zielsetzung, die erhobenen Daten im Hinblick auf die aufgestellte Hypothese *H1* zu überprüfen. Demzufolge wird zunächst das Gesamtergebnis von Vor- und Nachtest, d.h. die jeweilig vom Probanden erreichte Punktzahl, dargestellt und verglichen.

Zu H1 Gesamtergebnisse

Zur Prüfung der Hypothese *H1* sollen die Gesamtergebnisse zu beiden Messzeitpunkten verglichen werden. Dementsprechend umfasst die folgende Tabelle die erreichte Gesamtpunktzahl der einzelnen Probanden.

Gesamtergebnis
(max Punkte = 172)

Proband	**Ergebnis Vortest**		**Ergebnis Nachtest**		**Vergleich**
Kürzel	**nominal**	**prozentual**	**nominal**	**prozentual**	**Tendenz**
AN05	77	44,77%	96	55,81%	**+11,05%**
NI06	99,5	57,85%	105,5	61,34%	**+3,49%**
IR14	61,5	35,76%	90,5	52,62%	**+16,86%**
JJ13	47	27,33%	96	55,81%	**+28,49%**
OI04	84	48,84%	114	66,28%	**+17,44%**
AR16	55,5	32,27%	105,5	61,34%	**+29,07%**
OA27	76,5	44,48%	118	68,60%	**+24,13%**
LG18	71,5	41,57%	74	43,02%	**+1,45%**
AB26	74,5	43,31%	129,5	75,29%	**+31,98%**
EI06	74,5	43,31%	118,5	68,90%	**+25,58%**
EA26	114	66,28%	131,5	76,45%	**+10,17%**
AG15	117	68,02%	144	83,72%	**+15,70%**
ES29	95,5	55,52%	130	75,58%	**+20,06%**
OR01	76	44,19%	94,5	54,94%	**+10,76%**
EA05	65	37,79%	100	58,14%	**+20,35%**
IL08	52,5	30,52%	84	48,84%	**+18,31%**
AG12	126	73,26%	102	59,30%	**-13,95%**
AR01	87	50,58%	118	68,60%	**+18,02%**
UG23	93	54,07%	133,5	77,62%	**+23,55%**
LE06	114	66,28%	79,5	46,22%	**-20,06%**
arithm. Mittel	**83,08**	**48,30%**	**108,23**	**62,92%**	***+14,62%***

Tabelle 7.1.: Gesamtergebnisse der Erprobung

Die erste Spalte mit der Bezeichnung *Proband* beinhaltet die einzelnen eindeutigen Kennungen der Probanden. Innerhalb der Spalten *Vortest* werden die nominal erreichte Punktzahl und die prozentual erreichte Punktzahl der einzelnen Probanden im Vortest aufgeführt. Die Spalten *Nachtest* beinhalten dementsprechend die nominalen und anteiligen Punktzahlen der Probanden im Nachtest. Innerhalb der Spalte *Vergleich/Tendenz* wird die Differenz zwischen anteilig erreichter Punktzahl im Nachtest (Minuend) und anteilig erreichter Punktzahl im Vortest (Subtrahend) berechnet und aufgeführt.
Betrachtet man die Gesamtergebnisse der Kompetenztests haben die Probanden im Vortest durchschnittlich $48,30\%$ der möglichen Punkte erreicht. Im Nachtest erreichten sie $62,92\%$ der Punkte. Folglich haben die Probanden im Nachtest im Mittel ca. $14,62\%$ mehr Punkte erreicht als im Vortest.

Bei Betrachtung der Ergebnisse der Probanden im Vergleich zeigt sich, dass 18 von 20 Probanden im Nachtest besser abgeschnitten haben als im Vortest. Dies könnte neben dem deutlichen prozentualen Zuwachs als Indiz für einen möglichen Kompetenzzuwachs interpretiert werden.
Um die Verteilung der Probanden im Hinblick auf die jeweils erreichte Punktzahl im Kompetenztest zu illustrieren, werden im Folgenden zwei Histogramme dargestellt. Diese zeigen die Häufigkeitsverteilung und enthalten auf der Ordinate die jeweilige Anzahl der Probanden und auf der Abszisse die prozentual erreichte Punktzahl. Hierbei wird die anteilig erreichte Punktzahl in zehn Leistungsklassen von 0% bis 100% eingeteilt. Um darüber hinaus die Verteilung von Vor- und Nachtest zu vergleichen, werden die beiden Histogramme für Vortest (Histogramm mit blauen Balken) und Nachtest (Histogramm mit roten Balken) nebeneinander dargestellt[1].

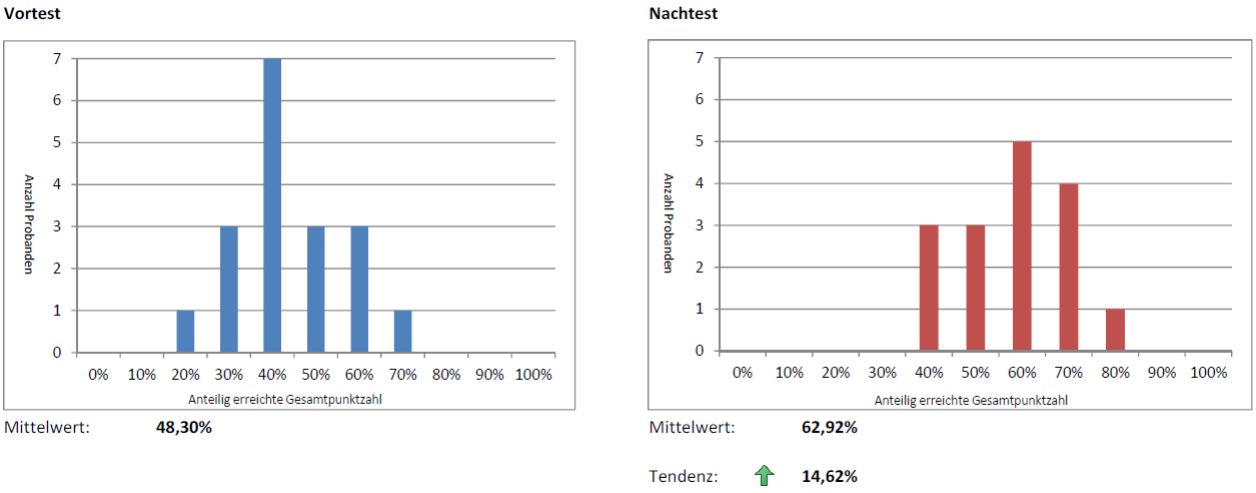


Abbildung 7.2.: Vor- und Nachtest im Vergleich

Bei Betrachtung der Histogramme wird deutlich, dass sich die Leistung der Probanden merklich verbessert hat. Im Vortest haben ca. ein Viertel der Probanden (vier Probanden) nicht die 40 %-Marke erreicht. Im Nachtest hingegen konnten alle 20 Probanden mehr als 40 % der Punkte erreichen. Bei der Verteilung der Probanden wird ein weiterer Unterschied deutlich: Im Vortest konnten die meisten (hier sieben Probanden) Probanden mehr als 50 % der Punkte erreichen. Im Nachtest hingegen haben fünf Probanden mehr als 60 % der Punkte erreicht und darüber hinaus vier Probanden sogar mehr als 70

[1] Die Vorkenntnisse der Probanden wurden bei dieser Auswertung nicht berücksichtigt. Dies sollte in einer breiten Erprobung des Messinstruments mit berücksichtigt werden.

% der Punkte erreichen können. Im Vergleich zum Vortest hat sich somit ein erheblicher Leistungszuwachs abgezeichnet.
Unter der Annahme, dass die durchgeführte Unterrichtsreihe Modellierungskompetenz fördert, könnten diese Ergebnisse einen Hinweis darauf geben, dass sich mit dem verwendeten Instrument ein Kompetenzzuwachs messen lässt.
Hierbei ist anzumerken, dass im Rahmen der Prüfung der Hypothese *H1* noch keine differenzierte Betrachtung der abgeprüften Kompetenzbereiche stattgefunden hat. Bevor diese in *H2* definierte Frage genauer betrachtet wird, sollen zunächst die Gesamtergebnisse von Vor- und Nachtest (*H1*) mit einer induktiven statistischen Methode zum Mittelwertvergleich untersucht werden. In diesem Zusammenhang soll festgestellt werden, ob sich die Ergebnisse der Probanden zufällig oder systematisch verbessert haben und ob das Kompetenzmessinstrument dazu in der Lage ist, diesen Zuwachs messtechnisch aufzuzeigen.

7.2.2. Induktive statistische Analyse

Wie einleitend beschrieben, soll neben der deskriptiven statistischen Analyse in einem weiteren Schritt mit Hilfe eines induktiven Verfahrens überprüft werden, ob sich tendenziell abzeichnende Kompetenzzuwächse statistisch signifikant sind.

7.2.3. Auswahl eines geeigneten Testverfahrens

Um zu überprüfen, ob die empirische Mittelwertdifferenz signifikant oder zufällig ist, kommt der t-Test als induktives statistisches Verfahren zum Mittelwertvergleich zum Einsatz. Die Voraussetzungen für den t-Test sind, dass die unabhängige Variable (hier das Treatment durch die Unterrichtseinheit) dichotom, also in genau zwei Ausprägungen vorliegt (vor Treatment durch die Unterrichtsreihe; nach Treatment durch die Unterrichtsreihe) und die abhängige Variable (hier die erreichte Gesamtpunktzahl oder die erreichte Punktzahl je Aufgabencluster metrisch skaliert ist. Des Weiteren muss sichergestellt sein, dass die zu vergleichenden Stichproben normalverteilt sind.

	Variable	**Skalierung**	**Ausprägungen**
uv	Unabhängige Variable	dichotom	{vor der U-Reihe, nach der U-Reihe}
av	Abhängige Variable	metrisch	{0..max Punktzahl}

Zur Prüfung der Normalverteilung kommt der Kolmogorov-Smirnov-Test zum Einsatz. Mit Hilfe von SPSS (Version 19) wurde dieser Test für alle Stichproben angewendet.

Im Folgenden werden die Ergebnisse des Kolmogorov-Smirnov-Test für das Gesamtergebnis und für die *Aufgabencluster 1-3* dargestellt. Hierbei wird jeweils die Normalverteilung der Stichproben von Vor- und Nachtest untersucht.

Deskriptive Statistiken

	N	Mittelwert	Standardabweichung	Minimum	Maximum
Gesamtergebnis (VT)	20	83,0750	22,53084	47,00	126,00
Gesamtergebnis (NT)	20	108,2250	19,49796	74,00	144,00

Kolmogorov-Smirnov-Anpassungstest

		Gesamtergebnis (VT)	Gesamtergebnis (NT)
N		20	20
Parameter der Normalverteilung[a,b]	Mittelwert	83,0750	108,2250
	Standardabweichung	22,53084	19,49796
Extremste Differenzen	Absolut	,156	,112
	Positiv	,156	,106
	Negativ	-,115	-,112
Kolmogorov-Smirnov-Z		,699	,503
Asymptotische Signifikanz (2-seitig)		,713	,962

a. Die zu testende Verteilung ist eine Normalverteilung.

b. Aus den Daten berechnet.

Abbildung 7.3.: Kolmogorov-Smirnov-Test (Gesamtergebnis VT / NT)

Deskriptive Statistiken

	N	Mittelwert	Standardabweichung	Minimum	Maximum
Ergebnis Cluster 1 (VT)	20	13,5250	2,73609	8,00	18,00
Ergebnis Cluster 1 (NT)	20	17,9500	5,45773	10,50	29,00

Kolmogorov-Smirnov-Anpassungstest

		Ergebnis Cluster 1 (VT)	Ergebnis Cluster 1 (NT)
N		20	20
Parameter der Normalverteilung[a,b]	Mittelwert	13,5250	17,9500
	Standardabweichung	2,73609	5,45773
Extremste Differenzen	Absolut	,169	,140
	Positiv	,101	,140
	Negativ	-,169	-,086
Kolmogorov-Smirnov-Z		,755	,624
Asymptotische Signifikanz (2-seitig)		,618	,831

a. Die zu testende Verteilung ist eine Normalverteilung.

b. Aus den Daten berechnet.

Abbildung 7.4.: Kolmogorov-Smirnov-Test (Aufgabencluster1 VT / NT)

Deskriptive Statistiken

	N	Mittelwert	Standardabweichung	Minimum	Maximum
Ergebnis Cluster 2 (VT)	20	34,9000	6,65820	22,00	46,00
Ergebnis Cluster 2 (NT)	20	36,3750	8,76427	20,00	46,50

Kolmogorov-Smirnov-Anpassungstest

		Ergebnis Cluster 2 (VT)	Ergebnis Cluster 2 (NT)
N		20	20
Parameter der Normalverteilung[a,b]	Mittelwert	34,9000	36,3750
	Standardabweichung	6,65820	8,76427
Extremste Differenzen	Absolut	,116	,225
	Positiv	,081	,124
	Negativ	-,116	-,225
Kolmogorov-Smirnov-Z		,517	1,007
Asymptotische Signifikanz (2-seitig)		,952	,263

a. Die zu testende Verteilung ist eine Normalverteilung.

b. Aus den Daten berechnet.

Abbildung 7.5.: Kolmogorov-Smirnov-Test (Aufgabencluster2 VT / NT)

Deskriptive Statistiken

	N	Mittelwert	Standardabweichung	Minimum	Maximum
Ergebnis Cluster 3 (VT)	20	34,6500	21,40530	10,50	70,50
Ergebnis Cluster 3 (NT)	20	53,9000	12,92957	33,00	76,00

Kolmogorov-Smirnov-Anpassungstest

		Ergebnis Cluster 3 (VT)	Ergebnis Cluster 3 (NT)
N		20	20
Parameter der Normalverteilung[a,b]	Mittelwert	34,6500	53,9000
	Standardabweichung	21,40530	12,92957
Extremste Differenzen	Absolut	,173	,121
	Positiv	,173	,116
	Negativ	-,130	-,121
Kolmogorov-Smirnov-Z		,772	,542
Asymptotische Signifikanz (2-seitig)		,589	,931

a. Die zu testende Verteilung ist eine Normalverteilung.

b. Aus den Daten berechnet.

Abbildung 7.6.: Kolmogorov-Smirnov-Test (Aufgabencluster3 VT / NT)

Wie in allen Tabellen erkennbar, sind die Werte für die asymptotische Signifikanz (2-seitig) p größer oder gleich 0,05. Folglich kann man annehmen, dass die Werte der getesteten Variablen hinreichend normalverteilt sind.
Da die Voraussetzungen für den t-Test bei gepaarten Stichproben erfüllt sind, ist dessen Anwendung sinnvoll um die Mittelwerte der Stichproben miteinander zu vergleichen.

7.2.4. t-Test

Die Daten der Auswertungstabellen zum Vor- und Nachtest werden mithilfe eines t-Tests daraufhin untersucht, ob sie sich statistisch signifikant voneinander unterscheiden. Hierbei soll überprüft werden, ob die Gesamtergebnisse des Nachtests höher ausfallen als die des Vortests. Der t-Test wird mit SPSS 19 durchgeführt. Hierbei wird eine Sicherheitswahrscheinlichkeit von 5% ($\alpha = 0.05$) festgelegt, welches bei Untersuchungen dieser Art üblich ist.

Statistik bei gepaarten Stichproben

	Mittelwert	N	Standardabweichung	Standardfehler des Mittelwertes
Vortest	83,0750	20	22,53084	5,03805
Nachtest	108,20000	20	19,50196	4,36077

Korrelation bei gepaarten Stichproben

	N	Korrelation	Signifikanz
Vortest & Nachtest	20	,399	,081

Test bei gepaarten Stichproben

	Gepaarte Differenzen		
	Mittelwert	Standardabweichung	Standardfehler des Mittelwertes
Vortest - Nachtest	$-25,12500$	23,17660	5,18244

Test bei gepaarten Stichproben

	Gepaarte Differenzen		T
	95% Konfidenzintervall der Differenz		
	Untere	Obere	
Vortest - Nachtest	$-35,97198$	$-14,27802$	$-4,848$

Test bei gepaarten Stichproben

	df	Sig. (2-seitig)
Vortest-Nachtest	19	,000

Die statistische Auswertung liefert ein höchstsignifikantes Ergebnis[2].
Dies bedeutet, dass die Mittelwertunterschiede zwischen dem Vor- und dem Nachtest überzufällig also systematisch sind. Da der Mittelwert für den Vortest kleiner ausfällt als der für den Nachtest kann man behaupten, dass sich die Werte zum zweiten Messzeitpunkt systematisch verbessert haben. Dementsprechend scheint das Messinstrument in der Lage zu sein, einen Kompetenzzuwachs aufzuzeigen. Folglich wird die Hypothese *H1* bestätigt.

[2] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

Im weiteren Verlauf soll mit Hypothese *H2* überprüft werden, bei welchen Kompetenzbereichen sich ein Kompetenzzuwachs abzeichnet. Hierbei wird vermutet, dass bei den Kompetenzen zur Konstruktion von Informatiksystemen (*Aufgabencluster 3*) ein besonders deutlicher Kompetenzzuwachs messbar ist, der größer als die jeweiligen Zuwächse bei den *Aufgabenclustern 1* und *2* ist.

7.3. H2 - Ergebnisse zur Konstruktion von IS im Vergleich

Im Folgenden werden die Ergebnisse von Vor- und Nachtest entsprechend der vorgenommenen thematischen Clusterung der Aufgaben des Messinstruments untersucht. Hierbei soll geprüft werden, ob wie vermutet für die Aufgaben zur Konstruktion von IS (*Cluster 3*) ein höherer Kompetenzzuwachs als bei den Aufgaben zur Dekonstruktion von Informatiksystemen (*Cluster 2*) und allgemeinen Aufgaben zu Vorgehensmodellen in der Softwaretechnik (*Cluster 1*) gemessen werden können.

7.3.1. Cluster 1 - Aufgaben zu Vorgehensmodellen in der Softwaretechnik

Zunächst sollen die Ergebnisse für das *Aufgabencluster 1* (Aufgaben zu Vorgehensmodellen in der Softwaretechnik) dargestellt werden. Wie im vorherigen Kapitel beschrieben, umfasst jener Aufgabenbereich Kompetenzen zu den unterschiedlichen Vorgehensmodellen der Softwaretechnik. Hierbei können die Lernenden verschiedene lineare und iterative Vorgehensmodelle des Software-Engineerings (linear: z.B. vereinfachtes Wasserfallmodell, ...; iteratives Vorgehen: z.B. *RUP*) zur Lösung eines komplexen Problems aus der Softwaretechnik benennen, sinnvoll absolvieren sowie beurteilen und planen.
Hierzu erfolgt zunächst die deskriptive statistische Analyse und Interpretation der Ergebnisse von Vor- und Nachtest zu *Cluster 1* im Vergleich. Ferner wird für diesen Bereich ein induktiver Mittelwertvergleich mittels t-Test vorgenommen.

Ergebnis Cluster 1
(max Punkte = 36)

Proband	**Ergebnis Vortest**		**Ergebnis Nachtest**		**Vergleich**
Kürzel	**nominal**	**prozentual**	**nominal**	**prozentual**	**Tendenz**
AN05	12,5	34,72%	13	36,11%	+**1,39**%
NI06	17	47,22%	27	75,00%	+**27,78**%
IR14	10	27,78%	12,5	33,33%	+**5,56**%
JJ13	10	27,78%	16	44,44%	+**16,67**%
OI04	8	22,22%	22	61,11%	+**38,89**%
AR16	14	38,89%	22	61,11%	+**22,22**%
OA27	14	38,89%	11	30,56%	**-8,33**%
LG18	15	41,67%	10,5	29,17%	**-12,50**%
AB26	16	44,44%	21	58,33%	+**13,89**%
EI06	14,5	40,28%	15,5	43,06%	+**2,78**%
EA26	9,5	26,39%	23	63,89%	+**37,50**%
AG15	13	36,11%	25	69,44%	+**33,33**%
ES29	14	38,89%	29	80,56%	+**41,67**%
OR01	17,5	48,61%	12	33,33%	**-15,28**%
EA05	12,5	34,72%	18,5	51,39%	+**16,67**%
IL08	14,5	40,28%	16	44,44%	+**4,17**%
AG12	15	41,67%	13	36,11%	**-5,56**%
AR01	14,5	40,28%	15,5	43,06%	+**2,78**%
UG23	18	50,00%	19	52,78%	+**2,78**%
LE06	11	30,56%	18	50,00%	**19,44**%
arithm. Mittel	**13,525**	**37,57**%	**17,95**	**49,86**%	*+**12,29***%

Tabelle 7.2.: Ergebnisse zu Aufgabencluster 1

Deskriptive statistische Analyse

Die obige Tabelle zeigt die Ergebnisse aller Probanden zu *Aufgabencluster 1*. Analog der Tabelle 7.1 zu den Gesamtergebnissen, beinhaltet die Spalte *Proband* die Kennungen für die Probanden. Innerhalb der Spalten *Vortest* und *Nachtest* werden wiederum die nominal und prozentual erreichte Punktzahl der einzelnen Probanden im Vortest und Nachtest aufgeführt. Innerhalb der Spalte *Vergleich/Tendenz* wird die Differenz zwischen anteilig erreichter Punktzahl für das *Aufgabencluster 1* im Nachtest (Minuend) und anteilig erreichter Punktzahl im Vortest (Subtrahend) berechnet.
Im Vortest haben die Probanden durchschnittlich $37,57\%$ der möglichen Punkte erreicht. Im Nachtest erreichten sie $49,86\%$ der Punkte. Folglich haben die Probanden im Nachtest im Mittel ca. $12,29\%$ mehr Punkte erreicht als im Vortest.

Um auch für das *Aufgabencluster 1* die Verteilung der Probanden im Hinblick auf die jeweils erreichte Punktzahl im Kompetenztest zu illustrieren, werden wiederum zwei Histogramme dargestellt. Auch hier wird die Häufigkeitsverteilung mit der jeweiligen Anzahl der Probanden auf der Ordinate und auf der Abszisse, die prozentual erreichte Punktzahl dargestellt. Um den Vergleich von Vor- und Nachtest zu verdeutlichen, werden die entsprechenden Histogramme nebeneinander aufgeführt.

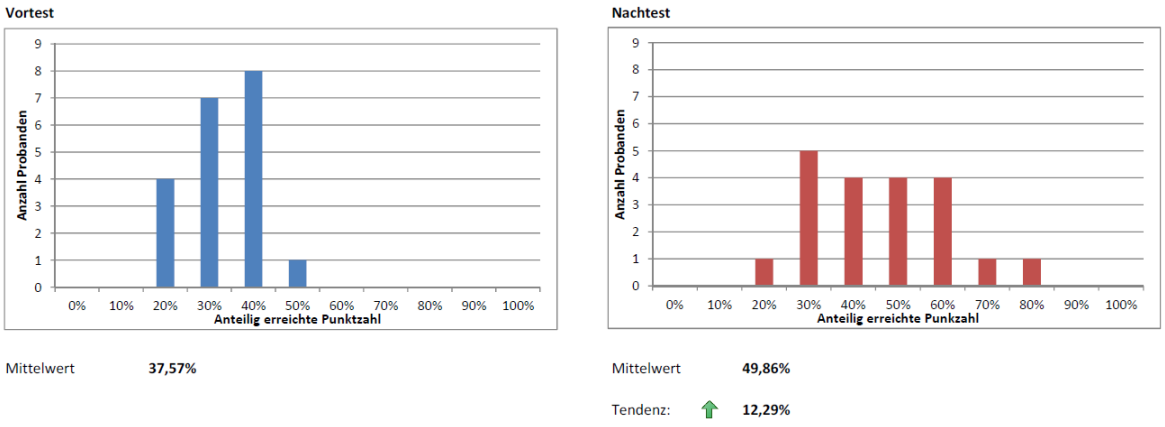


Abbildung 7.7.: Cluster 1 - Vor- und Nachtest im Vergleich

Bei Betrachtung der Histogramme zum *Aufgabencluster 1* wird deutlich, dass sich die Leistung der Probanden merklich verbessert hat. Im Vortest gab es nur einen Probanden der 50 % der Punkte oder mehr erreicht hatte. Im Nachtest hingegen hat sich eine deutliche Verbesserung ergeben: Hier haben vier Probanden mehr als 50 % der Punkte erreicht und vier weitere Probanden mehr als 60 % erreicht. Es gab sogar jeweils einen Probanden mit mehr als 70% und mehr als 80% der erreichten Punkte.
Dieser Zuwachs könnte damit zusammenhängen, dass vor, während und nach der Unterrichtseinheit stets Wert darauf gelegt wurde, die aktuelle Phase des Software-Engineering Prozesses zu besprechen und den weiteren Prozessverlauf abzustimmen. Hierbei wurde beispielsweise innerhalb der Testphase ein Rückgriff auf die Anforderungsdefinition gemacht, um zu prüfen, ob alle funktionalen Anforderungen korrekt umgesetzt wurden.

Induktive statistische Analyse

Die Daten der Auswertungstabellen zum Vor- und Nachtest wurden auch für die Ergebnisse des *Clusters 1* mithilfe des t-Tests daraufhin untersucht, ob sie sich statistisch signifikant voneinander unterscheiden.

Bei der Durchführung des t-Tests wurde wiederum eine Sicherheitswahrscheinlichkeit von 5% ($\alpha = 0.05$) festgelegt.

Statistik bei gepaarten Stichproben

	Mittelwert	**N**	**Standardabweichung**	**Standardfehler des Mittelwertes**
Cluster1 (VT)	13,5250	20	2,73609	,61181
Cluster1 (NT)	17,9500	20	5,45773	1,22039

Korrelation bei gepaarten Stichproben

	N	**Korrelation**	**Signifikanz**
Cluster1 (VT) & Cluster1 (NT)	20	,054	,822

Test bei gepaarten Stichproben

	Gepaarte Differenzen		
	Mittelwert	**Standardabweichung**	**Standardfehler des Mittelwertes**
Vortest - Nachtest	$-4,42500$	6,23503	1,39420

Test bei gepaarten Stichproben

	Gepaarte Differenzen		**T**
	95% Konfidenzintervall der Differenz		
	Untere	**Obere**	
Cluster1 (VT) - Cluster1 (NT)	$-7,34309$	$-1,50691$	$-3,174$

Test bei gepaarten Stichproben

	df	Sig. (2-seitig)
Cluster1 (VT) - Cluster1 (NT)	19	**,005**

Die statistische Auswertung liefert ein hochsignifikantes Ergebnis[3]. Das bedeutet, dass die Mittelwertunterschiede zwischen dem Vor- und dem Nachtest überzufällig also als systematisch interpretiert werden können. Da der Mittelwert für den Vortest kleiner ausfällt als der für den Nachtest kann man behaupten, dass sich die Werte zum zweiten Messzeitpunkt systematisch verbessert haben.

[3] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

7.3.2. Cluster 2 - Aufgaben zur Dekonstruktion von IS

Analog zum *Aufgabencluster 1* sollen für das *Aufgabencluster 2* die Ergebnisse der Auswertung dargestellt werden. Dieser Aufgabenbereich umfasst Kompetenzen zur Analyse von Informatiksystemen. Dies beinhaltet die Ableitung von funktionalen Anforderungen bis hin zur Feinanalyse eines Gegenstandbereichs unter Zuhilfenahme der entsprechenden *UML-Notation*. Hierbei spielen implementierungsspezifische Details bei der Modellierung eine untergeordnete Rolle und sind lediglich für das Re-Engineering des Quellcodes eines bestehenden Informatiksystems relevant.
Zunächst erfolgt die deskriptive statistische Analyse und Interpretation der Ergebnisse von Vor- und Nachtest zu *Cluster 1* im Vergleich. Ferner wird für diesen Bereich ein induktiver Mittelwertvergleich mittels t-Test durchgeführt.

Ergebnis Cluster 2
(max Punkte = 57)

Proband	**Ergebnis Vortest**		**Ergebnis Nachtest**		**Vergleich**
Kürzel	**nominal**	**prozentual**	**nominal**	**prozentual**	**Tendenz**
AN05	25	43,86%	43	75,44%	**+31,58%**
NI06	36	63,16%	32	56,14%	**-7,02%**
IR14	40	70,18%	43	75,44%	**+5,26%**
JJ13	22	38,60%	30	52,63%	**+14,04%**
OI04	41	71,93%	46	80,70%	**+8,77%**
AR16	31	54,39%	46,5	81,58%	**+27,19%**
OA27	36	63,16%	43,5	76,32%	**+13,16%**
LG18	38,5	67,54%	30,5	53,51%	**-14,04%**
AB26	46	80,70%	45	78,95%	**-1,75%**
EI06	46	80,70%	45	78,95%	**-1,75%**
EA26	35	61,40%	44	77,19%	**+15,79%**
AG15	37	64,91%	43	75,44%	**+10,53%**
ES29	27,5	48,25%	36	63,16%	**+14,91%**
OR01	36,5	64,04%	22,5	39,47%	**-24,56%**
EA05	33	57,89%	26	45,61%	**-12,28%**
IL08	23,5	41,23%	23	40,35%	**-0,88%**
AG12	40,5	71,05%	31,5	55,26%	**-15,79%**
AR01	34	59,65%	38	66,67%	**+7,02%**
UG23	32,5	57,02%	39	68,42%	**+11,40%**
LE06	37	64,91%	20	35,09%	**-29,82%**
arithm. Mittel	**34,9**	**61,23%**	**36,375**	**63,82%**	***+2,59%***

Tabelle 7.3.: Ergebnisse zu Aufgabencluster 2

Deskriptive statistische Analyse

Im Vortest haben die Probanden durchschnittlich $61,23\%$ der möglichen Punkte erreicht. Im Nachtest erreichten sie $63,82\%$ der Punkte. Folglich haben die Probanden im Nachtest im Mittel ca. $2,59\%$ mehr Punkte erreicht als im Vortest.
Zur Illustration der Verteilung der Probanden im Vergleich von Vor- zu Nachtest werden wiederum zwei Histogramme dargestellt. Auch hier wird die Häufigkeitsverteilung mit der jeweiligen Anzahl der Probanden auf der Ordinate und auf der Abszisse die prozentual erreichte Punktzahl dargestellt. Um den Vergleich von Vor- und Nachtest zu verdeutlichen werden die entsprechenden Histogramme nebeneinander aufgeführt.

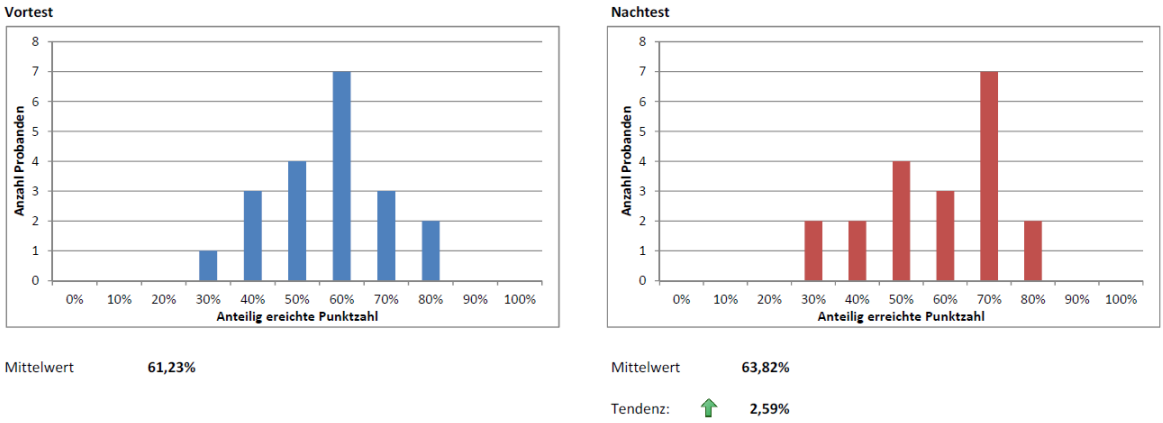


Abbildung 7.8.: Cluster 2 - Vor- und Nachtest im Vergleich

Wie innerhalb der Ergebnistabelle zu *Cluster 2* zu entnehmen ist, hat sich im Nachtest im Vergleich zum Vortest lediglich ein Zuwachs von 2,59 % ergeben. Betrachtet man die Verteilung der Probanden beim Vergleich der Histogramme, fällt auf, dass sich die Verteilung kaum geändert hat.
Das könnte aus Sicht des Autors damit zusammenhängen, dass die Lernenden gute Vorkenntnisse in der Analyse von Informatiksystemen haben und innerhalb der Unterrichtsreihe der Fokus vielmehr auf der Konstruktion von Informatiksystemen gelegen hat. Zwar wurde das Fallbeispiel *Kommissionierstation* (dargestellt durch verschiedene mediale Repräsentationsformen) analysiert, jedoch lag der Fokus eindeutig auf der Konstruktion und der Erweiterung des Systems.
Im Folgenden muss überprüft werden, ob der Kompetenzzuwachs zufällig oder systematisch war. Dementsprechend wird ein t-Test zum Mittelwertvergleich durchgeführt.

Induktive statistische Analyse

Die Daten der Auswertungstabellen zum Vor- und Nachtest wurden auch für die Ergebnisse des *Clusters 2* mithilfe eines t-Tests daraufhin untersucht, ob sich die Mittelwerte der erreichten Punktzahl statistisch signifikant voneinander unterscheiden.
Bei der Durchführung des t-Tests wurde wiederum eine Sicherheitswahrscheinlichkeit von 5% ($\alpha = 0.05$) festgelegt.

Statistik bei gepaarten Stichproben

	Mittelwert	N	Standardabweichung	Standardfehler des Mittelwertes
Cluster2 (VT)	34,9000	20	6,65820	1,48882
Cluster2 (NT)	36,3750	20	8,76427	1,95975

Korrelation bei gepaarten Stichproben

	N	Korrelation	Signifikanz
Cluster2 (VT) & Cluster2 (NT)	20	,308	,187

Test bei gepaarten Stichproben

	Gepaarte Differenzen		
	Mittelwert	Standardabweichung	Standardfehler des Mittelwertes
Vortest - Nachtest	$-1,47500$	9,23306	2,06457

Test bei gepaarten Stichproben

	Gepaarte Differenzen		T
	95% Konfidenzintervall der Differenz		
	Untere	Obere	
Cluster2 (VT) - Cluster2 (NT)	$-5,79620$	$-2,84620$	$-,714$

Test bei gepaarten Stichproben

	df	Sig. (2-seitig)
Cluster2 (VT) - Cluster2 (NT)	19	**,484**

Die statistische Auswertung liefert ein nicht signifikantes Ergebnis[4]. Das bedeutet, dass die Mittelwertunterschiede zwischen dem Vor- und dem Nachtest wahrscheinlich zufällig sind. Obwohl der Mittelwert für den Vortest kleiner ausfällt als der für den Nachtest kann nicht behauptet werden, dass sich die Ergebnisse des *Clusters 2* zum Nachtest systematisch verbessert haben.

[4] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

7.3.3. Cluster 3 - Aufgaben zur Konstruktion von IS

Abschließend werden die Ergebnisse für das *Aufgabencluster 3* (Aufgaben zur Konstruktion von Informatiksystemen) dargestellt. Dieser Aufgabenbereich fokussiert Kompetenzen zum Design und zur Gestaltung eines neuen Informatiksystems. Dies umfasst im Gegensatz zu den Aufgaben in *Cluster 2* vorrangig das konkrete Lösungsdesign eines Informatiksystems bzw. dessen Bestandteilen und die konkrete Implementierung auf Grundlage von Design-Modellen in *UML-Notation*. Hierbei liegt der Fokus bei der Modellierung auf Modellen mit implementierungsspezifischen Details. Neben der Konstruktion von Informatiksystemen adressieren die Aufgaben aus *Cluster 3* ebenso den Test eines Informatiksystems.
Zur Analyse der Testergebnisse im *Aufgabencluster 3* erfolgt zunächst die deskriptive statistische Analyse und Interpretation der Ergebnisse von Vor- und Nachtest zu *Cluster 1* im Vergleich. Ferner wird auch für diesen Bereich ein induktiver Mittelwertvergleich mittels t-Test durchgeführt.

Ergebnis Cluster 3
(max Punkte = 79)

Proband	**Ergebnis Vortest**		**Ergebnis Nachtest**		**Vergleich**
Kürzel	**nominal**	**prozentual**	**nominal**	**prozentual**	**Tendenz**
AN05	39,5	50,00%	40	50,63%	**+0,63%**
NI06	46,5	58,86%	46,5	58,86%	**+0,00%**
IR14	11,5	14,56%	35,5	44,94%	**+30,38%**
JJ13	15	18,99%	50	63,29%	**+44,30%**
OI04	35	44,30%	46	58,23%	**+13,92%**
AR16	10,5	13,29%	37	46,84%	**+33,54%**
OA27	26,5	33,54%	63,5	80,38%	**+46,84%**
LG18	18	22,78%	33	41,77%	**18,99%**
AB26	12,5	15,82%	63,5	80,38%	**+64,56%**
EI06	14	17,72%	58	80,38%	**+55,70%**
EA26	69,5	87,97%	64,5	81,65%	**-6,33%**
AG15	67	84,81%	76	9,20%	**+11,39%**
ES29	54	68,35%	65	82,28%	**+11,39%**
OR01	22	27,85%	60	75,95%	**+48,10%**
EA05	19,5	24,68%	55,5	70,25%	**+45,57%**
IL08	14,5	18,35%	45	56,96%	**+38,61%**
AG12	70,5	89,24%	57,5	72,78%	**-16,46%**
AR01	38,5	48,73%	64,5	81,65%	**+32,91%**
UG23	42,5	53,80%	75,5	95,57%	**+41,77%**
LE06	66	83,54%	41,5	52,53%	**-31,01%**
arithm. Mittel	**34,65**	**43,86%**	**53,9**	**68,23%**	+24,37%

Tabelle 7.4.: Ergebnisse zu Aufgabencluster 3

Deskriptive statistische Analyse

Im Vortest haben die Probanden durchschnittlich 43, 86% der möglichen Punkte erreicht. Im Nachtest erreichten sie 68, 23% der Punkte. Folglich haben die Probanden im Nachtest im Mittel ca. 24, 37% mehr Punkte erreicht als im Vortest.
Wie erwartet hat bei den Ergebnisse zum *Aufgabencluster 3* der größte Zuwachs an Punkten im Nachtest im Vergleich zum Vortest stattgefunden.
Um die Verteilung der Probanden im Hinblick auf die jeweils erreichte Punktzahl im Kompetenztest zu illustrieren, zeigen die bereits für die anderen Aufgabencluster verwendeten Histogramme die Häufigkeitsverteilung. Wie zuvor enthalten sie auf der Ordinate die jeweilige Anzahl der Probanden und auf der Abszisse die prozentual erreichte

Punktzahl. Um darüber hinaus die Verteilung von Vor- und Nachtest zu vergleichen, werden die beiden Histogramme für Vortest (Histogramm mit blauen Balken) und Nachtest (Histogramm mit roten Balken) nebeneinander dargestellt.

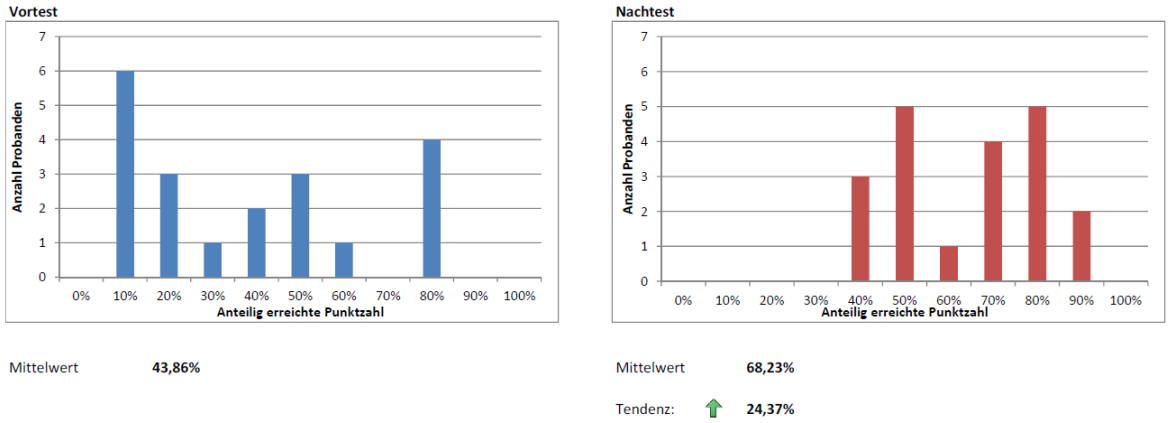


Abbildung 7.9.: Cluster 3 - Vor- und Nachtest im Vergleich

Bei Betrachtung der Histogramme wird deutlich, dass sich die Leistung der Probanden erheblich verbessert hat. Im Vortest haben die Hälfte der Probanden weniger als 40 % der Punkte erreicht. Im Nachtest hingegen haben alle Probanden deutlich besser abgeschnitten: In diesem Zusammenhang hat jeder Proband mehr als 40 % der Punkte erreicht. Dieser Zuwachs von über 24% der Punkte im Nachtest bestätigt die Vermutung, dass das Instrument genau bei diesem Aufgabencluster einen besonders hohen Kompetenzzuwachs aufzeigt. Bestärkt wird diese Folgerung auch durch die Tatsache, dass im Nachtest mehr als die Hälfte der Probanden 70% oder mehr der Punkte erreicht haben und somit eine fast vollständige Bearbeitung der Aufgaben des *Clusters 3* geleistet haben.
Dieser erhebliche Zuwachs lässt sich dadurch erklären, dass die Unterrichtsreihe besonders jene Modellierungskompetenzen anspricht, die die Entwicklung eines Informatiksystems adressieren. Als zentraler thematischer Schwerpunkt waren die Probanden hier angehalten, einen komplexen und zeitlich umfänglichen Auftrag zur Erweiterung der *Kommissionierstation* durchzuführen. Hier mussten Änderungen an der Softwarearchitektur modelliert werden, um neue Sensoren und Aktoren in das Informatiksystem zu integrieren. Ferner galt es in diesem Zusammenhang unbekannte Programmschnittstellen anzusprechen und die dazugehörigen Software-Module sinnvoll in das Gesamtsystem zu integrieren. Die Lernenden waren weiterhin gefordert, die daraus resultierenden neuen Quellcode Fragmente so in das Gesamtsystem zu integrieren, dass die Gesamtlösung weiterhin lauffähig bleibt.

Die Probanden erfuhren durch die dargebotenen unterschiedlichen medialen Repräsentationsformen einen Perspektivwechsel auf das Informatiksystem und mussten die technischen Anpassungen im Modell auf die Modellanpassungen und Quellcodeanpassungen überführen. Die Probanden haben in diesem Kontext auch technische Änderungen am *LEGO Mindstorms-Modell* vorgenommen und in Design-Modelle (z.B. State-Charts) mit implementierungsspezifischen Details überführt. Die Designmodelle wurden wiederum in objektorientierten Quellcode übersetzt. Konkret haben die Probanden hierbei Klassendiagramme mit Assoziationen und Vererbungsstrukturen in *Java-Quellcode* übersetzt.
Insgesamt ergibt sich also bei dem Lehr-/Lernarrangement ein sehr starker Fokus auf die Konstruktion und Entwicklung von Informatiksystemen.
Unter der Annahme, dass die durchgeführte Unterrichtsreihe Modellierungskompetenz fördert, könnten diese Ergebnisse einen Hinweis darauf geben, dass sich mit dem verwendeten Instrument ein Kompetenzzuwachs messen lässt.

Induktive statistische Analyse

Die Daten der Auswertungstabellen zum Vor- und Nachtest wurden auch für die Ergebnisse des *Clusters 3* mithilfe eines t-Tests daraufhin untersucht, ob sich die Mittelwerte der erreichten Punktzahl statistisch signifikant voneinander unterscheiden.
Bei der Durchführung des t-Tests wurde wieder eine Sicherheitswahrscheinlichkeit von 5% ($\alpha = 0.05$) festgelegt.

Statistik bei gepaarten Stichproben

	Mittelwert	**N**	**Standardabweichung**	**Standardfehler des Mittelwertes**
Cluster3 (VT)	34,6500	20	21,40530	4,78637
Cluster3 (NT)	53,9000	20	12,92957	2,89114

Korrelation bei gepaarten Stichproben

	N	**Korrelation**	**Signifikanz**
Cluster3 (VT) & Cluster3 (NT)	20	,390	,089

Test bei gepaarten Stichproben

	Gepaarte Differenzen		
	Mittelwert	**Standardabweichung**	**Standardfehler des Mittelwertes**
Cluster3 (VT) - Cluster3 (NT)	−19,25000	20,22993	4,52355

Test bei gepaarten Stichproben

	Gepaarte Differenzen		T
	95% Konfidenzintervall der Differenz		
	Untere	**Obere**	
Cluster1 (VT) - Cluster1 (NT)	−28,71790	−9,78210	−4,256

Test bei gepaarten Stichproben

	df	Sig. (2-seitig)
Cluster3 (VT) - Cluster3 (NT)	19	**,000**

Die statistische Auswertung des *Aufgabenclusters 3* liefert ein höchstsignifikantes Ergebnis[5]. Das bedeutet, dass die Mittelwertunterschiede zwischen dem Vor- und dem Nachtest systematisch sind. Da der Mittelwert für den Vortest hier deutlich kleiner ausfällt als der für den Nachtest kann man behaupten, dass sich die Werte zum zweiten Messzeitpunkt systematisch verbessert haben. Dementsprechend scheint das Messinstrument in der Lage zu sein, einen Kompetenzzuwachs aufzuzeigen.

7.4. Zusammenfassung

Im Rahmen dieses Kapitels wurden die Ergebnisse zur Erprobung des Messinstruments beschrieben.
Zur Überprüfung der Hypothesen *H1* und *H2* erfolgte sowohl eine deskriptive statistische Analyse der Ergebnisse als auch ein induktives Verfahren in Form eines t-Tests, um die statistische Signifikanz der Ergebnisse des Nachtests im Vergleich zum Vortest zu untersuchen.
Im Hinblick auf das Gesamtergebnis und den Mittelwertvergleich von Vor- und Nachtest, bei dem sich ein Zuwachs von 14,62 % ergeben hat, zeigen die deskriptiven und induktiven Ergebnisse, dass das Messinstrument in der Lage ist, die positiven Veränderungen beim

[5] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

Nachtest im Vergleich zum Vortest zu messen. Demzufolge wurde die Hypothese *H1* akzeptiert.

Test bei gepaarten Stichproben

		T	df	Sig. (2-seitig)
Paaren 1	Gesamtergebnis (VT) - Gesamtergebnis (NT)	-4,847	19	,000
Paaren 2	Ergebnis Cluster 1 (VT) - Ergebnis Cluster 1 (NT)	-3,174	19	,005
Paaren 3	Ergebnis Cluster 2 (VT) - Ergebnis Cluster 2 (NT)	-,714	19	,484
Paaren 4	Ergebnis Cluster 3 (VT) - Ergebnis Cluster 3 (NT)	-4,256	19	,000

Abbildung 7.10.: Ergebnisse des t-Test im Vergleich 1/2

Bei Betrachtung der Ergebnisse zu den *Aufgabenclustern 1* bis *3* konnten die in *H2* aufgestellten Vermutungen bestätigt werden, dass die Unterrichtsreihe insbesondere diejenigen Modellierungskompetenzen fördert, die zur Konstruktion von Informatiksystemen erforderlich sind. Hierbei hat sich innerhalb der deskriptiven statistischen Analyse gezeigt, dass beim *Aufgabencluster 1* (Aufgaben zu Vorgehensmodellen in der Softwaretechnik) ein durchschnittlicher Zuwachs von 12,29% stattgefunden hat. Dieser Zuwachs könnte damit zusammenhängen, dass während der Unterrichtsreihe stets Wert darauf gelegt wurde, die aktuelle Phase des Software-Engineering Prozesses zu besprechen und den weiteren Prozessverlauf abzustimmen. Beim *Aufgabencluster 2* (Aufgaben zur Dekonstruktion von IS) wurde lediglich ein Zuwachs von 2,59% gemessen. Dies könnte insbesondere daran gelegen haben, dass die Schülerinnen und Schüler gute Vorkenntnisse in der Analyse von Informatiksystemen hatten und innerhalb der Unterrichtsreihe der Fokus eher auf der Konstruktion von Informatiksystemen gelegen hat. Wie erwartet wurde der deutlichste Zuwachs bei dem *Aufgabencluster 3* (Aufgaben zur Konstruktion von IS) gemessen. Hierbei haben sich die Probanden im Nachtest durchschnittlich um 24,37% im Gegensatz zum Vortest verbessert. Der enorme Zuwachs erklärt sich durch die starke Fokussierung der Unterrichtsreihe auf Modellierungskompetenzen, die mit der Entwicklung eines Informatiksystems zu tun haben. Der größte thematische Schwerpunkt war die Anpassung und Entwicklung der technisch erweiterten *Kommissionierstation*.
Neben diesen Ergebnissen und den Rückschlüssen aus der deskriptiven statistischen Ana-

lyse, wurde auch mit dem t-Test gezeigt, dass der Zuwachs bei dem *Aufgabencluster 3* im Vergleich zu den *Clustern 1* und *2* das höchste Signifikanzniveau aufweist. Wie in der folgenden Tabelle dargestellt, ist der Zuwachs bei den Aufgaben des *Clusters 3* höchstsignifikant, wobei der Zuwachs beim *Cluster 1* signifikant und der Zuwachs von *Cluster 2* keine Signifikanz aufweist[6].
Da sich dieses Signifikanzniveau auch implizit in der Gewichtung der einzelnen Themen innerhalb der Unterrichtsreihe widerspiegelt, ist davon auszugehen, dass das Messinstrument in dem vorliegenden Setting in der Lage ist, messtechnisch zu differenzieren. Folglich wird auch die Hypothese *H2* akzeptiert.

Test bei gepaarten Stichproben

		T	df	Sig. (2-seitig)
Paaren 1	Gesamtergebnis (VT) - Gesamtergebnis (NT)	-4,847	19	,000
Paaren 2	Ergebnis Cluster 1 (VT) - Ergebnis Cluster 1 (NT)	-3,174	19	,005
Paaren 3	Ergebnis Cluster 2 (VT) - Ergebnis Cluster 2 (NT)	-,714	19	,484
Paaren 4	Ergebnis Cluster 3 (VT) - Ergebnis Cluster 3 (NT)	-4,256	19	,000

Abbildung 7.11.: Ergebnisse des t-Test im Vergleich 2/2

[6] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

8. Fazit und Weiterführende Forschungsfragen

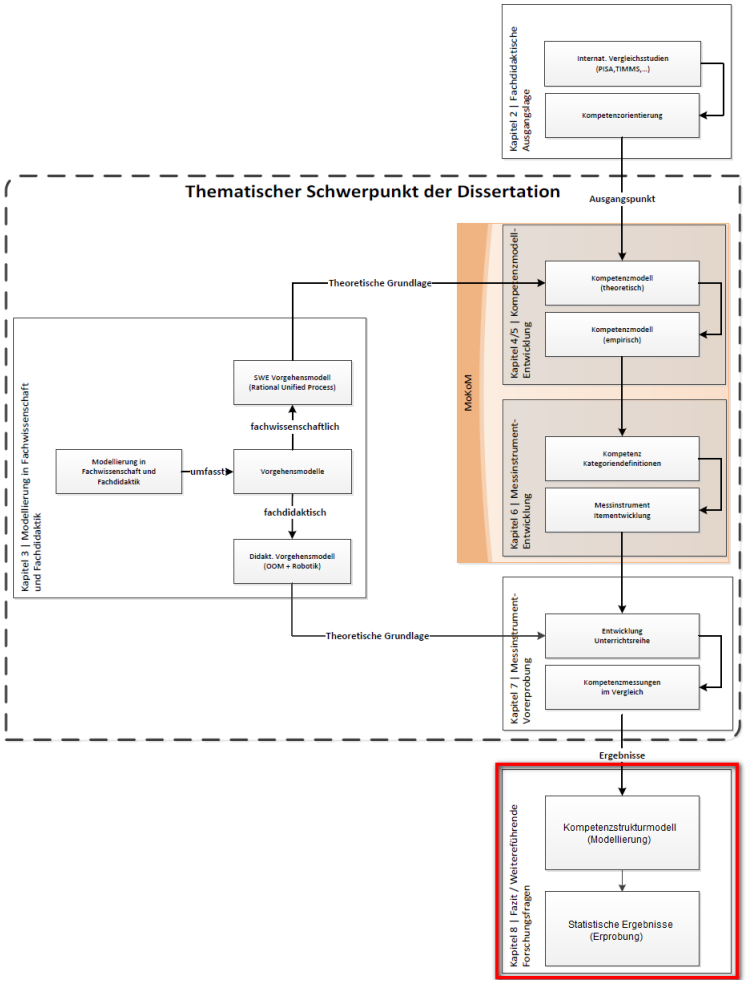


Abbildung 8.1.: Kapitel 8 im Gesamtkontext der Arbeit

8.1. Zu Forschungsfrage 1

„Welche kognitiven und nicht-kognitiven Facetten umfasst informatische Modellierungskompetenz? (Entwicklung eines Kompetenzstrukturmodells)“

Im Rahmen der Kapitel 4 und 5 wurde die Entwicklung eines empirisch gesicherten Kompetenzmodells für informatisches Modellieren aufgezeigt. Dieses umfasst die kognitiven und nicht-kognitiven Facetten informatischer Modellierungskompetenz. Neben der theoretisch-normativen Ableitung von Kompetenzaspekten zur Modellierung konnten mit Hilfe der Experteninterviews wertvolle Hinweise zur Überarbeitung der Dimensionen erlangt werden. Die folgende Abbildung 8.2 illustriert den Prozess der empirischen Verfeinerung der Kompetenzdimension *K1.3 Systemgestaltung*. Die Abbildung 8.3 stellt das gesamte empirisch überarbeitete Kompetenzmodell für informatische Modellierung und Systemverständnis dar. Die für die informatische Modellierung relevanten Dimensionen, deren Entwicklungsprozess in Abbildung 8.2 dargestellt wurde, ist in Abbildung 8.3 rot hervorgehoben.
Das resultierende, empirisch verfeinerte Kompetenzstrukturmodell und die dazugehörigen Kategoriendefinitionen waren eine wichtige Voraussetzung zur Entwicklung von Aufgabenitems sowie deren inhaltlicher Fokussierung.

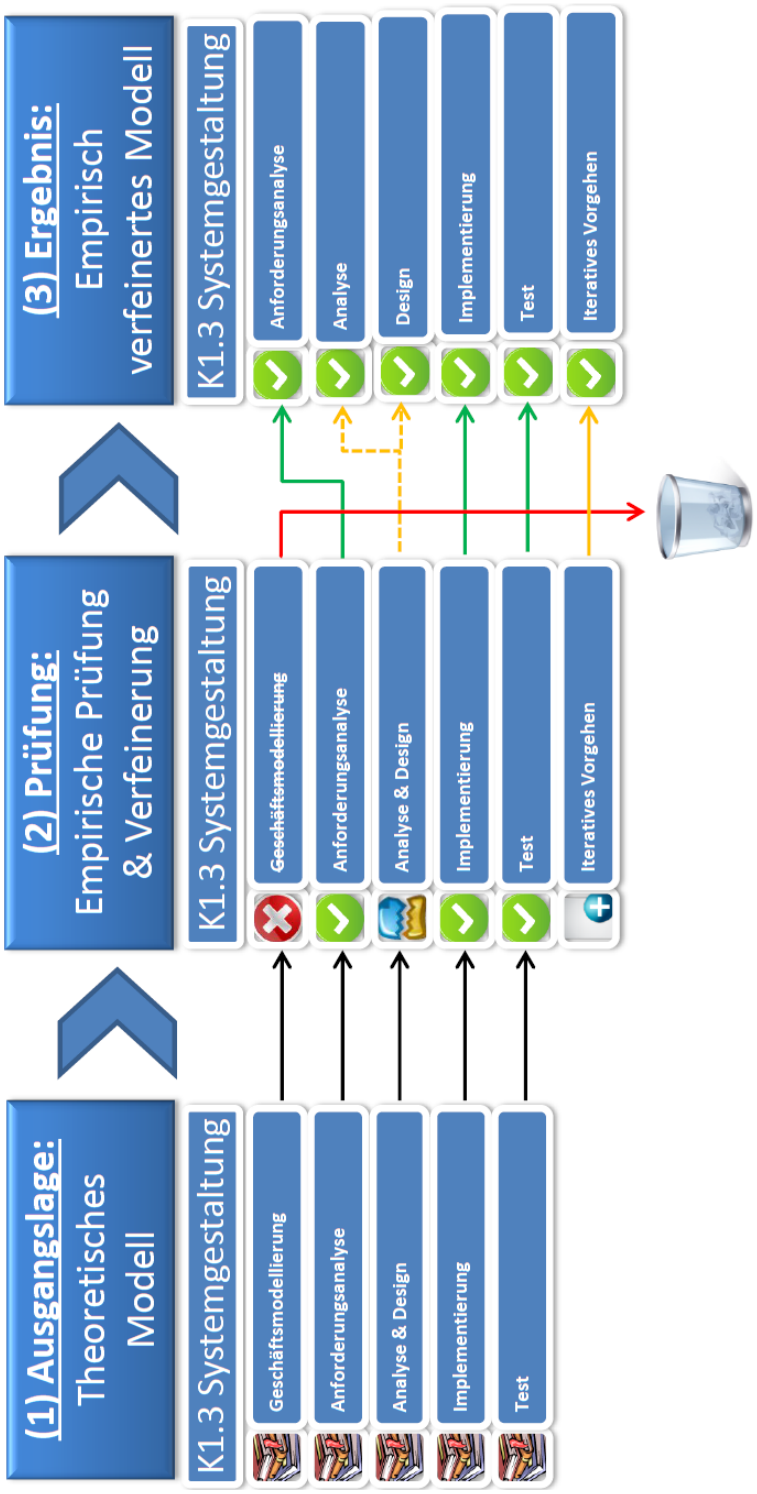


Abbildung 8.2.: Prozess zur empirischen Verfeinerung des Teilmodells *Modellierung*

Abbildung 8.3.: Empirisch Verfeinertes Kompetenzstrukturmodell

8.2. Zu Forschungsfrage 2

„Lässt sich ein Zuwachs an informatischer Modellierungskompetenz messbar machen? (Entwicklung und Erprobung eines Messinstruments)“

Innerhalb des Kapitels 6 wurde auf Grundlage des empirisch gesicherten Kompetenzmodells die Entwicklung des Messinstruments für informatische Modellierungskompetenz dargestellt. Neben der eigentlichen Entwicklung kam die Frage auf, wie eine erste Erprobung des Instruments erfolgen könnte. Dementsprechend wurde ein Lehr-/Lernarrangement auf theoretischer Grundlage des *Informatik Lernlabors* entwickelt.
Unter der Annahme, dass das entwickelte Lehr-/Lernarrangement Kompetenzen in der informatischen Modellierung fördert, wurden die folgenden Hypothesen zur Tauglichkeit des Instruments, Kompetenzzuwächse zu messen, angeführt.

Hypothese H1:

Das Messinstrument zeigt einen systematischen Kompetenzzuwachs der Lernenden beim Nachtest im Vergleich zum Vortest auf.

Die deskriptive statistische Analyse zeigt eine durchschnittliche Verbesserung der Punktzahl der Probanden von Nachtest im Vergleich zu Vortest von 14,62%. Darüber hinaus liefert auch die statistische Auswertung mittels des t-Tests ein höchstsignifikantes Ergebnis[1]. Da der Mittelwert für den Vortest geringer ist als der des Nachtests, lässt sich behaupten, dass sich die Werte zum zweiten Messzeitpunkt systematisch verbessert haben. Somit hat sich das Instrument im Kontext des beschriebenen Untersuchungssettings als tauglich erwiesen Kompetenzzuwächse zu messen. Die Hypothese *H1* wird somit bestätigt.
Neben dem Gesamtergebnis der Erprobung wurde geprüft, welche Kompetenzbereiche einen besonders hohen Kompetenzzuwachs aufzeigen und welche weniger.
In Anlehnung an die Phasen des *ILL* wurde eine phasenabhängige Bündelung der Aufgaben in sog. Aufgabencluster vorgenommen. Anhand der Inhalte und der zu fördernden Kompetenzen der Unterrichtsreihe wurden drei Aufgaben-Cluster gebildet die den einzelnen Phasen der in der Lerneinheit geförderten Kompetenzbereiche zugeordnet wurden.

1. *Cluster 1*: Allgemeine Kompetenzen zu Vorgehensmodellen in der Softwaretechnik (Bündelung der Aufgaben 1,2,3)

[1] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

2. *Cluster 2*: Kompetenzen für die Dekonstruktion und Analyse von Informatiksystemen (Bündelung der Aufgaben 4,5,6)

3. *Cluster 3*: Kompetenzen für die Konstruktion von Informatiksystemen (Bündelung der Aufgaben 7,8,9,10)

Aufgrund der besonderen Fokussierung und der Konstruktionsphase wurde ein Kompetenzzuwachs insbesondere bei *Cluster 3* im Vergleich zu *Cluster 1* und *Cluster 2* erwartet. Demzufolge sollte das Instrument insbesondere für diesen Kompetenzbereich einen deutlichen Kompetenzzuwachs messen können. Die Hypothese *H2* fasst den Sachverhalt in eine Aussage, die es zu überprüfen galt, zusammen:

Hypothese H2:

Das Messinstrument zeigt einen systematischen Kompetenzzuwachs bei dem* Aufgabencluster 3 *(Konstruktion von IS) auf. Dieser ist größer als der Kompetenzzuwachs bei den* Aufgabenclustern 1 *(allgemeine Aufgaben zu Vorgehensmodellen in der Softwaretechnik) und 2 (Dekonstruktion von IS).

Sowohl die deskriptive statistische Analyse als auch die Ergebnisse der t-Tests führen zur Bestätigung der Hypothese *H2*. Bei Betrachtung der Tabelle 8.1 ergibt sich bei den Probanden ein durchschnittlicher Zuwachs von 24,37% der erreichbaren Punkte. Dieser Wert liegt (wie erwartet) deutlich über dem Zuwachs beim *Cluster 1* von 12,29% und beim *Cluster 2* von 2,59%. Diese durchschnittlichen Zuwächse wurden darüber hinaus mit dem t-Test untersucht. Wie in Tabelle 8.1 dargestellt zeigt der t-Test, dass der Zuwachs beim *Cluster 3* nicht zufällig sondern systematisch ist. Hierbei ergibt sich für das *Cluster 3* ein höchstsignifikantes Niveau[2], wo hingegen *Cluster 1* ein hochsignifikantes Niveau und *Cluster 2* als nicht signifikant einzustufen ist.

[2] $\alpha \leq 0,001$: höchst signifikant | $\alpha \leq 0,01$: hoch signifikant | $\alpha \leq 0,05$: signifikant

	Cluster1	**Cluster2**	**Cluster3**
Vortest	37,57%	61,23%	43,86%
Nachtest	49,86%	63,82%	68,23%
Tendenz	**+12,29%**	**+2,59%**	**+24,37%**
Sign. (t-Test)	**,005**	**,484**	**,000**

Tabelle 8.1.: Statistischer Vergleich von Vor- und Nachtest für die Cluster 1-3

Auf dieser Grundlage wird auch die Hypothese *H2* bestätigt.
Die Bestätigung der Hypothesen *H1* und *H2* beantwortet die Forschungsfrage 2: Das Instrument scheint in dem vorgestellten Setting Kompetenzzuwächse im Bereich der objektorientierten Modellierung messen zu können. Dies gilt es allerdings in weiterführenden Forschungsvorhaben in breiten Erprobungen zu überprüfen. Im Rahmen dieser Dissertation sind die Ergebnisse als Indiz für die Tauglichkeit des Instruments, welches weiterer Überarbeitung bedarf, anzusehen.

8.3. Weiterführende Forschungsfragen

Die vorliegende Arbeit beschreibt die Resultate einer ersten Erprobung des Messinstruments. Hierbei ist es möglich einen statistisch signifikanten Kompetenzzuwachs im Vergleich von Vor- und Nachtest zu messen. Voraussetzung ist, dass die verwendete Unterrichtsreihe Modellierungskompetenz fördert.
Dieses noch nicht repräsentative Ergebnis muss in einer künftigen breiten Erprobung verifiziert werden. Hierfür bedarf es einer größeren repräsentativen Stichprobenzahl. Es ist somit erforderlich, mehr Lehrende und Schulklassen für weitergehende Kompetenzmessungen zu gewinnen.
Die deskriptive statistische Auswertung der beiden Kompetenzmessungen hat gezeigt, dass das Messinstrument hinsichtlich seines Umfangs überarbeitet und optimiert werden muss. Insbesondere die letzten beiden Aufgaben des Instruments lassen erkennen, dass die Aufgabensammlung offenbar zu umfangreich und innerhalb der vorgegebenen Zeit von 90 Minuten kaum zu lösen ist. Hier muss in weiteren Arbeitsschritten eine Kürzung des Instruments erarbeitet werden. Wahrscheinlich ist es sinnvoll, dass Messinstrument in geeigneter Weise in Booklet-Form aufzuteilen. In diesem Zusammenhang sollte auch geprüft werden, welche Aufgabentypen sich als besonders geeignet erwiesen haben und somit vermehrt zu berücksichtigen sind.
Die Unterrichtsreihe *Kommissionierstation* ist in der hochschuldidaktischen Praxis erfolg-

reich erprobt worden; die für die gymnasiale Oberstufe modifizierte Unterrichtsreihe muss hinsichtlich ihrer Kompetenzförderlichkeit in weiteren Schritten überprüft werden. Hierbei sollten die verwendeten medialen Repräsentationsformen sowie die lernerzentrierten Instruktionsformen mit berücksichtigt werden, die bei der Planung der Unterrichtsreihe als methodische Grundlage verwendet werden.
Im Gegensatz zu der ersten *MoKoM* Projektperiode haben sich die Entwicklungs- und Forschungsarbeiten auf die Konzeption und empirisch gestützte Überprüfung und Generierung eines Kompetenzstrukturmodells für informatisches Systemverständnis und Modellieren sowie die Konstruktion eines Instrumentariums zur Kompetenzmessung konzentriert. Aufbauend auf diesen Arbeiten soll in der zweiten Förderperiode das Strukturmodell im Hinblick auf ein Kompetenzniveaumodell sowie in Ausschnitten auch in Bezug auf ein Kompetenzentwicklungsmodell weiterentwickelt werden. Darüber hinaus soll das Instrumentarium zur Kompetenzmessung breit erprobt und weiterentwickelt sowie zwei Ansätze zur Förderung des Kompetenzerwerbs mit einem Schwerpunkt auf informatisches Systemverständnis einerseits und informatisches Modellieren andererseits hinsichtlich ihrer Wirkungen evaluiert werden.
Im Rahmen von fünf Arbeitspaketen werden daher folgende Zielsetzungen bei *MoKoM II* verfolgt:

1. Breite empirische Erprobung des Instrumentariums zur Kompetenzmessung und Überprüfung der psychometrischen Gütekriterien des Verfahrens (Arbeitspaket PI),

2. Entwicklung eines Kompetenzniveaumodells für Informatisches Systemverständnis und Modellieren (Arbeitspaket PII),

3. Analyse von bildungsbiographischen und unterrichtsbezogenen Einflussfaktoren des Kompetenzerwerbs (Arbeitspaket PIII),

4. Konzeption und empirische Überprüfung eines Kompetenzentwicklungsmodells zum Informatischen Systemverständnis sowie eines Kompetenzentwicklungsmodells zum Informatischen Modellieren (Arbeitspaket PIV1.1, PIV1.2, PIV2.1 und PIV2.2),

5. Konzeption und Evaluation von Lehr-/Lernarrangements zur Förderung des Kompetenzerwerbs beim informatischen Modellieren (Arbeitspaket PIV2.1 und PIV2.3),

6. Erarbeitung von Implikationen aus den vorangegangenen empirischen Untersuchungsschritten für die fachdidaktische Theoriebildung und Weiterentwicklung von Konzepten zur kompetenzförderlichen Unterrichtsgestaltung und praxisgerechten diagnostischen Ansätzen (Arbeitspaket PV).

Um die vierte und fünfte Zielsetzung zu erreichen, werden an den beiden Standorten (Paderborn und Siegen) verschiedene informatikdidaktische Ansätze zur Förderung des informatischen Systemverständnisses einerseits und des informatischen Modellierung andererseits genutzt und in die Schulpraxis implementiert. Dadurch werden unterschiedliche Teilbereiche des gemeinsamen Kompetenzmodells unter jeweils spezifischen lerninhaltlichen und informatikdidaktischen Foki hinsichtlich Fragen zur Förderung des Kompetenzerwerbs in den Blick genommen und empirisch überprüft.

Literaturverzeichnis

[Tucker2006 2006] Tucker, A. (Hrsg.): *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee,. 2nd Edition.* 2006

[ACM 2008] ACM: Computer Science Curriculum 2008: An Interim Revision of CS 2001 Report from the Interim Review Task / Association for Computing Machinery IEEE Computer Society. 2008 (December). – Forschungsbericht

[Agile Alliance 2013] Agile Alliance: Agile Alliance Website. (2013). – URL `http://www.agilealliance.org/ (geprüft: 18.02.2013)`

[Balzert 2000] Balzert, H.: *Lehrbuch der Software-Technik: Software-Entwicklung. 2. Auflage.* Heidelberg : Spektrum Akademischer Verlag, 2000

[Baumann 1996] Baumann, R.: *Didaktik der Informatik. 2. vollständig neu bearbeitete Auflage.* Ernst Klett Verlag, 1996

[Baumert et al. 2000] Baumert, J. ; Bos, W. ; Lehmann, R.: *TIMSS/III. Dritte Internationale Mathematik- und Naturwissenschaftsstudie. Mathematische und naturwissenschaftliche Bild, Bd.2, Mathematische und phys ... Kompetenzen am Ende der gymnasialen Oberstufe.* Leske + Budrich Verlag, 2000

[Beck und Klieme 2007] Beck, B. ; Klieme, E.: *Sprachliche Kompetenzen: Konzepte und Messung: DESI-Studie (Deutsch Englisch Schülerleistungen International).* Weinheim u.a. : Beltz, 2007

[Bennedsen und Caspersen 2005] Bennedsen, J. ; Caspersen, M.: Revealing the programming process. In: *ACM SIGCSE Bulletin* (2005), S. S. 186–190

[Bischofberger und Pomberger 1992] Bischofberger, W. ; Pomberger, G.: Prototyping-oriented software development: Concepts and tools. Springer, 1992

[Brauer und Brauer 1992] Brauer, W. ; Brauer, U.: Wissenschaftliche Herausforderungen für die Informatik: Änderungen von Forschungszielen und Denkgewohnheiten. In: *Informatik cui bono.* Springer, 1992, S. S. 11–19

[Brinda 2004] Brinda, T.: *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II.* Universität Siegen, Didaktik der Informatik und E-Learning, Dissertation, 2004

[Bussmann und Heymann 1987] Bussmann, H. ; Heymann, H.: Computer und Allgemeinbildung. In: *Neue Sammlung I* (1987), S. S. 2–39

[Chan et al. 2010] Chan, C. ; Tsui, M.S. ; Chan, M. ; Hong, Joe H. ; Joe, H.: Assessment & Evaluation in Higher Education Applying the Structure of the Observed Learning Outcomes (SOLO) Taxonomy on Student ' s Learning Outcomes : An empirical study. (2010), Nr. November 2012, S. S. 37–41

[Claus und Schwill 2006] Claus, V. ; Schwill, A.: *Duden Informatik A-Z. Fachlexikon für Studium, Ausbildung und Beruf.* Mannheim : Dudenverlag, 2006

[Cognition and Technology Group at Vanderbilt 1994] Cognition and Technology Group at Vanderbilt: Multimedia environments for enhancing student learning in mathematics. In: *Technology based learning environments. Psychological and educational foundations.* (1994)

[Collins 1989] Collins, A.: Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In: *Knowing, Learning and Instruction* (1989)

[Denning 2003] Denning, P.: Great principles of computing. In: *Communications of the ACM* (2003)

[Denning 2007] Denning, P.: Computing is a natural science. In: *Communications of the ACM* (2007)

[Diethelm 2007] Diethelm, I.: *Strictly models and objects first-Unterrichtskonzept und-methodik für objektorientierte Modellierung im Informatikunterricht.* Universität Kassel, Dissertation, 2007

[Diethelm et al. 2005] Diethelm, I. ; Geiger, L. ; Zündorf, A.: Teaching modeling with objects first. In: *IFIP World Conference on Computers in Education* (2005)

[Dietzel und Rinkens 2001] Dietzel, R. ; Rinkens, T.: Eine Einführung in die Objektorientierung mit Lego Mindstorms Robotern Erfahrungsbericht aus dem Unterricht. INFOS 2001, 2001, S. 193–199

[Dohmen et al. 2009] Dohmen, M. ; Engbring, D. ; Magenheim, J.: Kreativer Einstieg in die Programmierung Alice im Informatik-Anfangsunterricht. In: *INFOS 2009* (2009), S. S. 329ff

[Drieschner 2009] Drieschner, E.: *Bildungsstandards praktisch.* Verlag für Sozialwissenschaften, 2009

[Droeschel 1998] Droeschel, W.: Inkrementelle und objektorientierte Vorgehensweise mit dem VModell 97. Muenchen, Wien : Oldenbourg Verlag, 1998

[Ebert 2005] Ebert, J.: Zitat aus der Podiumsdiskussion SSoftware-Entwicklung und Modellierungäuf dem Workshop Modellierung 2005. In: Paech, B. (Hrsg.) ; Desel, J. (Hrsg.): *Workshop Modellierung 2005*, 2005

[Europäische Gemeinschaften 2008] Europäische Gemeinschaften: The European Qualifications Framework for Lifelong Learning (EQF) NC-30-08-266-EN-P Europäischer Qualifikationsrahmen Der Europäische Qualifikationsrahmen für lebenslanges Lernen (EQR). (2008). ISBN 9789279084720

[Fakultätentag Informatik (Hrsg) 2004] Fakultätentag Informatik (Hrsg): *Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten.* 2004

[Fieber et al. 2008] Fieber, F. ; Huhn, M. ; Rumpe, B.: Modellqualität als Indikator für Softwarequalität: eine Taxonomie. In: *Informatik-Spektrum* 31 (2008), August, Nr. 5, S. S. 408–424. – ISSN 0170-6012

[Gesellschaft für Informatik e.V. (GI) (Hrsg) 2004] Gesellschaft für Informatik e.V. (GI) (Hrsg): *Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen.* Bonn, 2004

[GI 2008] GI: Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I; Empfehlungen der Gesellschaft für Informatik e.V. In: *LOG IN* 28 (2008)

[Glinz 2008] Glinz, M.: Modellierung in der Lehre an Hochschulen: Thesen und Erfahrungen. In: *Informatik-Spektrum* 31 (2008), August, Nr. 5, S. S. 425–434. – ISSN 0170-6012

[Goldin und Rudahl 2009] Goldin, S. ; Rudahl, K.: Software Process in the Classroom : A Comparative Study. In: *Computer Engineering* (2009), S. S. 427–431. ISBN 9781424445226

[Hampel et al. 1999] Hampel, T. ; Magenheim, J. ; Schulte, C.: Dekonstruktion von Informatiksystemen als Unterrichtsmethode-Zugang zu objektorientierten Sichtweisen im Informatikunterricht. In: *Informatik und Schule. Fachspezifische und fachübergreifende didaktische Konzepte.* Berlin, Heidelberg, New York u.a. : Schwill, A., 1999, S. S.149ff

[Hesse und Mayr 2008] Hesse, W. ; Mayr, H.: Modellierung in der Softwaretechnik. In: *Informatik-Spektrum* 31 (2008), August, Nr. 5, S. S. 375–375. – ISSN 0170-6012

[Horton 2007] Horton, F.: Understanding information literacy: A primer. Paris : UNESCO, 2007

[Hubwieser 2000] Hubwieser, P.: *Informatik am Gymnasium. Ein Gesamtkonzept für einen zeitgemäßen Informatikunterricht.* Habilitationsschrift, Fakultät für Informatik, Technische Universität München, 2000

[Hubwieser 2005] Hubwieser, P.: Von der Funktion zum Objekt Informatik für die Sekundarstufe. In: Friedrich, S. (Hrsg.): *INFOS 2005, 11. GI-Fachtagung Informatik und Schule, 28.-30. September 2005 an der TU Dresden.* 2005

[Hubwieser 2007] Hubwieser, P: *Didaktik der Informatik, 3. überarbeitete und erweiterte Auflage.* Springer examen.press, 2007

[Hubwieser und Broy 1996] Hubwieser, P. ; Broy, M.: Ein neuer Ansatz für den Informatikunterricht am Gymnasium. In: *LOG IN* 17 (1996), S. S. 42–47

[Ishii et al. 2010] Ishii, N. ; Suzuki, Y. ; Fujiyoshi, H. ; Fujii, T.: Fostering UML Modeling Skills and Social Skills through Programming Education. In: *2010 23rd IEEE Conference on Software Engineering Education and Training* (2010), März, S. S. 25–32. ISBN 978-1-4244-7052-5

[Joint Task Force on Computing 2001] Joint Task Force on Computing: Computing Curricula 2001 Computer Science. In: *Journal of Educational Resources in Computing (JERIC), 1 (3es)* 1 (2001), September. – ISSN 15314278

[Keil-Slawik 2002] Keil-Slawik, R.: Denkmedien-Mediendenken: Zum Verhältnis von Technik und Didaktik (Media For Thinking-Thinking About Media: On the Relati-

onship of Technology and Didactics). In: *it-Information Technology and Didactics (vormals it+ ti)* 44 (2002), S. S. 181–186

[Klafki 2007] Klafki, W.: Neue Studien zur Bildungstheorie und Didaktik: Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik. (2007)

[Kleuker 2011] Kleuker, S.: *Grundkurs Software-Engineering mit UML.* Wiesbaden : Vieweg+Teubner, 2011

[Klieme 2004] Klieme, E.: Was sind Kompetenzen und wie lassen sie sich messen. In: *Pädagogik* 56 (2004), S. S. 10–13

[Klieme et al. 2007] Klieme, E. ; Avenarius, H. ; Blum, W. ; Dörbrich, P. ; Gruber, H. ; Prenzel, M. ; Reiss, K. ; Riquarts, K. ; Rost, J. ; Tenorth, H. ; Vollmer, H.: *Zur Entwicklung nationaler Bildungsstandards. Bildungsreform Band 1. Expertise.* Bonn, Berlin : Bundesministerium fürr Bildung und Forschung (BMBF) Referat Publikationen; Internetredaktion., 2007

[KMK (Hrsg) 2004] KMK (Hrsg): *Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik.* KMK. Bonn : KMK Ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland, 2004

[Kohl 2009] Kohl, L.: *Kompetenzorientierter Informatikunterricht in der Sekundarstufe I unter Verwendung der visuellen Programmiersprache Puck.* Friedrich-Schiller-Universität Jena, Dissertation, 2009

[Kollee et al. 2009] Kollee, C. ; Magenheim, J. ; Nelles, W. ; Rhode, T. ; Schaper, N. ; Schubert, S. ; Stechert, P.: Computer science education and key competencies. In: *World Conference on Computers in Education* (2009)

[Kölling und Quig 2005] Kölling, M. ; Quig, B.: The BlueJ system and its pedagogy. In: *Computer Science ....* Computer Science Education, Special Issue of ACM Computing Surveys, Vol. 37, 2005

[Lankes 2006] Lankes, E.: Bildungsstandards in Deutschland. In: *Kompetenzorientierter Deutschunterricht.* Kronshangen : Institut für Qualitätsentwicklung an Schulen Schleswig Holstein (Hrsg.), 2006

[Lehner et al. 2010] Lehner, L. ; Magenheim, J. ; Nelles, W. ; Rhode, T. ; Schubert, S. ; Stechert, P. ; Schaper, N.: Informatics Systems and Modelling - Case Studies of Expert Interviews. In: *Key Competencies in the Knowledge Society.* Boston : Springer, 2010

[Leutner 2006] Leutner, D.: Kompetenzmodelle zur Erfassung individueller Lernergebnisse und zur Bilanzierung von Bildungsprozessen - Beschreibung eines neu eingerichteten Schwerpunktprogramms der DFG. In: *Zeitschrift für Pädagogik* 56 (2006), Nr. 6, S. S. 876–903

[Magenheim 2000] Magenheim, J.: Informatiksystem und Dekonstruktion als didaktische Kategorien - Theoretische Aspekte und unterrichtspraktische Implikationen einer systemorientierten Didaktik der Informatik. In: *Tagungsbeitrag zur GI-Tagung Informatik - Ausbildung und Beruf* (2000)

[Magenheim 2003a] Magenheim, J.: Informatik Lernlabor - Systemorientierte Didaktik in der Praxis. In: *Informatische Fachkonzepte im Unterricht, Proceedings der infos2003, 10.GI-Fachtagung Informatik und Schule, 17.-19. September in Garching bei München* (2003)

[Magenheim 2003b] Magenheim, J.: *Wissensmanagement, Dekonstruktion und E-Learning Communities in der Softwaretechnik-Didaktische Konzepte im BMBF-Projekt MuSofT*. Wa, 2003. – S. 255–269 S

[Magenheim 2005] Magenheim, J.: Towards a competence model for educational standards of informatics. In: *WCCE 2005 - Proceedings of the 8th IFIP World Conference on Computers in Education, University of Stellenbosch, Cape Town (SA)* (2005)

[Magenheim et al. 2010a] Magenheim, J. ; Nelles, W. ; Rhode, T. ; Schaper, N.: Towards a methodical approach for an empirically proofed competency model. In: Hromkovič, Juraj (Hrsg.) ; Královič, Richard (Hrsg.) ; Vahrenhold, Jan (Hrsg.): *Teaching Fundamentals Concepts of Informatics* Bd. 5941. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, S. S. 124–135

[Magenheim et al. 2010b] Magenheim, J. ; Nelles, W. ; Rhode, T. ; Schaper, N. ; Schubert, S. ; Stechert, P.: Competencies for informatics systems and modeling: Results of qualitative content analysis of expert interviews. In: *IEEE EDUCON 2010 Conference*, IEEE, April 2010, S. S. 513–521. – ISBN 978-1-4244-6568-2

[Magenheim und Scheel 2004] Magenheim, J. ; Scheel, O.: Using Learning Objects in an ICT-based Learning Environment. In: *Proceedings of E-Learn* (2004)

[Magenheim und Schulte 2005] Magenheim, J. ; Schulte, C.: Erwartungen und Wahlverhalten von Schülerinnen und Schülern gegenüber dem Schulfach Informatik

Ergebnisse einer Umfrage. In: *Friedrich, S. (Hrsg.) Unterrichtskonzepte für informatische Bildung, infos2005-11. GI - Fachtagung Informatik und Schule 28.-30.September 2005 in Dresden, Proceedings* (2005), S. S. 111–122

[Martin 2003] Martin, R.: *Agile software development: principles, patterns, and practices.* Prentice-Hall, Inc, 2003

[Mayring 2010] Mayring, P.: *Qualitative Inhaltsanalyse, Grundlagen und Techniken.* Beltz, 2010

[NCTM - National Council of Teachers of Mathematics 2000] NCTM - National Council of Teachers of Mathematics: Principles and Standards for School Mathematics. (2000)

[Nelles et al. 2009] Nelles, W. ; Rhode, T. ; Stechert, P.: Entwicklung eines Kompetenzrahmenmodells - Informatisches Modellieren und Systemverständnis. In: *Informatik-Spektrum* 33 (2009), Juni, Nr. 1, S. S. 45–53. – ISSN 0170-6012

[Nievergelt 1995] Nievergelt, J.: Welchen Wert haben theoretische Grundlagen für die Berufspraxis? Gedanken zum Fundament des Informatikturms. In: *Informatik Spektrum* (1995)

[Nygaard 1986] Nygaard, K.: Program development as a social activity. In: *H. Kugler (Hrsg.), Information Processing 86.* Amsterdam, 1986

[OECD 2005] OECD: The definition and selection of key competencies: Executive summary. Paris : OECD. Directorate for Education, 2005

[OECD 2001] OECD, Pisa Konsortium D.: *Schülerleistungen im internationalen Vergleich. Eine neue Rahmenkonzeption für die Erfassung von Wissen und Fähigkeiten.* Berlin : OECD PISA Deutschland, 2001

[Penon und Spolwig 1998] Penon, J. ; Spolwig, S.: Schöne visuelle Welt? Objektorientierte Programmierung mit DELPHI und JAVA. In: *LOG IN* 18 (1998), Nr. 5

[Pichler 2009] Pichler, R.: *Scrum-Agiles Projektmanagement erfolgreich einsetzen.* Heidelberg : dPunkt-Verlag, 2009

[Prenzel und Deutschland 2004] Prenzel, M. ; Deutschland, Pisa K.: *PISA 2003: Der Bildungsstand der Jugendlichen in Deutschland: Ergebnisse des zweiten internationalen Vergleichs.* Waxmann, 2004

[Rational Software Corporation IBM. 1998] Rational Software Corporation IBM.: *Rational unified process. Best practices for software development teams, white paper.* 1998

[Rectors, Universities 1999] Rectors, Universities, European: The Bologna Declaration on the European space for higher education : an explanation. (1999)

[Riecke-Baulecke und Artelt 2004] Riecke-Baulecke, C. ; Artelt, T.: *Bildungsstandards.* Oldenbourg Schulbuchverlag, 2004

[Roggio 2006] Roggio, R.: A Model for the Software Engineering Capstone Sequence using the Rational Unified Process. In: *Work* (2006), S. S. 306–311. ISBN 1595933158

[Ropohl 1999] Ropohl, G.: Philosophy of socio-technical systems. In: *Society for Philosophy and Technology* (1999)

[Royce 1970] Royce, W.: Managing the development of large software systems. In: *proceedings of IEEE WESCON* (1970)

[Schaper und Hochholdinger 2006] Schaper, N. ; Hochholdinger, S.: Psychologische Konzepte zur Modellierung und Messung von Kompetenzen in der Lehrerausbildung. In: *In: Hilligus A, Rinkens H-D (Hrsg) Standards und Kompetenzen - Neue Qualität in der Lehrerausbildung?* Münster : LIT-Verlag, 2006

[Schaper und Horvath 2008] Schaper, N. ; Horvath, E.: Development and Evaluation of a Model of eTeaching Competence. In: *In: Hambach S, Martens A, Urban B (eds) e-Learning Baltics 2008. Proceedings of the 1st International eLBa Science Conference.* Rostock : Fraunhofer IRB Verlag, 2008

[Schecker und Parchmann 2006] Schecker, H. ; Parchmann, I.: Modellierung naturwissenschaftlicher Kompetenz. In: *Zeitschrift für Didaktik der Naturwissenschaften* 12 (2006), Nr. 2006, S. S. 45–66

[Schubert und Schwill 2004] Schubert, S. ; Schwill, A.: *Didaktik der Informatik. 1. Auflage.* Heidelberg, Berlin : Spektrum Akademischer Verlag, 2004

[Schubert und Stechert 2007] Schubert, S. ; Stechert, P.: A strategy to structure the learning process towards understanding of informatics systems. In: *Benzie, David (Hrsg.) ; Iding, Marie (Hrsg.): Informatics, Mathematics and ICT: A golden triangle (IMICT2007). Northeastern University. Boston. MA* (2007)

[Schubert und Stechert 2010] Schubert, S. ; Stechert, P.: Competence Model Research on Informatics System Application. In: *Proceedings of the IFIP Conference New developments in ICT and Education, Amiens, Frankreich, June 28-30* (2010)

[Sonnentag 2006] Sonnentag, S.: Expertise in software design. In: *Cambridge handbook of expertise and expert performance / K. Anders Ericsson, Neil Charness, Paul J. Feltovich, & Robert R. Hoffmann (Eds.).* Cambridge : Cambridge University Press, 2006, S. S. 373–387

[Spiro und Feltovich 1992] Spiro, R. ; Feltovich, P.: Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In: *Constructivism and the Technology of Instruction.* Hillsale, NJ : Erlbaum, 1992

[Stachowiak 1973] Stachowiak, H.: Allgemeine Modelltheorie. Wien : Springer, 1973

[Standards 2001] Standards, IEEE Learning T.: IEEE LOM working draft 6.1. (2001)

[Stechert 2009] Stechert, Peer: *Fachdidaktische Diskussion von Informatiksystemen und der Kompetenzentwicklung im Informatikunterricht.* Universität Siegen, Didaktik der Informatik und E-Learning, Dissertation, 2009

[Thomas 2002] Thomas, M.: *Informatische Modellbildung. Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht.* Universität Potsdam, Dissertation, 2002

[Tulodziecki und Herzig 2002] Tulodziecki, G. ; Herzig, B.: Computer und Internet im Unterricht: Medienpädagogische Grundlagen und Beispiele. Cornelsen Scriptor, 2002

[Utting et al. 2010] Utting, I. ; Cooper, S. ; Kölling, M. ; Maloney, J. ; Resnick, M.: Alice, greenfoot, and scratch–a discussion. In: *ACM Transactions on Computing Education (TOCE)* (2010)

[Weinert 2002] Weinert, F.: *Leistungsmessungen in Schulen.* Beltz, 2002

[Whitehead 1939] Whitehead, A.: *Aims Of Education and Other Essays.* New York : The Free Press, 1939

Abbildungsverzeichnis

Tabellenverzeichnis

A. Anhang

A.1. Interviewszenarien

Hypothetische Szenarien (Critical Incidents)

Im Folgenden möchte ich Ihnen einige Szenarien vorstellen, die Aufgaben beinhalten, die Relevanz für den Informatikunterricht der Sekundarstufe II besitzen. Ich bitte Sie darum, sich sehr genau und detailliert die darin enthaltenen Problemstellungen vorzustellen und ebenso genau und detailreich zu schildern, wie Sie bei der Bewältigung der Aufgaben und Lösung der darin enthaltenen Probleme vorgehen würden.

Sz. 1

Sie erhalten den Softwareentwicklungsauftrag, ein Warenwirtschaftssystem für einen (Schul-)Kiosk zu entwickeln. Im Rahmen der Geschäftsmodellierung und Anforderungsanalyse sollen typische / alltägliche Geschäftsvorgänge erfasst werden.

- Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?
- Welche grafischen Beschreibungsmittel würden Sie dafür einsetzen?
- Welche Kenntnisse und Fähigkeiten benötigen Sie zur Modellierung der Geschäftsprozesse und zur Anforderungsanalyse?
    - Welche informatischen Sichten sind hierbei von Bedeutung?
    - Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
- Welche (motivationalen) Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zur Modellierung der Geschäftsprozesse und zur Anforderungsanalyse?
- Welche informatik-fremden Personen könnten (oder sollten) bei der Modellierung miteinbezogen werden? Welche Anforderungen kämen auf Sie zu, wenn Sie mit informatischen Laien über dieses SE-Projekt kommunizieren wollen?
- Wie würde ein Schüler die Aufgabe angehen?

Sie erhalten den Auftrag, die weiteren Phasen des Softwareengineerings-Prozesses zu planen.

- Welche weiteren Phasen müssen Ihrer Meinung nach bis zur Verteilung des Software-Produkts durchlaufen werden?
- Wie würden Sie hierbei vorgehen und was muss dabei beachtet werden?
- Welche Kenntnisse und Fähigkeiten benötigen Sie in diesen Phasen des SE-Prozesses (insbesondere welche informatische Sichten)?
- Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten sind in diesen SE-Phasen besonders relevant?
- Welche Phasen würden Sie im Rahmen eines Schulprojekts: „Schulkiosk“ im Informatikunterricht der Sekundarstufe durchlaufen wollen?
- In welcher Form würden Sie informatik-fremde Personen auch in die weiteren SE-Phasen mit einbeziehen? Was wäre dabei zu beachten?

In der Implementierungsphase des Projekts sollen Kleingruppen gebildet werden, um die verschiedenen Module der Software zeitgleich zu entwickeln.

- Was muss bei der Einteilung von SE-Gruppen im professionellen Umfeld berücksichtigt werden? Welche Anforderungen ergeben sich an die Gruppenmitglieder?
- Was muss bei der Gruppeneinteilung im Informatikunterricht beachtet werden? Welche sozialen und motivationalen Fähigkeiten und Einstellungen müssen seitens der Schüler vorhanden sein?
- Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren?

Sz. 2	**Sie erhalten den Auftrag ein Chat-System zu entwickeln. Im Rahmen der Designphase sollen Sie die potentiellen Programmmodule (Klassen) jeweils dem Client oder Server zuordnen.**
	
	o Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?
	o Welche grafischen Beschreibungsmittel würden Sie dafür einsetzen?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie zum Design des Client-Server-Systems?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zum Design des Client-Server-Systems?
	o Wie würde ein Schüler die Aufgabe angehen?
	
	Nach Abschluss der Analyse- und Designphase soll eine zeitlich parallele Implementierung von Client- und Server-Softwarekomponenten geschehen. Sie als Projektleiter stehen nun vor der Aufgabe, die Aufgaben sinnvoll auf Teilgruppen ihres Teams zu verteilen.
	
	o Wie würden Sie dabei vorgehen?
	o Was müsste in einem professionellen Umfeld bei der Gruppeneinteilung beachtet werden?
	o Wie würden Sie die Einteilung der Gruppen im schulischen Umfeld vornehmen um eine chancengleiche Kompetenzentwicklung zu ermöglichen?
	o Welche sozialen bzw. motivationalen Fähigkeiten der Schüler sollten zur erfolgreichen Implementierung vorhanden sein?
	o Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren?
	o Durch welche kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv?
	o Welche arbeitsbezogenen sozialen Umstände könnten den Erfolg gefährden?

Sz. 3	**Im Rahmen eines Softwareprojekts soll ein web-basierte Spiel implementiert werden. Sie haben bereits mit Hilfe von CRC-Karten Verantwortlichkeiten von Klassen herausgestellt und mögliche Zusammenhänge von Klassen lokalisiert. In einem weiteren Schritt soll nun ein umfassendes Klassendiagramm entwickelt werden.**
	
	o Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie zur Modellierung des Klassendiagramms?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zur Modellierung einer solchen, web-basierten Anwendung?
	o Beschreiben Sie die Unterschiede in der methodischen Vorgehensweise, die sich bei Anfängern, Fortgeschrittenen und Experten zeigen würden.
	o Wie würde ein Schüler die Aufgabe angehen?
	
	In einem späteren Schritt (kurz vor Abschluss des Projekts) soll die Software im Rahmen der Testphase bzgl. Ihrer Robustheit überprüft werden. Hierbei soll sichergestellt werden, dass keinerlei unerwartete Benutzereingaben das Programm zum Absturz bringen.
	
	o Wie würden Sie bei einem derartigen Test vorgehen, und was müssen Sie dabei beachten?
	o Wie würde ein Schüler die Aufgabe angehen?

Sz. 4	**Sie erhalten im Rahmen der Entwicklung einer einfachen Kontoführungs-Software den Auftrag, ein Klassendiagramm zu entwickeln. Die Software soll zunächst einfache Ein- und Auszahlvorgänge auf <u>einem</u> Bankkonto realisieren.**
	
	o Wie gehen sie dabei vor, und was müssen Sie dabei beachten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareentwicklung?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareentwicklung?
	o Welche möglichen Probleme könnten bei Schülern der Sekundarstufe auftreten?
	o Welche grafischen Beschreibungsmittel würden Sie einsetzen und warum?
	
	Für eine weitere Ausbaustufe der oben genannten Software soll nun ebenfalls ein Klassendiagramm erstellt werden. Im Gegensatz zu der ersten Ausbaustufe lassen sich nun beliebig viele Konten eröffnen. Neben Ein- und Auszahlungen auf das jeweilige Konto lassen sich nun auch Überweisungen zwischen den Konten vornehmen.
	
	o Wie würden Sie dabei vorgehen?
	o Welche zusätzlichen Anforderungen ergeben sich durch den Übergang zur erweiterten Ausbaustufe der Kontoführungs-Software?
	  o Rechtfertigen diese zusätzlichen Anforderungen eine Einteilung in Kleingruppen?
	  o Durch welche kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv?
	  o Welche arbeitsbezogenen sozialen Umstände könnten den Erfolg gefährden?
	o Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren?
	o Welche Anforderungen für die Schüler ergeben sich bei dieser komplexeren Version der Software?
	Wie würde ein Schüler die Aufgabe angehen?

Sz. 5	**Sie haben im Informatikunterricht der Sekundarstufe II Sortieralgorithmen thematisiert und hierbei ausgewählte Sortierverfahren innerhalb von Programmmodulen implementiert. Zum Abschluss der Unterrichtsreihe soll nun ein Visualisierungsmodul implementiert werden. Dieses soll das zu sortierende Feld (Array) visualisieren und die jeweiligen Teilschritte während der Sortierung darstellen, indem sämtliche Änderungen im Feld grafisch hervorgehoben werden.**
	
	o Wie würden Sie in diesem Zusammenhang vorgehen?
	o Was muss bei der Auswahl der Architektur, bei der Gestaltung der Schnittstellen und bei der Entwicklung der Benutzungsschnittstelle beachtet werden.
	o Welche Phasen ergeben sich bei der Entwicklung des Visualisierungsmoduls?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareimplementierung?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareimplementierung?
	o Wie würde Schüler an eine derartige Aufgabe herangehen?
	Welche Phasen sehen sie im Rahmen der schulischen Projektarbeit?

Sz.6	**Sie erhalten eine Software zur Verwaltung von persönlichen Gegen-ständen, die sie verliehen bzw. entliehen haben. Diese soll später auf modernen Mobiltelefonen eingesetzt werden.**
	Die Abteilung, die für die eigentliche Programmierung des Werkzeugs zuständig ist, bittet Sie, das Produkt in Bezug auf seine Ergonomie zu testen.
	
	o Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie um die Ergonomieeigenschaften der Verwaltungssoftware zu testen?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Bereitschaften und Einstellungen und welche sozial-kommunikativen Fähigkeiten benötigen Sie zum Testen der Ergonomieeigenschaften der Verwaltungssoftware?
	o Wie würde ein Schüler die Aufgabe angehen?

Sz. 7	**Sie haben gerade die neueste Version einer Standardsoftware installiert. Diese unterscheidet sich in der Funktionalität und Bedienung von der vorhergehenden Version. Um einen ersten Eindruck zu erhalten, möchten Sie die Software systematisch erkunden.**
	
	o Mit welchen Erwartungen gehen Sie an die neue Software heran?
	o Welche Erwartungshaltung begünstigt Ihre Arbeit? Welche Einstellung stünde der effektiven Absolvierung Ihres Arbeitsauftrags im Weg?
	o Inwieweit spielen bereits gesammelte Vorerfahrungen eine Rolle bei der systematisches Erkundung der Software?
	o Wie gehen Sie bei der Erkundung der neuen Software vor?
	o Empfehlen Sie diese oder eine andere Vorgehensweise auch für Schüler?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareerkundung?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareerkundung?
	o Wie würde ein Schüler die Aufgabe angehen?
	o Welche Probleme und Fallstricke können bei Schülern auftreten?
	o Wie können Sie Schüler unterstützen?

Sz. 8	**Eine kleine Firma handelt mit verschiedenen Werkzeugen. Um zu entscheiden, ob es sich lohnt das Geschäft an Samstagen länger zu öffnen, möchte der Geschäftsführer eine tägliche Umsatzübersicht über die wichtigsten Werkzeugkategorien. Das Geschäft arbeitet bisher ohne ein professionelles, rechnergestütztes Kassensystem und verfügt nicht über die finanziellen Mittel um ein solches zu installieren.**
	
	o Beschreiben Sie, wie Sie diese Aufgabe lösen können.
	o Welche Kenntnisse und Fähigkeiten benötigen Sie um diese Aufgabe lösen zu können?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Was genau müsste im Vorfeld mit dem Geschäftsführer besprochen werden, damit die Umsetzung des Auftrages erfolgreich verläuft?
	o Inwieweit spielt es eine Rolle, dass der Geschäftsführer ein Informatiklaie ist? Was müssten Sie in einem Gespräch mit ihm beachten?
	o Welche Einstellungen und Bereitschaften benötigen Sie um diese Aufgabe lösen zu können?
	o Welche kommunikativen Schwierigkeiten können auftreten und durch welches Vorgehen ließen sich diese überwinden?
	o Welche Probleme können bei einem Schüler auftreten?
	o Wie können informatische Konzepte einen Schüler beim lösen dieser Aufgabe unterstützen?

Sz. 9	**Sie bitten Ihre Kollegen, eine von Ihnen entwickelte Software auf Herz und Nieren zu testen. Diese soll in Computer-Fachgeschäften verwendet werden, um es Kunden zu ermöglichen den gewünschten Computer selbständig zu konfigurieren. Dazu kann aus verschiedenen Gehäuseformen, Prozessortypen und einer begrenzten Zahl von weiteren Komponenten und Peripheriegeräten ausgewählt werden. Zur Durchführung der Tests stehen Ihren Kollegen keine besonderen Werkzeuge zur Verfügung. Aus lizenzrechtlichen Gründen dürfen sie den Quellcode Ihrer Anwendung nicht weitergeben.**
	
	o Wie sollten die Kollegen dabei vorgehen, und was müssen sie dabei beachten?
	o Welches Handlungsmuster empfehlen Sie für den Softwaretest?
	o Welche Kenntnisse und Fähigkeiten benötigen die Kollegen für die Testung der entwickelten Software?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften sollten die Kollegen für diesen Testvorgang mitbringen?
	o Inwieweit wird es notwendig sein, sich neues, relevantes Wissen anzueignen? Wie kann dieses Wissen beschafft werden?
	o Wie können Sie die Tester bei Ihrer Aufgabe unterstützen?
	o Welche Erfolgs- oder Misserfolgserlebnisse können während der Projektdurchführung auftreten? Im Falle von Misserfolg: Welchen Anforderungen stehen Sie gegenüber, um sich neu zu motivieren?
	o Durch welche kommunikativen und kooperativen Voraussetzungen gelänge die Arbeit effektiv?
	o Wie würde ein Schüler die Aufgabe angehen?
	o Welche Probleme und Fallstricke können bei diesen Schülern auftreten?
	o Welche arbeitsbezogenen sozialen Umstände könnten den Erfolg gefährden?
	o Welche Erwartungshaltung bei Durchführung der Tests fördert den Erfolg des Projekts?

Sz. 10	**Stellen Sie sich vor, Sie haben in einer Fachzeitschrift einen Artikel zu einem neuen Forschungsgebiet gelesen und möchten nun mehr darüber erfahren. Um weitere Informationen zu bekommen, konsultieren Sie zunächst den Online-Katalog der nächstgelegenen Universitätsbibliothek.**
	
	o Was erwarten Sie von einem solchen Online-Katalog?
	o Wie versuchen Sie Informationen aus einem solchen Online-Katalog zu erhalten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Recherche?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Recherche?
	o Welche Probleme können bei einem Schüler auftreten?
	
	Leider hat die Recherche keine Ergebnisse gebracht und Sie haben sich zur Nutzung einer Internetsuchmaschine entschlossen.
	
	o Welche Erwartungen haben Sie an eine solche Suchmaschine?
	o Wie gehen Sie vor, um die gewünschten Informationen zu erhalten?
	o Welche Probleme können bei einem Schüler auftreten?
	
	Sie haben nun eine Anfrage an eine Suchmaschine gestellt, dabei aber keine Treffer erhalten.
	
	o Welche Gründe könnten dafür verantwortlich sein?
	o Wie gehen Sie vor, um die gewünschte Information zu erhalten?
	o Welche Probleme können bei einem Schüler auftreten?
	
	Bei einer weiteren Suchanfrage erhalten Sie über 3 Mio. Treffer. Viele davon sind für Sie jedoch nicht relevant.
	
	o Welche Gründe könnten dafür verantwortlich sein?
	o Wie gehen Sie vor, um die gewünschte Information zu erhalten?
	o Welche Probleme können bei einem unerfahrenen Anwender auftreten?
	
	Sie möchten, dass bei den Ergebnissen einer Suchanfrage mit dem Namen / Spezialgebiet Ihrer Schule / Institutes Ihr Internetauftritt weit vorne angezeigt wird. Ihr Ziel ist nicht unbedingt Platz 1, aber zumindest die erste Seite.
	
	o Wie gehen Sie vor, um dies zu realisieren?
	Welche Probleme können bei einem Schüler auftreten?

Sz. 11	**Sie haben gerade die neueste Version einer Datenbanksoftware installiert. Diese unterscheidet sich in der Funktionalität und Bedienung von der vorhergehenden Version. Um einen ersten Eindruck zu erhalten, möchten Sie die Software systematisch erkunden.**
	
	o Wie würden Sie dabei vorgehen, und was müssen Sie dabei beachten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareerkundung?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareerkundung?
	o Welche Erwartungshaltung begünstigt Ihre Arbeit? Welche Einstellung stünde der effektiven Absolvierung Ihres Arbeitsauftrags im Weg?
	o Inwieweit spielen bereits gesammelte Vorerfahrungen eine Rolle bei der systematisches

	Erkundung der Software? **Im Informatikunterricht werden Datenbanken entweder vom Ansatz des Anwendens oder des Modellierens behandelt. Diese beiden Ansätze sind als „Verwenden von Strukturen“ bzw. „Erzeugen von Strukturen“ zueinander komplementär.** o Wie würden Sie vorgehen, um das Erlernen der bei relationalen Datenbanken verwendeten Strukturierungsmethoden sowie das Hinterfragen der Strukturen zu fördern? o Welches Handlungsmuster empfehlen Sie für Schüler, um die Struktur des Systems zu erforschen? o Was ist bei der Dateneingabe zu beachten? o Was bei der Datenspeicherung? o Wie geschieht die Datengewinnung (Abfrage)? **Die Verknüpfungen, die zum Extrahieren gewünschter Daten benötigt und beim Erforschen gefunden wurden, können direkt als Datenbankabfragen formuliert werden.** o Wie würde ein Schüler eine Datenbank „Schule“ mit den Entitäten Klassen, Schüler, Lehrer etc. nutzen? o Welche Probleme und Fallstricke können bei diesen Schülern auftreten? o Welche Vorgehensweisen werden diese einsetzen? **Die durch die Anfragen gewonnenen Informationen bilden die Grundlage für die Rückkoppelung des Modells mit der Realität, für das „Hinterfragen erzeugter Strukturen“.** o Was erwarten Sie von Schülern, die den logischen Entwurf einer Datenbank erkundet haben? o Welche Vorgehensweisen werden diese Einsetzen? **Die erschlossenen Strukturierungsregeln schließlich erlauben das eigenständige Erstellen weiterer Datenbankmodelle von Realweltausschnitten durch die Lernenden („gestaltender“ Anwendungskontext).** o Wie unterstützt das Strukturieren eines Realweltproblems und das Überführen in einen logischen Entwurf das Verständnis für die Datenbank? o Inwieweit hilft das Strukturieren, die Komplexität des Datenbanksystems zu bewältigen? o Wie können Schüler motiviert werden, die Strukturen der Datenbank zu hinterfragen? o Wie können Sie die Schüler bei Ihrer Aufgabe unterstützen?

Sz. 12	**Sie werden von einem Kollegen gebeten, dessen Neuentwicklung zu testen. Es handelt sich dabei um eine Software, die Autohäusern verwendet werden soll, um es Kunden zu ermöglichen das gewünschte Auto selbständig zu konfigurieren. Dazu kann aus verschiedenen Fahrzeugtypen, Sondermodellen und Zusatzausstattungen ausgewählt werden. Zusätzlich kann das Autohaus in bestimmten Fällen einen Rabatt von bis zu 10 Prozent auf das erstellte Fahrzeug erlassen.**
	
	o Welche Teststrategie würde sie wählen und warum? Und was müssen Sie bei einem solchen Vorgehen beachten?
	o Welche Kenntnisse und Fähigkeiten benötigen Sie für eine entsprechende Softwareerkundung?
	  o Welche informatischen Sichten sind hierbei von Bedeutung?
	  o Welche Komplexität bzw. Komplexitätsaspekte weist das Projekt auf?
	o Welche Einstellungen und Bereitschaften benötigen Sie für eine entsprechende Softwareerkundung?
	o Auf welche Weise würden sich Unterschiede im Kompetenzniveau zwischen Laien und Experten bei dieser Arbeit zeigen?
	o Wie können Sie als Tester bei Ihrer Aufgabe unterstützt werden?
	o Durch welche kommunikativen und kooperativen Voraussetzungen gelänge dieses Vorhaben effektiv?
	o Wie würde ein Schüler die Aufgabe angehen?
	Welche Probleme und Fallstricke können bei diesen Schülern auftreten?

A.2. Messinstrument und Bewertungsschema

A.2.1. Fragebogen

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgaben zur informatischen Modellierung und Systemgestaltung

Vielen Dank, dass Sie sich bereit erklärt haben, diese Aufgabensammlung zu bearbeiten!

Selbstverständlich findet diese Bearbeitung **vollkommen anonym** statt. Uns geht es darum, in absehbarer Zeit ein Messinstrumentarium zu entwickeln für die beiden Bereiche *Informatisches Modellieren* und *Systemverständnis.* Da wir uns damit noch in der Vorerprobungsphase befinden, sind wir darauf angewiesen, einen ersten Testdurchlauf zu starten, um festzustellen, ob die gewählten Aufgaben sich für unseren Zweck eignen. Durch die Bearbeitung dieser Aufgaben helfen Sie uns dabei sehr. Herzlichen Dank!

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 1

A)

Ordnen Sie die folgenden **UML-Diagrammtypen** den jeweiligen **Phasen des Wasserfallmodells** zu. Beachten Sie, dass einzelne Diagrammtypen auch mehreren Phasen zugeordnet werden können und dass Felder ggf. frei bleiben können. Ergänzen Sie die untere Tabelle, indem Sie die Nummer der jeweiligen Diagrammtypen in der rechten Spalte ergänzen.

UML-Diagrammtypen (alphabetisch sortiert):

- **(1)** Aktivitätendiagramm
- **(2)** CRC-Karten
- **(3)** Klassendiagramm
- **(4)** Objektdiagramm (Objekt-Karten)
- **(5)** Sequenzdiagramm
- **(5)** Use Case Diagramm
- **(6)** Zustandsdiagramm

Anforderungsanalyse	
Analyse	
Design	
Implementierung	
Test	

B)

In der Praxis laufen die Phasen des Softwareengineerings selten linear ab. Häufig werden die einzelnen Phasen in mehreren Iterationen durchlaufen. Beschreiben Sie mindestens zwei Beispiele (in 2-3 Sätzen), in denen es notwendig ist, eine bereits abgeschlossene Phase des Wasserfallmodells nochmals zu durchlaufen.

Aufgabe 2

Sie sind als Projektmanager beauftragt, ein verteiltes Chatsystem zu entwickeln. Hierbei muss folgendes beachtet werden:

- Die verschiedenen Module der Software sollen von verschiedenen Teams entwickelt werden
- Zur gemeinsamen Modellierung sollen UML-Diagramme verwendet werden
- Das zu entwickelnde Chatsystem soll plattformunabhängig (Linux/Windows/Mac) lauffähig sein

i) Sie haben sich für Java als Plattform bzw. Programmiersprache entschieden: Begründen Sie Ihre Entscheidung, indem sie erläutern inwiefern Java als Programmiersprache/Plattform den oben genannten Ansprüchen genügt.

ii) Nennen Sie Nachteile von Java:

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 3

1. Erläutern Sie die Begriff „Klasse“.

2. Erläutern Sie den Begriff „Objekt“.

3. Beschreiben Sie den Zusammenhang zwischen „Klasse“ und „Objekt“. Erläutern Sie den Zusammenhang unter Verwendung der möglichen Klassen-/Objektkandidaten „Lehrer“, „Herr Meier“, „Herr Müller“.

4. Erläutern Sie den Begriff „Vererbung“ im Zusammenhang mit objektorientierter Modellierung. Erklären Sie den Begriff anhand der möglichen Klassenkandidaten „Person“, „Lehrer“, Schüler“.

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 4

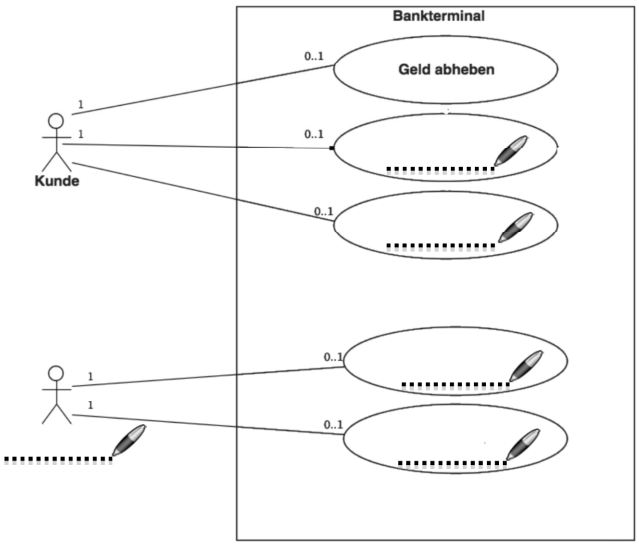
A)

Innerhalb dieser Aufgabe soll ein Bankterminal modelliert werden.
Ergänzen Sie das unten dargestellte **Use-Case-Diagramm**, indem Sie anhand der Szenariobeschreibung links die jeweiligen **Akteure** ergänzen und rechts die entsprechenden **Anwendungsfälle** eintragen.

Szenariobeschreibung:

Bankterminal:
Die Kunden sollen die Möglichkeit haben, Geld abzuheben und ihren Kontostand einzusehen. Des Weiteren sollen Sie eine Überweisung mit Hilfe des Terminals durchführen können. Um das System zu warten, müssen Servicetechniker in der Lage sein, Bargeld nachzufüllen und Softwareupdates einspielen zu können.

Use-Case-Diagramm:



B)

Im Rahmen eines Softwareentwicklungsauftrags für ein Kreditinstitut haben Sie mit dem Bankdirektor die Anforderungen an die zu entwickelnde Software erarbeitet. Sie müssen nun Ihren Entwicklerkollegen diese Information verständlich mitteilen.
In der Zwischenzeit haben jedoch Ihre Kollegen ohne Ihr Wissen eine aus Entwicklersicht sehr zeitsparende Vorgehensweise geplant, die jedoch aus Sicht des Kreditinstituts nicht geeignet ist. Sie haben nun die Aufgabe, Ihre Kollegen von der Notwendigkeit eines geeigneten Vorgehens im Sinne des Auftraggebers zu überzeugen.

i) Wie gehen Sie dabei vor? Was unternehmen Sie? **(Mehrfachnennungen möglich)**

- ☐ Ich vereinbare ein Treffen mit den Kollegen und stelle ihnen die Gesprächsergebnisse vor. Ich versuche sie von der Notwendigkeit der Umsetzung der Anforderungen zu überzeugen. Die Wünsche des Kunden zu berücksichtigen ist erforderlich.
- ☐ Ich begrüße die eigenen Vorschläge der Kollegen und veranlasse, dass nach diesen gearbeitet wird. Denn um erfolgreich zu sein, muss aus Entwicklersicht gearbeitet werden.

ii) Wie gehen Sie vor, um Ihren Wissensvorsprung durch das Gespräch mit dem Bankdirektor Ihren Entwicklerkollegen sinnvoll zu vermitteln. **(Mehrfachnennungen möglich)**

- ☐ Ich informiere die Kollegen über das geführte Gespräch mit dem Bankdirektor gar nicht, sondern lege fest, dass nach meinen Vorschlägen gearbeitet wird. Anstehende Diskussionen gefährden nur den Projekterfolg.
- ☐ Ich berufe ein Treffen mit den Kollegen ein und präsentiere ihnen die Gesprächsergebnisse mit dem Bankdirektor. Wenn nötig erstelle ein Handout zur besseren Verdeutlichung.

iii) Welche Probleme können sich dabei ergeben?

iv) Was müssten Sie bei der stattfindenden Diskussion beachten?
(Mehrfachnennungen möglich)

- ☐ Die verschiedenen Standpunkte sollen argumentativ vertreten werden.
- ☐ Ich höre den Ausführungen der anderen zu, berücksichtige diese allerdings für meine Entscheidung nicht.
- ☐ Wenn ich Diskussionsteilnehmer kritisiere, dann tue ich das konstruktiv um der Sache willen.
- ☐ Ich bin bereit, meinen Standpunkt von anderen kritisieren zu lassen, wenn sie Argumente dabei vorbringen.

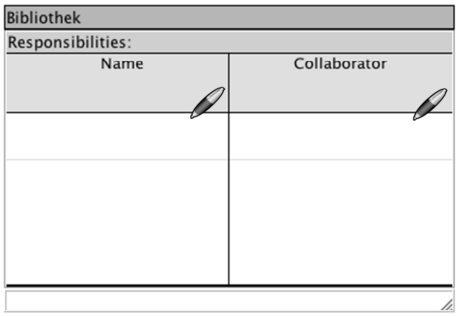
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

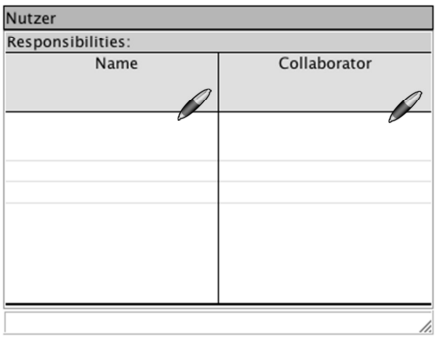
Aufgabe 5

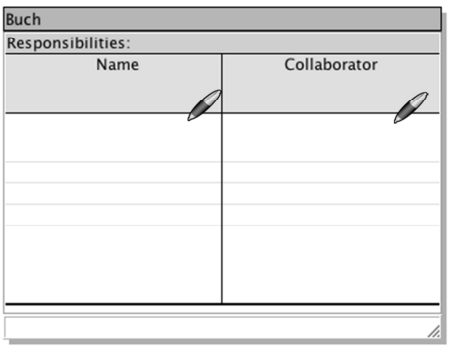
A)

Sie wurden beauftragt, eine Software zur Verwaltung Ihrer Schulbibliothek zu entwickeln. In der Analyse-Phase sollen zunächst CRC-Karten für die wichtigsten Klassen erstellt werden. Ergänzen Sie hierzu die unten dargestellten CRC-Karten um die jeweiligen *Responsibilities* und *Collaborators*. Orientieren Sie sich hierbei an der Beschreibung der Schulbibliothek I.

Schulbibliothek I
Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, id-Nummer und Anzahl der Seiten. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter.







UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

B)

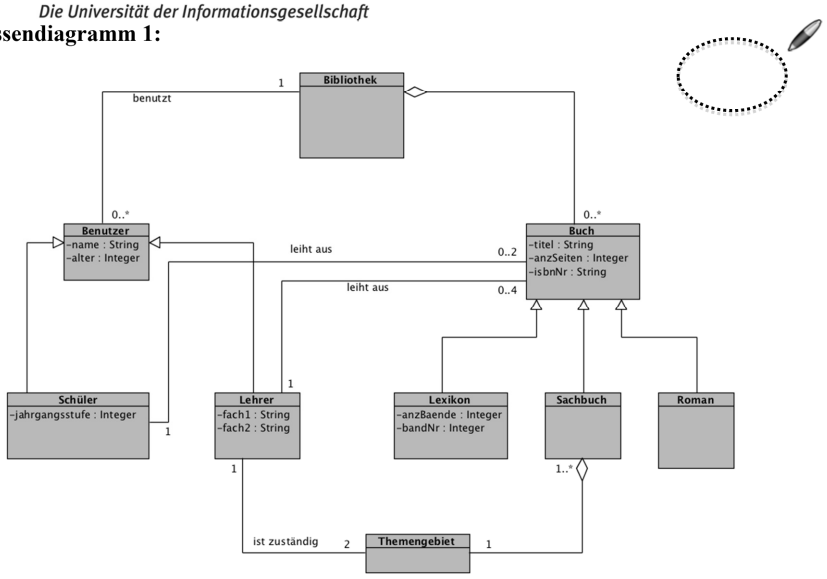
Wählen Sie das Klassendiagramm aus, dass die unten beschriebene erweiterte Version der Schulbibliothek (**Schulbibliothek II**) korrekt modelliert.

Klassendiagramm 1:☐
Klassendiagramm 2:☐ 

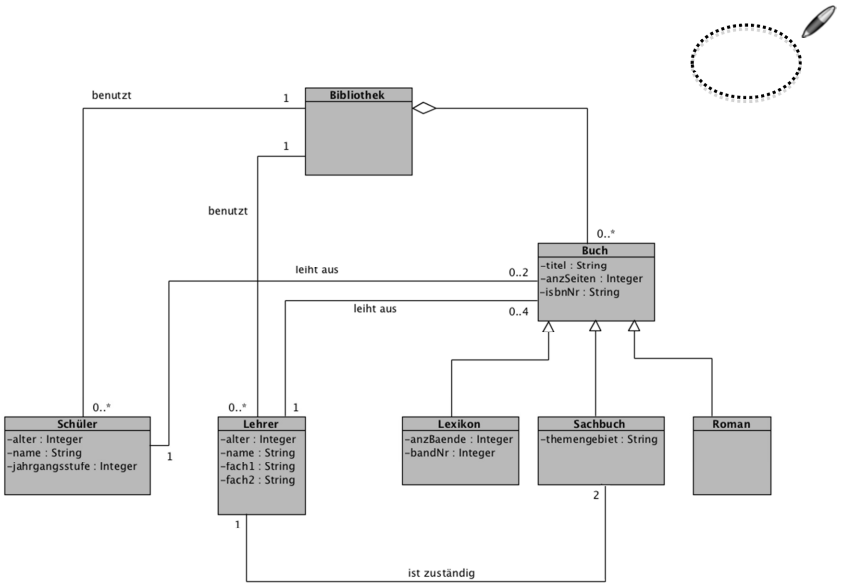
Im falschen Klassendiagramm befindet sich **(A) ein logischer Fehler** und eine **(B**) **eine Schwäche hinsichtlich doppelt gespeicherter Attribute**. Markieren Sie diese Fehler bzw. Schwächen im falschen Klassendiagramm, indem Sie die beteiligten Klassen und Assoziationen einkreisen und je nach Mangel/Schwäche mit **(A)** oder **(B)** beschriften.

Schulbibliothek II
Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, ISBN-Nummer und Anzahl der Seiten. Es gibt Sachbücher, Lexika und Romane. Sachbücher sind zusätzlich gekennzeichnet durch ein Themengebiet, Lexika durch die Anzahl Bände sowie die jeweilige Bandnummer des Exemplars. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter. Unterschieden wird zwischen verschiedenen Benutzergruppen: Lehrer haben ein erstes und zweites Unterrichtsfach und dürfen höchstens vier Bücher gleichzeitig ausleihen. Zusätzlich stehen Sie als Berater für zwei bestimmte Themengebiete der Fachbücher zur Verfügung. Schüler haben eine Jahrgangsstufe und dürfen höchstens zwei Bücher gleichzeitig ausleihen.

Klassendiagramm 1:



Klassendiagramm 2:



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 6

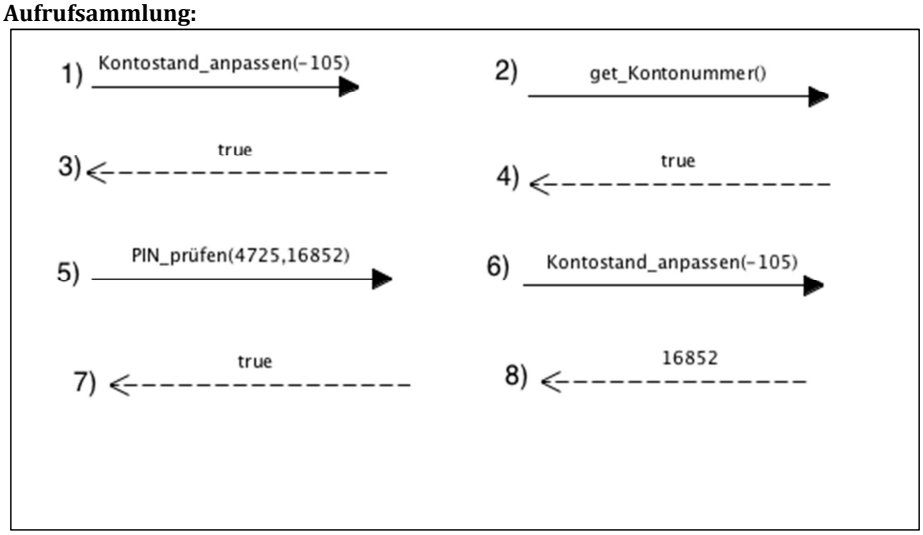
A)

Vervollständigen Sie anhand der Use-Case-Beschreibung „Geld abheben“ das entsprechende Sequenzdiagramm (siehe unten), indem Sie die einzelnen Aufrufe aus der unten dargestellten Aufrufsammlung auswählen und dem Sequenzdiagramm hinzufügen (*jeder Aufruf darf **einmal** verwendet werden; zeichnen Sie den jeweiligen **Aufrufpfeil** und ergänzen Sie die jeweilige **Nummer**; Die Aufruftext z.B. **„geld abheben(-105)** muss im Sequenzdiagramm **nicht** ergänzt werden*).

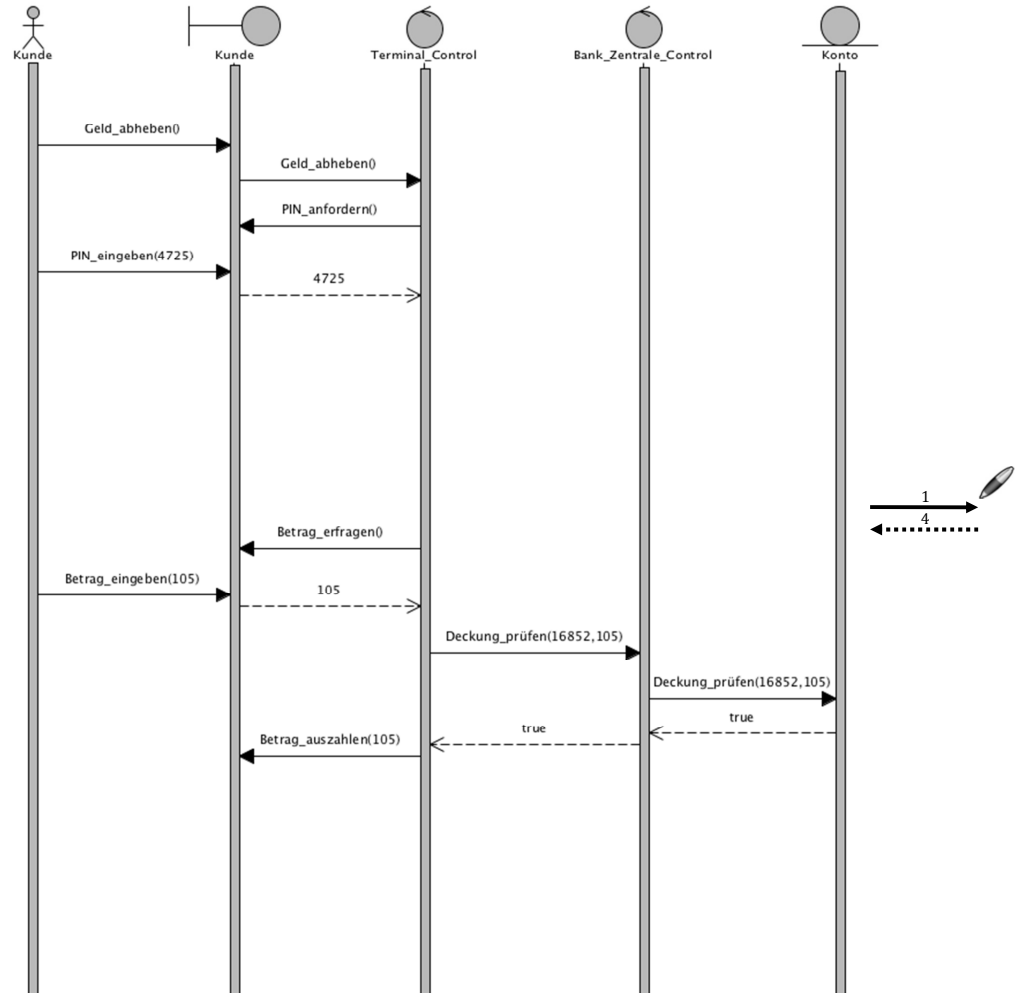
Szenario „Geld abheben“:

Schritt	Nutzer	Beschreibung der Aktivität
1	Kunde	„Geld abheben“ wählen
2	Bankterminal	PIN anfordern
3	Kunde	PIN eingeben: 4725
4	Bankzentrale	PIN prüfen
5	Bankterminal	Abzuhebenden Betrag erfragen
6	Kunde	Betrag eingeben: 105 Euro
7	Bankzentrale	Kontostand auf ausreichende Deckung prüfen
8	Bankterminal	Geld auszahlen
9	Kunde	Geld entnehmen
10	Bankzentrale	Kontostand anpassen

Aufrufsammlung:



Sequenzdiagramm zum Szenario „Geld abheben“:



B)

Stellen Sie sich vor, Sie würden im professionellen Umfeld Szenariobeschreibungen analysieren und möchten im nächsten Schritt ein Sequenzdiagramm erstellen. Welche Personen kämen als Gesprächspartner in Frage, die wichtige Informationen über das Geschäftsfeld liefern könnten?

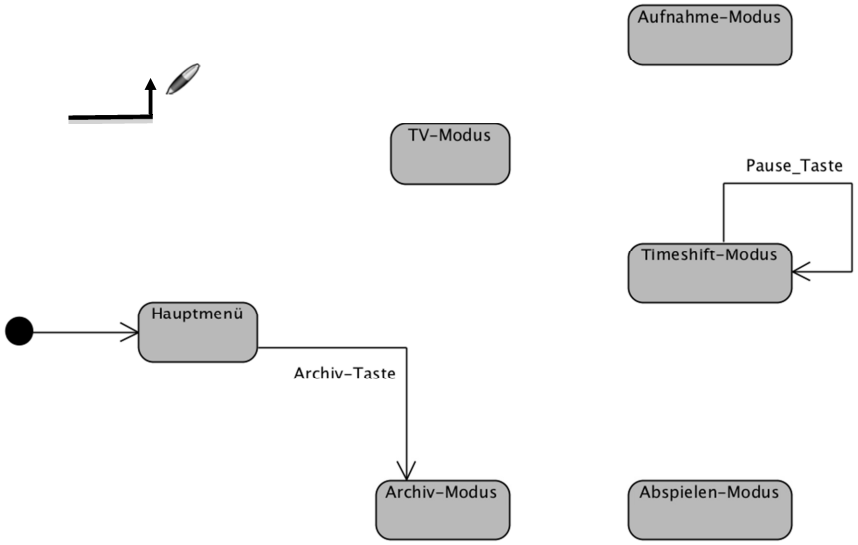
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 7

Ergänzen Sie ausgehend von der unten aufgeführten Funktionsbeschreibung eines Festplatten-Rekorders das Zustandsdiagramm: Ergänzen Sie hierbei die fehlenden Zustandsübergänge.

Festplatten-Rekorder
Das Gerät befindet sich nach dem Einschalten im *Hauptmenü*. Mittels der *TV-Taste* gelangt man in den *TV-Modus*, in dem das aktuelle Fernsehbild dargestellt wird. Betätigt man die *Record-Taste*, wechselt das Gerät in den *Aufnahme-Modus* und zeichnet das aktuelle Fernsehprogramm auf. Betätigt man in diesem Zustand die *Stop-Taste* wird die Aufnahme beendet und das Gerät wechselt wieder in den *TV-Modus*. Durch Betätigung der *Pause-Taste*innerhalb des *TV-Modus* wird der *Timeshift-Modus* aktiviert. Hierbei wird die aktuelle Fernsehsendung pausiert und ab diesem Zeitpunkt aufgenommen. Durch nochmaliges Drücken der *Pause-Taste* wird das Fernsehprogramm von der zuvor pausierten Position fortgesetzt. Drückt man die *Stop-Taste* wechselt der Festplatten-Rekorder wieder in den *TV-Modus* und spielt das TV-Programm ohne zeitlichen Versatz ab.
Drückt man innerhalb des Hauptmenüs die *Archiv-Taste,* wechselt das Gerät in den *Archiv-Modus*. Hier kann durch Betätigung der *Play-Taste* eine ausgewählte – zuvor aufgenommene – Sendung abgespielt werden (das Gerät wechselt in den *Abspielen-Modus*). Mit Hilfe der *Stop-Taste* gelangt man wiederum in den *Archiv-Modus*.
Sowohl im *TV*- als auch im *Archiv-Modus* gelangt man durch Drücken der *Menü-Taste* ins *Hauptmenü*.

Zustandsdiagramm des Festplatten-Rekorders:

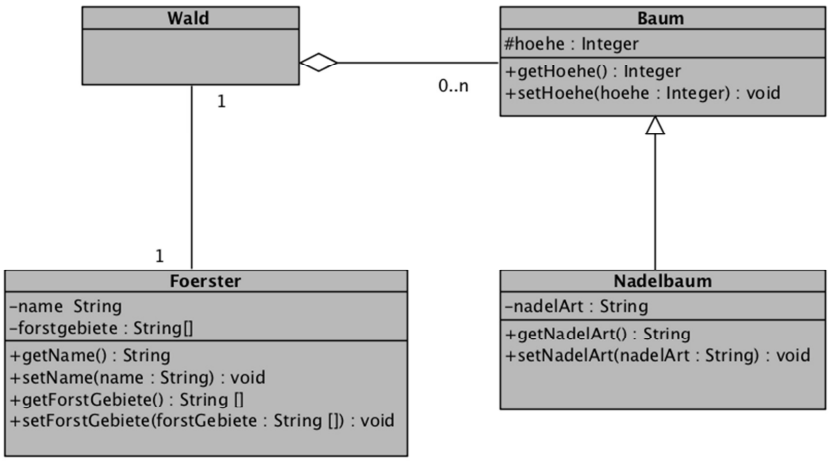


UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 8

Implementieren Sie die Klassen **`Wald`**, **`Baum`**, **`Foerster`** und **`Nadelbaum`** (Attribute, Methoden und Assoziationen/Aggregationen) anhand des unten dargestellten Klassendiagramms. Verwenden Sie die vorgegebenen Klassenrümpfe. Beachten Sie, dass die **Konstruktoren** der Klassen implementiert werden müssen, obwohl diese nicht im Klassendiagramm zu finden sind.

Klassendiagramm:



Quellcode:

Klasse `Wald`:

```
public class Wald                                    {
}
```

Klasse `Foerster`:

```
public class Foerster                                {
}
```

Klasse `Baum`:

```
public class Baum                                    {









}
```

Klasse `Nadelbaum`

```
public class NadelBaum                               {









}
```

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 9

A)

Gegeben sei die API der Klasse **`java.util.Vector`**. (siehe Anhang des Fragebogens) Verwenden sie diese, um die erforderlichen Methoden sowie deren Parameter und Rückgabetypen für den Umgang mit der Klasse **`Vector`** zu recherchieren.

Ergänzen Sie innerhalb des gegebenen Klassenrumpfes die **`main-Methode`** um Anweisungen (siehe Vector-API), sodass die folgende Funktionalität umgesetzt wird:

- Es soll ein Objekt der Klasse **`Vector`** erzeugt werden.
- Die folgenden **`Strings`** sollen sukzessive in den **`Vector`** eingefügt werden: „eins“, „zwei“, „drei“, „vier, „fünf“
- Innerhalb der im Klassenrumpf enthaltenen **`for-Schleife`** sollen sämtliche Elemente des **`Vectors`** auf der Konsole ausgegeben werden

Illustration des Vector-Objekts:

Index	0	1	2	3	4
Inhalt	„eins“	„zwei“	„drei“	„vier“	„fünf“

Klasse `Vectortest`

```
import java.util.Vector
public class VectorTest{

     public static void main(String[] args){
          //Vector-Objekt erzeugen




          //Strings zum Vector hinzufügen




          //alle Elemente des Vectors auf Konsole ausgeben
          for (                                          ){




          }
     }
}
```

B)

i) Stellen Sie sich vor, Sie arbeiten im Team an der Entwicklung einer MP3-Player-Software. Sie persönlich – als Experte auf diesem Gebiet - haben nun eine Klassenbibliothek zur Tonausgabe auf der Soundkarte entwickelt.
Wie gehen Sie vor, um Ihren Kollegen die Verwendung Ihres Programmmoduls zu ermöglichen? **(Mehrfachnennungen möglich)**

- ☐ Ich schicke ihnen den Quellcode meiner Klassenbibliothek zu und bitte sie, sich detailliert einzuarbeiten. Wenn Sie mein Programm vollständig verstehen können Sie es in ihr Projekt einbinden.
- ☐ Ich lasse ihnen eine Schnittstellenbeschreibung zukommen. Diese umfasst lediglich Methoden der Klassen und deren Signaturen. Das sollte für die Verwendung meines Programmmoduls vollkommen ausreichen.

ii) Ein weiterer Kollege hat zu einem späteren Zeitpunkt eine Alternative zu Ihrer Programmbibliothek zur Soundausgabe entwickelt. Diese erweist als deutlich besser als Ihre Programmbibliothek im Hinblick auf zukünftige Features des Mp3-Players. Wie verhalten Sie sich in dieser Situation, um den bestmöglichen Erfolg des Projekts zu erzielen? **(Mehrfachnennungen möglich)**

- ☐ Ich setze alle Energie in die Überarbeitung meiner Version, um es meinem Kollegen zu zeigen.

- ☐ Ich spreche mich mit meinem Kollegen ab, um aus unseren beiden Versionen das Beste herauszuholen und diese zu einer optimalen lauffähigen Version zu verbinden.
- ☐ Ich kündige, weil meine Arbeit nicht wertgeschätzt worden ist.
- ☐ Ich stelle meine eigene Lösung zurück und lasse zu, dass die bessere Lösung meines Kollegen genutzt wird, um den Projekterfolg nicht zu gefährden.

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 10

A)

Entscheiden Sie, ob die folgenden Aussagen **wahr** sind:

i) Im Rahmen der Testphase wird ausschließlich überprüft, ob der Auftraggeber mit dem für ihn entwickelten Softwaresystem zurechtkommt.	**ja❑ nein❑** ✕
ii) In der Testphase wird überprüft, ob sämtliche funktionalen Anforderungen aus der Anforderungsanalyse innerhalb des Softwaresystems umgesetzt wurden	**ja❑ nein❑**
iii) Es kann sinnvoll sein im Rahmen der Testphase einen Rückgriff auf die bereits abgeschlossene Anforderungsdefinition zu machen	**ja❑ nein❑**
iv) Wenn man eine Software innerhalb der Testphase auf Robustheit überprüft, testet man wie zuverlässig das System über einen längeren Zeitraum läuft.	**ja❑ nein❑**

v) Es gibt bestimmte sicherheitskritische Bereiche, in denen Softwaresysteme zur Unterstützung eingesetzt werden. Hierbei ist es von enormer Wichtigkeit, dass die jeweilige Software auf Herz und Nieren getestet wird.
Nennen Sie mindestens zwei solcher Bereiche, in denen ein sorgfältiger Softwaretest vor dem Einsatz der Software außerordentlich wichtig (vielleicht sogar lebenswichtig) ist.

B)

Entwickeln Sie anhand des unten dargestellten **Screenshots** einer Webapplikation zur Reisebuchung und anhand des Ausschnitts der **Anforderungsdefinition** einen geeigneten Testplan. Gehen Sie dabei folgendermaßen vor:

i) Überprüfen Sie, ob sämtliche funktionalen Anforderungen an die Software umgesetzt wurden, indem Sie für jede Anforderung **jeweils einen Testfall** entwickeln. Tragen Sie diese Testfälle in **Tabelle 1** ein:

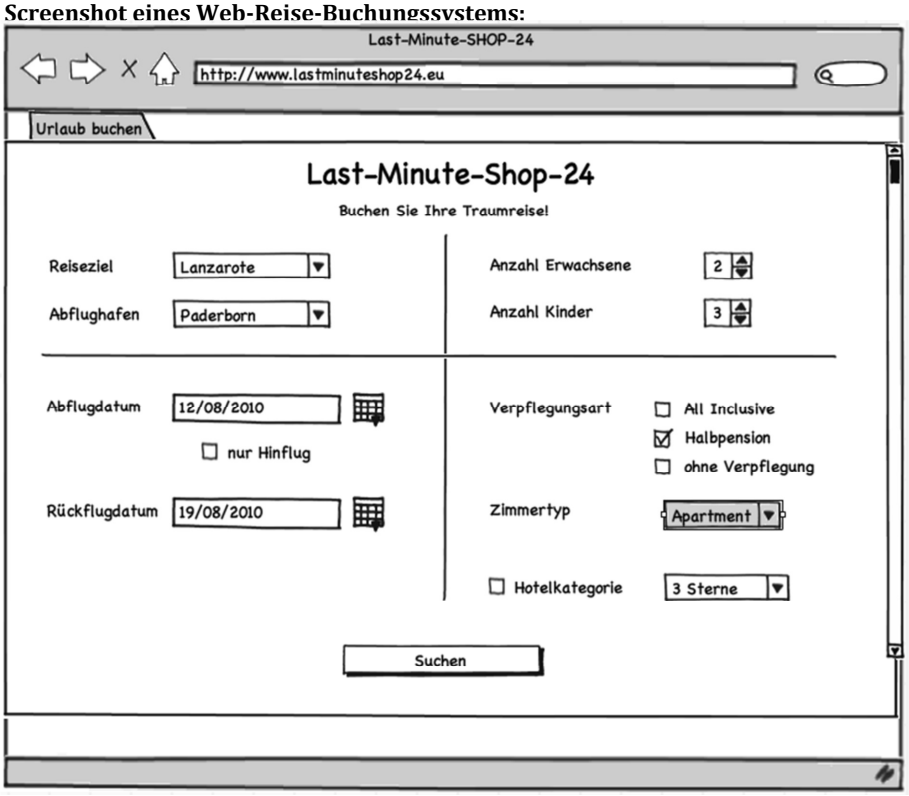
Anforderungsdefinition Reisebuchungssystem:

Anforderung 1: Benutzer kann Pauschalreisen suchen, indem er *Reiseziel*, *Abflughafen*, *Abflugdatum*, *Rückflugdatum* **(muss mindestens zwei Tage hinter dem Abflugdatum terminiert sein)**, *Anzahl Erwachsener* **(mindestens einer)**, *Anzahl Kinder*, *Verpflegungsarten* **(mindestens eine)** sowie *einen Zimmertyp* auswählt.

Anforderung 2: Der Benutzer kann ***optional die Hotelkategorie*** (Anzahl Sterne) mit in die Suche einbeziehen.

Anforderung 3: Benutzer kann auch **nur den Hinflug** buchen. Hierbei muss keine Eingabe in die Elemente der rechten Spalte gemacht werden.

Screenshot eines Web-Reise-Buchungssystems:



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Tabelle 1:

Testfall: Anforderung 1	**Testfall: Anforderung 2**	**Testfall: Anforderung 3**
Reiseziel: **`Lanzarote`**	Reiseziel:	Reiseziel:
Abflughafen: **`Paderborn`**	Abflughafen:	Abflughafen:
Abflugdatum: **`01.08.2010`**	Abflugdatum:	Abflugdatum:
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ❑
Rückflugdatum: **`08.08.2010`**	Rückflugdatum:	Rückflugdatum:
Anzahl Erwachsene: **`2`**	Anzahl Erwachsene:	Anzahl Erwachsene:
Anzahl Kinder: **`1`**	Anzahl Kinder:	Anzahl Kinder:
Verpflegungsart: AI ❑; VP ■; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑
Zimmertyp: **`Apartment`**	Zimmertyp:	Zimmertyp:
Hotelkategorie: ❑	Hotelkategorie: ❑	Hotelkategorie: ❑

ii) Überprüfen Sie die Robustheit der Anwendung, indem Sie nochmals den Auszug der **Anforderungsdefinition** betrachten und drei **unerwartete Testfälle** entwickeln, die die Anwendung zum Absturz bringen könnten. Ergänzen Sie diese Testfälle in **Tabelle 2**.

Tabelle 2:

Testfall: Fehleingabe 1	**Testfall: Fehleingabe 2**	**Testfall: Fehleingabe 3**
Reiseziel: **`Lanzarote`**	Reiseziel:	Reiseziel:
Abflughafen: **`Paderborn`**	Abflughafen:	Abflughafen:
Abflugdatum: **`01.08.2010`**	Abflugdatum:	Abflugdatum:
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ❑
Rückflugdatum: **`25.07.2010`**	Rückflugdatum:	Rückflugdatum:
Anzahl Erwachsene: **`2`**	Anzahl Erwachsene:	Anzahl Erwachsene:
Anzahl Kinder: **`1`**	Anzahl Kinder:	Anzahl Kinder:
Verpflegungsart: AI ■; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑
Zimmertyp: **`Apartment`**	Zimmertyp:	Zimmertyp:
Hotelkategorie: ■ **`(3 Sterne)`**	Hotelkategorie: ❑	Hotelkategorie: ❑

C)

i) Sie entwickeln eine Webseite für ein Reisebüro und befinden sich nach Abschluss der Implementierung in der Testphase. Um die Robustheit Ihres Softwareprodukts zu verbessern, sollen Betatester in den Prozess mit einbezogen werden. Welche Personen würden Sie in die Testphase mit einbeziehen? **(Mehrfachnennungen möglich)**

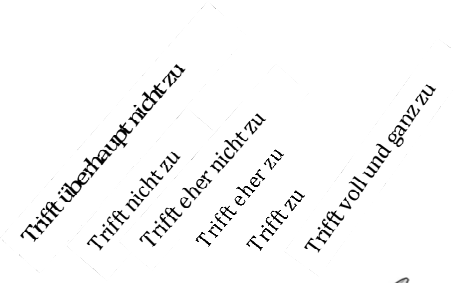
- ☐ Die Entwickler des Reisebuchungssystem
- ☒ Erfahrene Benutzer anderer Reisebuchungssysteme
- ☐ Benutzer, die Grundkenntnisse in der Benutzung des Internets haben
- ☐ Grundschüler, die gerade das Lesen gelernt haben

ii) Viele Betatester haben über Abstürze der Webseite berichtet. Wie gehen Sie vor, um die Eingaben in das System, die zum Absturz geführt haben, herauszufinden? Wie ermitteln Sie, was der Benutzer mit seinen Eingaben (die zum Absturz geführt haben)bezwecken wollte?

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Beurteilung verschiedener Aussagen zum informatischen Arbeiten

Im Folgenden werden verschiedene Aussagen präsentiert, die sich auf die Arbeit mit sogenannten informatischen Systemen beziehen. Unter informatischen Systemen versteht man einerseits das technische System an sich, andererseits aber auch den Umgang der Benutzer mit diesem technischen System.
Bitte geben Sie an, inwieweit die unten aufgeführten Aussagen Ihrer Meinung nach auf Sie zutreffen. Es handelt sich dabei um Ihre ganz persönliche Einschätzung. Es gibt keine richtigen und keine falschen Antworten. Bitte kreuzen Sie pro Frage genau eine Antwort an.



Informatische Systeme stecken voller Geheimnisse und ihr Funktionieren ist kaum nachvollziehbar.	○	○	○	○	○	○
Informatische Systeme kann man mit geeignetem Wissen gut nachvollziehen und verstehen. Man kann dann förmlich in sie hineinsehen.	○	○	○	○	○	○
Ich bin davon überzeugt, dass das Anwenden und Verstehen informatischer Systeme sehr wichtig ist.	○	○	○	○	○	○
Ich bin davon überzeugt, dass das Gestalten informatischer Systeme sehr bedeutsam und nützlich ist.	○	○	○	○	○	○
Wenn ich mit einem informatischen System arbeite, mache ich mir Gedanken um seine Tauglichkeit (Geeignetheit) für die zu bearbeitende Aufgabe.	○	○	○	○	○	○
Wenn ich mit einem informatischen System arbeite, bewerte ich dieses im Hinblick auf ihre Qualität.	○	○	○	○	○	○
Wenn ich mit einem informatischen System arbeite, bewerte ich dieses im Hinblick auf die Angemessenheit bezüglich der Aufgabenstellung.	○	○	○	○	○	○

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

	Trifft überhaupt nicht zu	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft zu	Trifft voll und ganz zu
Wenn ich mit informatischen Systemen arbeite, so muss ich mich den vorliegenden Gegebenheiten des informatischen Systems anpassen, da ich nichts daran ändern kann.	○	○	○	○	○	○
Wenn ich eine schwierige informatische Aufgabe löse, ist es mir wichtig, einen Plan dabei zu verfolgen. Unsystematisches Herumprobieren hilft da nicht weiter.	○	○	○	○	○	○
Wenn ich eine schwierige informatische Aufgabe löse, lege ich sofort los. Ich möchte zu Beginn nicht allzu viel Zeit mit der Planerei verlieren.	○	○	○	○	○	○
Informatische Systeme sind im Bereich der Informatik von großer Bedeutung, sind aber unbedeutend im alltäglichen Leben.	○	○	○	○	○	○
Informatische Systeme haben eine große Wirkung auf das alltägliche Leben. Kaum jemand kann sich ihrem Einfluss entziehen.	○	○	○	○	○	○
Informatische Aufgabenstellungen interessieren mich sehr.	○	○	○	○	○	○
Mir macht es Freude, informatische Modelle zu erstellen, mit denen man einen Ausschnitt realer Abläufe abbilden kann.	○	○	○	○	○	○
Ich beschäftige mich gerne auch in der Freizeit mit Informatik.	○	○	○	○	○	○

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

	Trifft überhaupt nicht zu	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft zu	Trifft voll und ganz zu
Manchmal wäre ich ganz froh, wenn die informatischen Aufgaben mehr mithilfe von festen Lösungsrezepten gelöst werden könnten.	○	○	○	○	○	○
Ich finde es wichtig, dass ich meine Arbeitsweise an die jeweilige informatische Aufgabenstellung anpasse, insbesondere wenn sie nur für mich ist.	○	○	○	○	○	○
Ich bin bereit, ganz neue Lösungswege, die ich bisher nicht kenne, zu entwickeln und einzusetzen, um eine informatische Aufgabe erfolgreich zu lösen.	○	○	○	○	○	○
Ich bin motiviert, meine informatischen Fähigkeiten stets zu verbessern (erhöhen) und mein Wissen diesbezüglich zu erweitern.	○	○	○	○	○	○
Ich will die mir anvertrauten informatischen Aufgaben erfolgreich bearbeiten.	○	○	○	○	○	○
Der Erfolg bei der Bearbeitung von informatischen Aufgaben ist mir wichtig.	○	○	○	○	○	○
Wenn ich mit einer informatischen Aufgabe betraut bin, werde ich alles daransetzen, sie auch erfolgreich zu bearbeiten.	○	○	○	○	○	○
Um Erfolg beim Bearbeiten von informatischen Aufgaben zu haben, bin ich bereit, mich anstrengen und bis an meine Leistungsgrenze zu gehen.	○	○	○	○	○	○

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

	Trifft überhaupt nicht zu	Trifft nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft zu	Trifft voll und ganz zu
An die Aufgaben, die man mir überträgt, fühle ich mich nicht so stark gebunden wie an die Aufgaben, für die ich mich ernsthaft interessiere.	○	○	○	○	○	○
Ich besitze ein ganz gutes informatisches Grundwissen, so dass ich mein Wissen nicht ständig erweitern muss.	○	○	○	○	○	○
Misserfolgserlebnisse bei informatischen Aufgaben sind nicht immer vermeidbar; dann heißt es jedoch weiterzumachen und durchzuhalten.	○	○	○	○	○	○
Mich faszinieren informatische Aufgaben besonders, wenn sie ein hohes Maß an Abstraktion verlangen.	○	○	○	○	○	○

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Abschlussbefragung

Bei dieser Aufgabensammlung zur informatischen Modellierung und Systemverständnis handelt es sich um eine Vorversion, die weiterhin noch verbessert werden muss. Durch die Beantwortung der folgenden Fragen tragen Sie aktiv zur Weiterentwicklung dieses Instrumentariums bei **(bitte beachten Sie die Aufgabenübersicht von im Anhang)**.
An welchen Stellen traten Verständnisschwierigkeiten auf? Z.B. durch schwammige Formulierungen oder durch die Aufgabenstellung selbst.

Konnten die Antwortformate sinnvoll genutzt werden? Waren die Instruktionen so formuliert, dass Sie darüber Bescheid wussten, was Sie zu tun haben? Wo genau (bei welchen Aufgaben) gab es Probleme? Wo fehlten hilfreiche Informationen?

Glauben Sie, dass die gewählten Formulierungen geeignet sind für Schülerinnen und Schüler der 12. Jahrgangsstufe? Welche Formulierungen in welchen Aufgaben sind Ihrer Meinung nach ungeeignet?

Welche Aufgaben fanden Sie besonders schwierig? Was genau war daran besonders schwierig?

Welche Aufgaben fanden Sie besonders einfach, trivial und damit vielleicht sogar überflüssig?

Für welche Aufgaben benötigten Sie besonders viel Zeit? Nennen Sie bitte diese Aufgaben?

Welche Aufgaben ließen sich sehr schnell bearbeiten? Nennen Sie bitte auch diese Aufgaben

Können Sie uns mitteilen, was Ihnen bei der Bearbeitung der Aufgaben besonders aufgefallen ist? Diese Auffälligkeiten können sich auf jeden erdenklichen Aspekt beziehen, der mit dieser Aufgabensammlung im Zusammenhang steht?

Aufgabenübersicht des Hauptfragebogens

1. Phasen des Wasserfallmodells / Zyklisches Vorgehen im Software Engineering Prozess
2. Auswahl der Architektur / Eigenschaften der Programmiersprache Java
3. Grundbegriffe der Objektorientierung
4. Entwicklung eines Use-Case-Diagramms / soziale Faktoren bei der Entwicklung einer Banksoftware
5. CRC-Karten / Klassendiagramm zur Schulbibliothekssoftware
6. Sequenzdiagramm zum Szenario „Geld abheben“
7. Zustandsdiagramm zum „Festplatten Rekorder“
8. Implementierung des Klassendiagramms „Wald, Förster, Baum, Nadelbaum“
9. Implementierung anhand einer API: Klasse „java.util.Vector“ / soziale Faktoren bei der Entwicklung von Programmmodulen „Mp3-Player“
10. Test des Reisebuchungssystems / soziale Faktoren beim Softwaretest

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Anhang zum Hauptfragebogen

API zu Aufgabe 9A)

Constructor Summary

Vector() Constructs an empty vector so that its internal data array has size 10 and its standard capacity increment is zero.
Vector(Collection<? extends E> c) Constructs a vector containing the elements of the specified collection, in the order they are returned by the collection's iterator.
Vector(int initialCapacity) Constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
Vector(int initialCapacity, int capacityIncrement) Constructs an empty vector with the specified initial capacity and capacity increment.

Method Summary

boolean	**add**(E e) Appends the specified element to the end of this Vector.
void	**add**(int index, E element) Inserts the specified element at the specified position in this Vector.
boolean	**addAll**(Collection<? extends E> c) Appends all of the elements in the specified Collection to the end of this Vector, in the order that they are returned by the specified Collection's Iterator.
boolean	**addAll**(int index, Collection<? extends E> c) Inserts all of the elements in the specified Collection into this Vector at the specified position.
void	**addElement**(E obj) Adds the specified component to the end of this vector, increasing its size by one.
int	**capacity**() Returns the current capacity of this vector.
void	**clear**() Removes all of the elements from this Vector.
Object	**clone**() Returns a clone of this vector.
boolean	**contains**(Object o) Returns true if this vector contains the specified element.
boolean	**containsAll**(Collection<?> c) Returns true if this Vector contains all of the elements in the specified Collection.
void	**copyInto**(Object[] anArray) Copies the components of this vector into the specified array.
E	**elementAt**(int index) Returns the component at the specified index.
Enumeration<E>	**elements**() Returns an enumeration of the components of this vector.
void	**ensureCapacity**(int minCapacity) Increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument.

boolean	**equals**(Object o) Compares the specified Object with this Vector for equality.
E	**firstElement**() Returns the first component (the item at index 0) of this vector.
E	**get**(int index) Returns the element at the specified position in this Vector.
int	**hashCode**() Returns the hash code value for this Vector.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this vector, or -1 if this vector does not contain the element.
int	**indexOf**(Object o, int index) Returns the index of the first occurrence of the specified element in this vector, searching forwards from index, or returns -1 if the element is not found.
void	**insertElementAt**(E obj, int index) Inserts the specified object as a component in this vector at the specified index.
boolean	**isEmpty**() Tests if this vector has no components.
E	**lastElement**() Returns the last component of the vector.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this vector, or -1 if this vector does not contain the element.
int	**lastIndexOf**(Object o, int index) Returns the index of the last occurrence of the specified element in this vector, searching backwards from index, or returns -1 if the element is not found.
E	**remove**(int index) Removes the element at the specified position in this Vector.
boolean	**remove**(Object o) Removes the first occurrence of the specified element in this Vector If the Vector does not contain the element, it is unchanged.
boolean	**removeAll**(Collection<?> c) Removes from this Vector all of its elements that are contained in the specified Collection.
void	**removeAllElements**() Removes all components from this vector and sets its size to zero.
boolean	**removeElement**(Object obj) Removes the first (lowest-indexed) occurrence of the argument from this vector.
void	**removeElementAt**(int index) Deletes the component at the specified index.
protected void	**removeRange**(int fromIndex, int toIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.
boolean	**retainAll**(Collection<?> c) Retains only the elements in this Vector that are contained in the specified Collection.
E	**set**(int index, E element) Replaces the element at the specified position in this Vector with the specified element.
void	**setElementAt**(E obj, int index)

	Sets the component at the specified `index` of this vector to be the specified object.
`void`	**`setSize`**`(int newSize)` Sets the size of this vector.
`int`	**`size`**`()` Returns the number of components in this vector.
`List<E>`	**`subList`**`(int fromIndex, int toIndex)` Returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive.
`Object[]`	**`toArray`**`()` Returns an array containing all of the elements in this Vector in the correct order.
`<T> T[]`	**`toArray`**`(T[] a)` Returns an array containing all of the elements in this Vector in the correct order; the runtime type of the returned array is that of the specified array.
`String`	**`toString`**`()` Returns a string representation of this Vector, containing the String representation of each element.
`void`	**`trimToSize`**`()` Trims the capacity of this vector to be the vector's current size.

A.2.2. Bewertungsschema

Musterlösung

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgaben zur informatischen Modellierung und Systemgestaltung

Vielen Dank, dass Sie sich bereit erklärt haben, diese Aufgabensammlung zu bearbeiten!

Selbstverständlich findet diese Bearbeitung **vollkommen anonym** statt. Uns geht es darum, in absehbarer Zeit ein Messinstrumentarium zu entwickeln für die beiden Bereiche *Informatisches Modellieren* und *Systemverständnis.* Da wir uns damit noch in der Vorerprobungsphase befinden, sind wir darauf angewiesen, einen ersten Testdurchlauf zu starten, um festzustellen, ob die gewählten Aufgaben sich für unseren Zweck eignen. Durch die Bearbeitung dieser Aufgaben helfen Sie uns dabei sehr. Herzlichen Dank!

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 1

A)

Ordnen Sie die folgenden **UML-Diagrammtypen** den jeweiligen **Phasen des Wasserfallmodells** zu. Beachten Sie, dass einzelne Diagrammtypen auch mehreren Phasen zugeordnet werden können und dass Felder ggf. frei bleiben können. Ergänzen Sie die untere Tabelle, indem Sie die Nummer der jeweiligen Diagrammtypen in der rechten Spalte ergänzen.

UML-Diagrammtypen (alphabetisch sortiert):

- **(1)** Aktivitätendiagramm
- **(2)** CRC-Karten
- **(3)** Klassendiagramm
- **(4)** Objektdiagramm (Objekt-Karten)
- **(5)** Sequenzdiagramm
- **(5)** Use Case Diagramm
- **(6)** Zustandsdiagramm

Anforderungsanalyse	Use-Case-Diagramm Aktivitätendiagramm
Analyse	Klassendiagramm Sequenzdiagramm
Design	Klassendiagramm Deploymentdiagramm Zustandsdiagramm
Implementierung	
Test	

B)

In der Praxis laufen die Phasen des Softwareengineerings selten linear ab. Häufig werden die einzelnen Phasen in mehreren Iterationen durchlaufen. Beschreiben Sie mindestens zwei Beispiele (in 2-3 Sätzen), in denen es notwendig ist, eine bereits abgeschlossene Phase des Wasserfallmodells nochmals zu durchlaufen.

1. **Beispiel:** In der Testphase muss u.a. geprüft werden, ob alle funktionalen Anforderungen korrekt umgesetzt wurden. Daher ist es sinnvoll einen Rückgriff auf die Anforderungsanalyse zu machen.
2. **Beispiel:** Innerhalb der Implementierungsphase kann sich herausstellen, dass Änderungen an der Architektur der Software vorgenommen werden müssen. Dementsprechend kann es sinnvoll/erforderlich sein, von der Implementierungsphase einen Rückgriff auf die Designphase zu machen.

Aufgabe 2

Sie sind als Projektmanager beauftragt, ein verteiltes Chatsystem zu entwickeln. Hierbei muss folgendes beachtet werden:

- Die verschiedenen Module der Software sollen von verschiedenen Teams entwickelt werden
- Zur gemeinsamen Modellierung sollen UML-Diagramme verwendet werden
- Das zu entwickelnde Chatsystem soll plattformunabhängig (Linux/Windows/Mac) lauffähig sein

i) Sie haben sich für Java als Plattform bzw. Programmiersprache entschieden: Begründen Sie Ihre Entscheidung, indem sie erläutern inwiefern Java als Programmiersprache/Plattform den oben genannten Ansprüchen genügt.

- objektorientierte Programmiersprache eignet sich gut zur Modularisierung
- Beispielsweise direkte Übersetzung von Klassendiagrammen in Java-Quellcode möglich
- Java ist durch die Java Virtual Machine plattformunabhängig

ii) Nennen Sie Nachteile von Java:

- Performance-Nachteile durch die JVM (leider nicht mehr aktuell aufgrund JIT-Compiler ;-( )
- Keine maschinennahe Programmierung möglich
- Kein direkter Speicherzugriff aufgrund fehlender Pointer-Arithmetik

Aufgabe 3

1. Erläutern Sie die Begriff „Klasse“.

Unter einer **Klasse** versteht man in der objektorientierten Programmierung ein abstraktes Modell bzw. einen *Bauplan* für eine Reihe von ähnlichen Objekten.
Die Klasse dient als Bauplan für die Abbildung von realen Objekten in Softwareobjekte und beschreibt Attribute (Eigenschaften) und Methoden (Verhaltensweisen) der Objekte

2. Erläutern Sie den Begriff „Objekt“.

Ein **Objekt** bezeichnet in der objektorientierten Programmierung (OOP) ein Exemplar eines bestimmten Datentyps oder einer bestimmten Klasse (auch „Objekttyp“ genannt). In diesem Zusammenhang werden Objekte auch als „Instanzen einer Klasse“ bezeichnet. Objekte sind also konkrete Ausprägungen („Instanzen“) eines Objekttyps.

3. Beschreiben Sie den Zusammenhang zwischen „Klasse“ und „Objekt“. Erläutern Sie den Zusammenhang unter Verwendung der möglichen Klassen-/Objektkandidaten „Lehrer“, „Herr Meier“, „Herr Müller“.

Die Klasse „Lehrer“ kann als abstraktes Modell für die Objekte „Herr Meier“ und „Herr Müller“ gesehen werden. Diese stellen dann eine Instanz der Klasse „Lehrer“ dar.

4. Erläutern Sie den Begriff „Vererbung“ im Zusammenhang mit objektorientierter Modellierung. Erklären Sie den Begriff anhand der möglichen Klassenkandidaten „Person“, „Lehrer“, Schüler“.

Klassen können miteinander in *hierarchischen Beziehungen* stehen und zu komplexen Strukturen werden. Die Gesetzmäßigkeiten, nach denen diese gebildet werden, beschreibt das grundlegende Konzept der Vererbung. Hier sind weiterhin die Begriffe *Basisklasse* und *abgeleitete Klasse* von Bedeutung, um die Verhältnisse der Klassen untereinander zu charakterisieren. Dabei beschreibt die Basisklasse allgemeine Eigenschaften, ist also eine *Verallgemeinerung* der abgeleiteten Klassen; diese sind somit *Spezialisierungen* der Basisklasse.

Beispiel: Basisklasse `Person` ist Verallgemeinerung der abgeleiteten Klassen (Spezialisierungen) `Lehrer und Schüler`.

Dabei *erben* die abgeleiteten Klassen alle Eigenschaften und Methoden der Basisklasse (d.h. ein Motorrad hat alle Eigenschaften eines Kraftfahrzeugs, und man kann alles mit ihm machen, das man mit einem Kraftfahrzeug machen kann). Zusätzlich führt die abgeleitete Klasse *zusätzliche* Eigenschaften und Methoden ein, die bei ihren Objekten möglich sind. (Das Motorrad hat z.B. einen Gepäckträger, ein Auto nicht, dafür aber einen Kofferraum.)

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 4

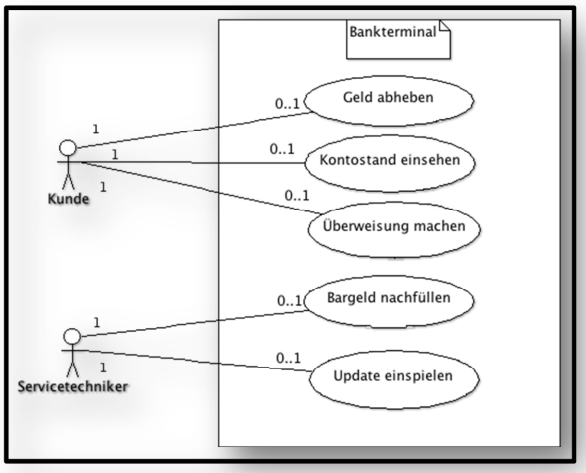
A)

Innerhalb dieser Aufgabe soll ein Bankterminal modelliert werden.
Ergänzen Sie das unten dargestellte **Use-Case-Diagramm**, indem Sie anhand der Szenariobeschreibung links die jeweiligen **Akteure** ergänzen und rechts die entsprechenden **Anwendungsfälle** eintragen.

Szenariobeschreibung:

Bankterminal:
Die Kunden sollen die Möglichkeit haben, Geld abzuheben und ihren Kontostand einzusehen. Des Weiteren sollen Sie eine Überweisung mit Hilfe des Terminals durchführen können. Um das System zu warten, müssen Servicetechniker in der Lage sein, Bargeld nachzufüllen und Softwareupdates einspielen zu können.

Use-Case-Diagramm:



B)

Im Rahmen eines Softwareentwicklungsauftrags für ein Kreditinstitut haben Sie mit dem Bankdirektor die Anforderungen an die zu entwickelnde Software erarbeitet. Sie müssen nun Ihren Entwicklerkollegen diese Information verständlich mitteilen.
In der Zwischenzeit haben jedoch Ihre Kollegen ohne Ihr Wissen eine aus Entwicklersicht sehr zeitsparende Vorgehensweise geplant, die jedoch aus Sicht des Kreditinstituts nicht geeignet ist. Sie haben nun die Aufgabe, Ihre Kollegen von der Notwendigkeit eines geeigneten Vorgehens im Sinne des Auftraggebers zu überzeugen.

i) Wie gehen Sie dabei vor? Was unternehmen Sie? **(Mehrfachnennungen möglich)**

√ Ich vereinbare ein Treffen mit den Kollegen und stelle ihnen die Gesprächsergebnisse vor. Ich versuche sie von der Notwendigkeit der Umsetzung der Anforderungen zu überzeugen. Die Wünsche des Kunden zu berücksichtigen ist erforderlich.

□ Ich begrüße die eigenen Vorschläge der Kollegen und veranlasse, dass nach diesen gearbeitet wird. Denn um erfolgreich zu sein, muss aus Entwicklersicht gearbeitet werden.

ii) Wie gehen Sie vor, um Ihren Wissensvorsprung durch das Gespräch mit dem Bankdirektor Ihren Entwicklerkollegen sinnvoll zu vermitteln. **(Mehrfachnennungen möglich)**

□ Ich informiere die Kollegen über das geführte Gespräch mit dem Bankdirektor gar nicht, sondern lege fest, dass nach meinen Vorschlägen gearbeitet wird. Anstehende Diskussionen gefährden nur den Projekterfolg.

√ Ich berufe ein Treffen mit den Kollegen ein und präsentiere ihnen die Gesprächsergebnisse mit dem Bankdirektor. Wenn nötig erstelle ein Handout zur besseren Verdeutlichung.

iii) Welche Probleme können sich dabei ergeben?

- unterschiedliches Verständnis Problembereichs
- Die Entwickler, die bereits in Vorleistung getreten empfinden ihre Arbeit als nicht wertgeschätzt
- …

iv) Was müssten Sie bei der stattfindenden Diskussion beachten?
(Mehrfachnennungen möglich)

√ Die verschiedenen Standpunkte sollen argumentativ vertreten werden.

□ Ich höre den Ausführungen der anderen zu, berücksichtige diese allerdings für meine Entscheidung nicht.

√ Wenn ich Diskussionsteilnehmer kritisiere, dann tue ich das konstruktiv um der Sache willen.

√ Ich bin bereit, meinen Standpunkt von anderen kritisieren zu lassen, wenn sie Argumente dabei vorbringen.

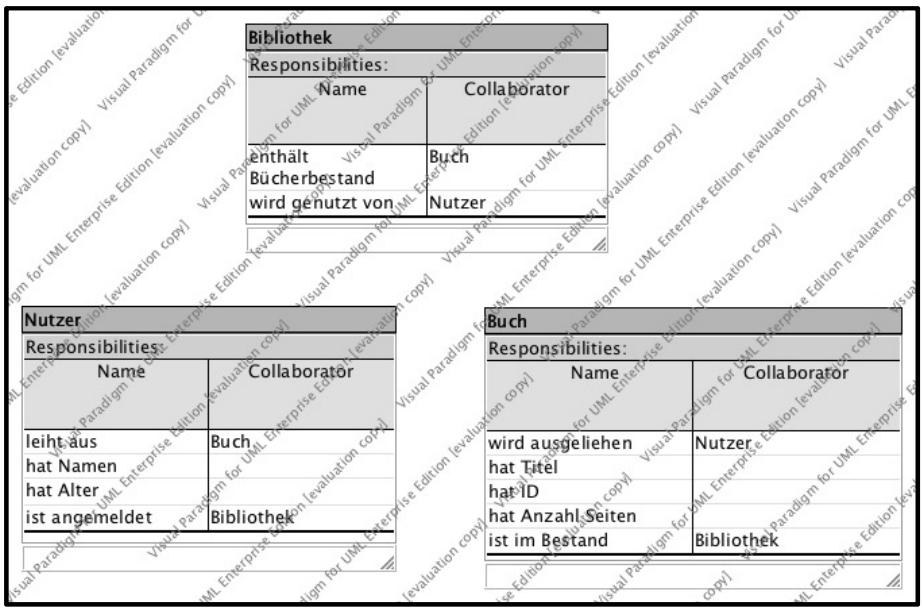
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 5

A)

Sie wurden beauftragt, eine Software zur Verwaltung Ihrer Schulbibliothek zu entwickeln. In der Analyse-Phase sollen zunächst CRC-Karten für die wichtigsten Klassen erstellt werden. Ergänzen Sie hierzu die unten dargestellten CRC-Karten um die jeweiligen *Responsibilities* und *Collaborators*. Orientieren Sie sich hierbei an der Beschreibung der Schulbibliothek I.

Schulbibliothek I
Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, id-Nummer und Anzahl der Seiten. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter.



B)

Wählen Sie das Klassendiagramm aus, dass die unten beschriebene erweiterte Version der Schulbibliothek (**Schulbibliothek II**) korrekt modelliert.

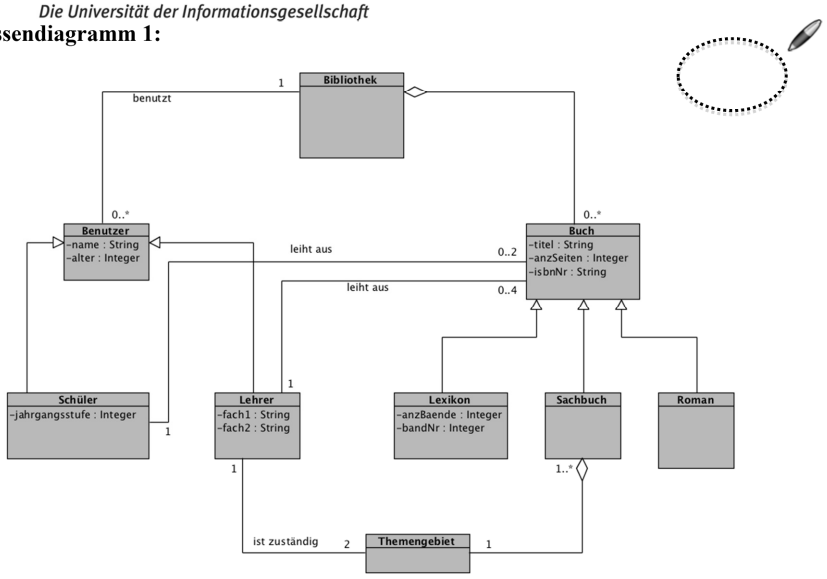
Klassendiagramm 1: ☐

Klassendiagramm 2: √ ✗

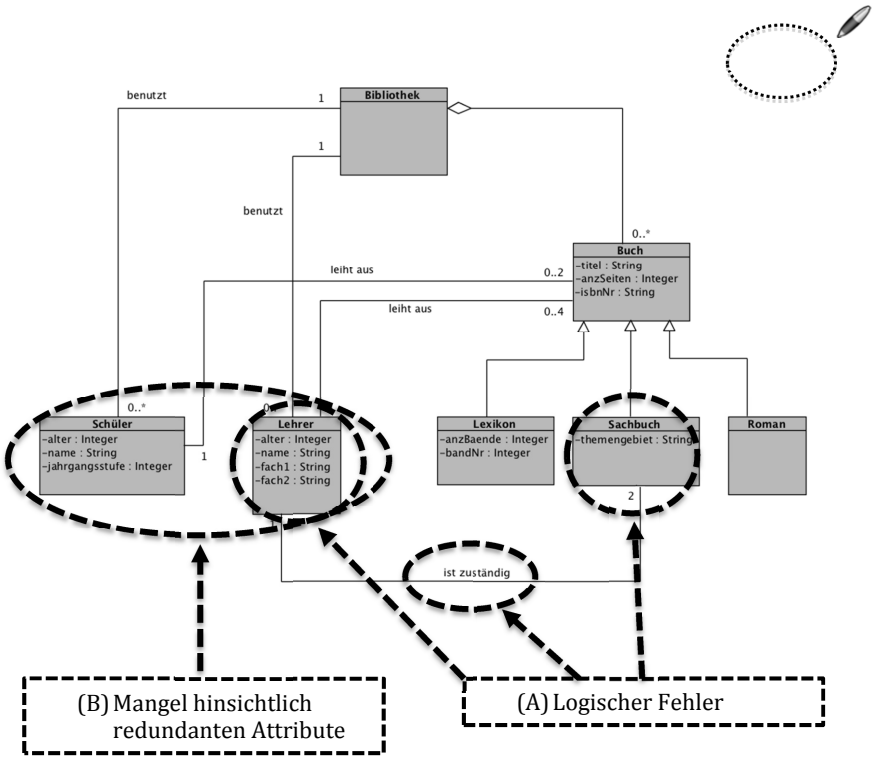
Im falschen Klassendiagramm befindet sich **(A) ein logischer Fehler** und eine **(B**) **eine Schwäche hinsichtlich doppelt gespeicherter Attribute**. Markieren Sie diese Fehler bzw. Schwächen im falschen Klassendiagramm, indem Sie die beteiligten Klassen und Assoziationen einkreisen und je nach Mangel/Schwäche mit **(A)** oder **(B)** beschriften.

Schulbibliothek II
Die Schulbibliothek umfasst einen Bestand von Büchern. Diese sind gekennzeichnet durch deren Titel, ISBN-Nummer und Anzahl der Seiten. Es gibt Sachbücher, Lexika und Romane. Sachbücher sind zusätzlich gekennzeichnet durch ein Themengebiet, Lexika durch die Anzahl Bände sowie die jeweilige Bandnummer des Exemplars. Die Schulbibliothek wird von verschiedenen Personen genutzt. Diese haben einen Namen und ein Alter. Unterschieden wird zwischen verschiedenen Benutzergruppen: Lehrer haben ein erstes und zweites Unterrichtsfach und dürfen höchstens vier Bücher gleichzeitig ausleihen. Zusätzlich stehen Sie als Berater für zwei bestimmte Themengebiete der Fachbücher zur Verfügung. Schüler haben eine Jahrgangsstufe und dürfen höchstens zwei Bücher gleichzeitig ausleihen.

Klassendiagramm 1:



Klassendiagramm 2:



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 6

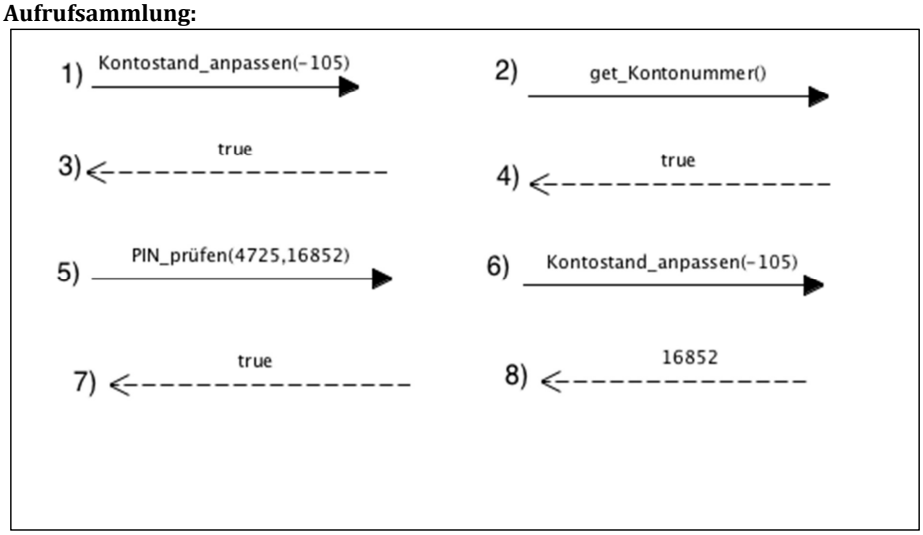
A)

Vervollständigen Sie anhand der Use-Case-Beschreibung „Geld abheben“ das entsprechende Sequenzdiagramm (siehe unten), indem Sie die einzelnen Aufrufe aus der unten dargestellten Aufrufsammlung auswählen und dem Sequenzdiagramm hinzufügen (*jeder Aufruf darf **einmal** verwendet werden; zeichnen Sie den jeweiligen **Aufrufpfeil** und ergänzen Sie die jeweilige **Nummer**; Die Aufruftext z.B. **„geld abheben(-105)** muss im Sequenzdiagramm **nicht** ergänzt werden*).

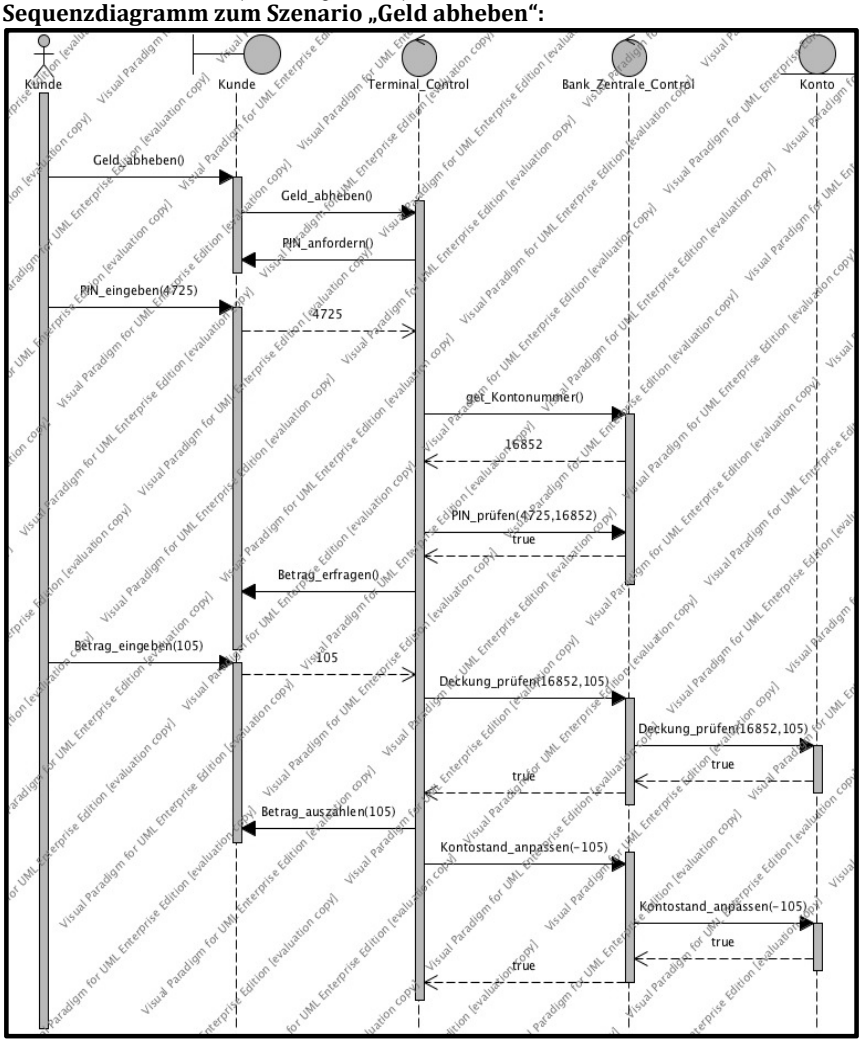
Szenario „Geld abheben“:

Schritt	Nutzer	Beschreibung der Aktivität
1	Kunde	„Geld abheben“ wählen
2	Bankterminal	PIN anfordern
3	Kunde	PIN eingeben: 4725
4	Bankzentrale	PIN prüfen
5	Bankterminal	Abzuhebenden Betrag erfragen
6	Kunde	Betrag eingeben: 105 Euro
7	Bankzentrale	Kontostand auf ausreichende Deckung prüfen
8	Bankterminal	Geld auszahlen
9	Kunde	Geld entnehmen
10	Bankzentrale	Kontostand anpassen

Aufrufsammlung:



Sequenzdiagramm zum Szenario „Geld abheben“:



B)

Stellen Sie sich vor, Sie würden im professionellen Umfeld Szenariobeschreibungen analysieren und möchten im nächsten Schritt ein Sequenzdiagramm erstellen. Welche Personen kämen als Gesprächspartner in Frage, die wichtige Informationen über das Geschäftsfeld liefern könnten?

- z.B. Domänenexperten, denen die Geschäftsprozesse bekannt sind

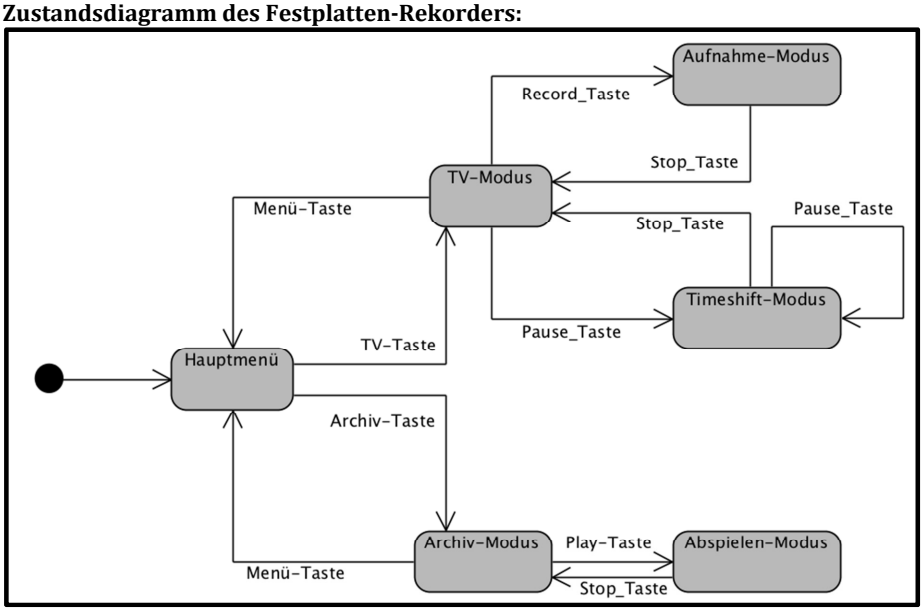
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 7

Ergänzen Sie ausgehend von der unten aufgeführten Funktionsbeschreibung eines Festplatten-Rekorders das Zustandsdiagramm: Ergänzen Sie hierbei die fehlenden Zustandsübergänge.

Festplatten-Rekorder
Das Gerät befindet sich nach dem Einschalten im *Hauptmenü*. Mittels der *TV-Taste* gelangt man in den *TV-Modus*, in dem das aktuelle Fernsehbild dargestellt wird. Betätigt man die *Record-Taste*, wechselt das Gerät in den *Aufnahme-Modus* und zeichnet das aktuelle Fernsehprogramm auf. Betätigt man in diesem Zustand die *Stop-Taste* wird die Aufnahme beendet und das Gerät wechselt wieder in den *TV-Modus*. Durch Betätigung der *Pause-Taste*innerhalb des *TV-Modus* wird der *Timeshift-Modus* aktiviert. Hierbei wird die aktuelle Fernsehsendung pausiert und ab diesem Zeitpunkt aufgenommen. Durch nochmaliges Drücken der *Pause-Taste* wird das Fernsehprogramm von der zuvor pausierten Position fortgesetzt. Drückt man die *Stop-Taste* wechselt der Festplatten-Rekorder wieder in den *TV-Modus* und spielt das TV-Programm ohne zeitlichen Versatz ab.
Drückt man innerhalb des Hauptmenüs die *Archiv-Taste,* wechselt das Gerät in den *Archiv-Modus*. Hier kann durch Betätigung der *Play-Taste* eine ausgewählte – zuvor aufgenommene – Sendung abgespielt werden (das Gerät wechselt in den *Abspielen-Modus*). Mit Hilfe der *Stop-Taste* gelangt man wiederum in den *Archiv-Modus*.
Sowohl im *TV*- als auch im *Archiv-Modus* gelangt man durch Drücken der *Menü-Taste* ins *Hauptmenü*.

Zustandsdiagramm des Festplatten-Rekorders:

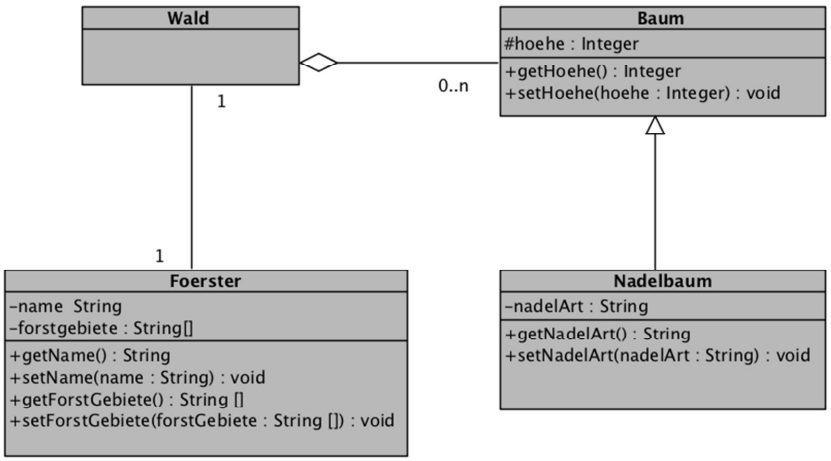


UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Aufgabe 8

Implementieren Sie die Klassen **`Wald`**, **`Baum`**, **`Foerster`** und **`Nadelbaum`** (Attribute, Methoden und Assoziationen/Aggregationen) anhand des unten dargestellten Klassendiagramms. Verwenden Sie die vorgegebenen Klassenrümpfe. Beachten Sie, dass die **Konstruktoren** der Klassen implementiert werden müssen, obwohl diese nicht im Klassendiagramm zu finden sind.

Klassendiagramm:



Quellcode:

Klasse `Wald`:

```
public class Wald{
    private Foerster foerster;
    private Baum[] baeume;
    //Konstruktor
    Wald(Foerster foerster){
        this.foerster=foerster;
        baeume = new Baum[];
    }
    //...
}
```

Klasse `Foerster`:

```
public class Foerster{
    private String name;
    //Konstruktor
    Foerster(String name){
        this.name=name;
    }
    //...
}
```

Klasse `Baum`:

```
public class Baum{
    private int hoehe;
    //Konstruktor
    Baum(int hoehe){
        this.hoehe=hoehe;
    }
    //...
}
```

Klasse `Nadelbaum`

```
public class NadelBaum extends Baum{
    private String nadelArt;
    //Konstruktor
    NadelBaum(int hoehe, String nadelArt){
        super(hoehe);
        this.nadelArt=nadelArt;
    }
    //...
}
```

Aufgabe 9

A)

Gegeben sei die API der Klasse **`java.util.Vector`**. (siehe Anhang des Fragebogens) Verwenden sie diese, um die erforderlichen Methoden sowie deren Parameter und Rückgabetypen für den Umgang mit der Klasse **`Vector`** zu recherchieren.

Ergänzen Sie innerhalb des gegebenen Klassenrumpfes die **`main-Methode`** um Anweisungen (siehe Vector-API), sodass die folgende Funktionalität umgesetzt wird:

- Es soll ein Objekt der Klasse **`Vector`** erzeugt werden.
- Die folgenden **`Strings`** sollen sukzessive in den **`Vector`** eingefügt werden: „eins“, „zwei“, „drei“, „vier, „fünf“
- Innerhalb der im Klassenrumpf enthaltenen **`for-Schleife`** sollen sämtliche Elemente des **`Vectors`** auf der Konsole ausgegeben werden

Illustration des Vector-Objekts:

Index	0	1	2	3	4
Inhalt	„eins“	„zwei“	„drei“	„vier“	„fünf“

Klasse `Vectortest`

```
import java.util.Vector
public class VectorTest{

      public static void main(String[] args){
            //Vector-Objekt erzeugen
            Vector v = new Vector();
            //Strings zum Vector hinzufügen
            v.add(„eins“);
            v.add(„zwei“);
            v.add(„drei“);
            v.add(„vier“);
            v.add(„fünf“);
            //alle Elemente des Vectors auf Konsole ausgeben
            for(int i = 0; i < v.size(); i++){
                  System.out.println(v.elementAt(i));
            }
      }
}
```

B)

i) Stellen Sie sich vor, Sie arbeiten im Team an der Entwicklung einer MP3-Player-Software. Sie persönlich – als Experte auf diesem Gebiet - haben nun eine Klassenbibliothek zur Tonausgabe auf der Soundkarte entwickelt.
Wie gehen Sie vor, um Ihren Kollegen die Verwendung Ihres Programmmoduls zu ermöglichen? **(Mehrfachnennungen möglich)**

□ Ich schicke ihnen den Quellcode meiner Klassenbibliothek zu und bitte sie, sich detailliert einzuarbeiten. Wenn Sie mein Programm vollständig verstehen können Sie es in ihr Projekt einbinden.

√ Ich lasse ihnen eine Schnittstellenbeschreibung zukommen. Diese umfasst lediglich Methoden der Klassen und deren Signaturen. Das sollte für die Verwendung meines Programmmoduls vollkommen ausreichen.

ii) Ein weiterer Kollege hat zu einem späteren Zeitpunkt eine Alternative zu Ihrer Programmbibliothek zur Soundausgabe entwickelt. Diese erweist als deutlich besser als Ihre Programmbibliothek im Hinblick auf zukünftige Features des Mp3-Players. Wie verhalten Sie sich in dieser Situation, um den bestmöglichen Erfolg des Projekts zu erzielen? **(Mehrfachnennungen möglich)**

□ Ich setze alle Energie in die Überarbeitung meiner Version, um es meinem Kollegen zu zeigen.

√ Ich spreche mich mit meinem Kollegen ab, um aus unseren beiden Versionen das Beste herauszuholen und diese zu einer optimalen lauffähigen Version zu verbinden.

□ Ich kündige, weil meine Arbeit nicht wertgeschätzt worden ist.

√ Ich stelle meine eigene Lösung zurück und lasse zu, dass die bessere Lösung meines Kollegen genutzt wird, um den Projekterfolg nicht zu gefährden.

Aufgabe 10

A)

Entscheiden Sie, ob die folgenden Aussagen **wahr** sind:

i) Im Rahmen der Testphase wird ausschließlich überprüft, ob der Auftraggeber mit dem für ihn entwickelten Softwaresystem zurechtkommt.	**ja❑ nein√** ✕
ii) In der Testphase wird überprüft, ob sämtliche funktionalen Anforderungen aus der Anforderungsanalyse innerhalb des Softwaresystems umgesetzt wurden	**ja√ nein❑**
iii) Es kann sinnvoll sein im Rahmen der Testphase einen Rückgriff auf die bereits abgeschlossene Anforderungsdefinition zu machen	**ja√ nein❑**
iv) Wenn man eine Software innerhalb der Testphase auf Robustheit überprüft, testet man wie zuverlässig das System über einen längeren Zeitraum läuft.	**ja❑ nein√**

v) Es gibt bestimmte sicherheitskritische Bereiche, in denen Softwaresysteme zur Unterstützung eingesetzt werden. Hierbei ist es von enormer Wichtigkeit, dass die jeweilige Software auf Herz und Nieren getestet wird.
Nennen Sie mindestens zwei solcher Bereiche, in denen ein sorgfältiger Softwaretest vor dem Einsatz der Software außerordentlich wichtig (vielleicht sogar lebenswichtig) ist.

- **Software von Flugzeugen, z.B. die Steuerung des Fahrwerks**
- **Software im medizinischen Bereich, z.B. Software zur Verabreichung von Medikationen über einen Tropf**

B)

Entwickeln Sie anhand des unten dargestellten **Screenshots** einer Webapplikation zur Reisebuchung und anhand des Ausschnitts der **Anforderungsdefinition** einen geeigneten Testplan. Gehen Sie dabei folgendermaßen vor:

i) Überprüfen Sie, ob sämtliche funktionalen Anforderungen an die Software umgesetzt wurden, indem Sie für jede Anforderung **jeweils einen Testfall** entwickeln. Tragen Sie diese Testfälle in **Tabelle 1** ein:

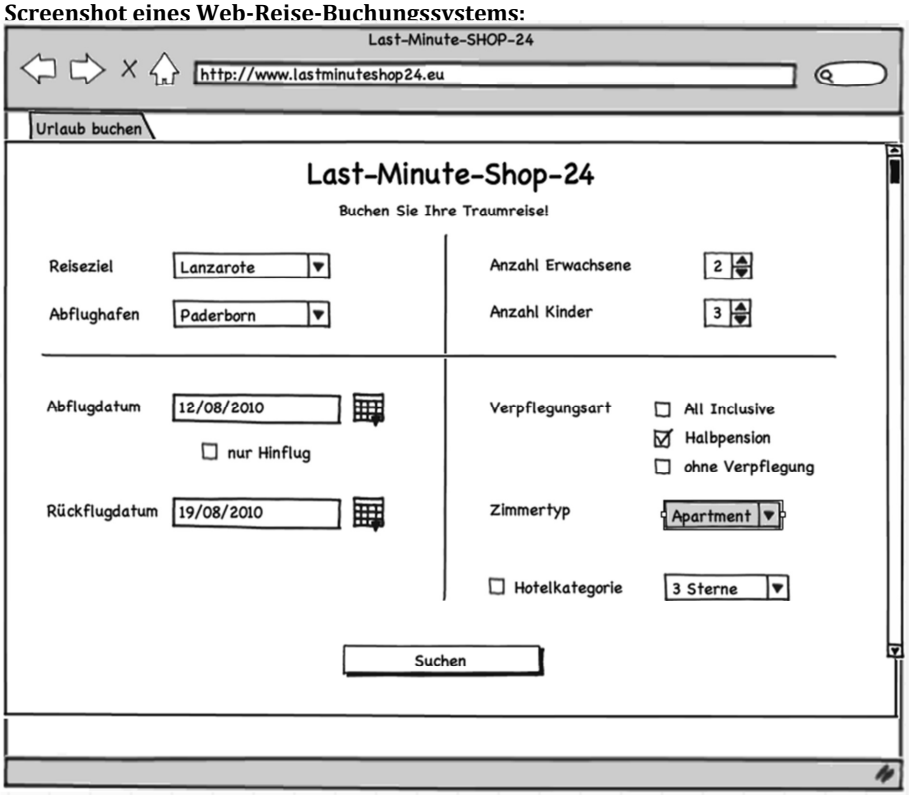
Anforderungsdefinition Reisebuchungssystem:

Anforderung 1: Benutzer kann Pauschalreisen suchen, indem er *Reiseziel*, *Abflughafen*, *Abflugdatum*, *Rückflugdatum* **(muss mindestens zwei Tage hinter dem Abflugdatum terminiert sein)**, *Anzahl Erwachsener* **(mindestens einer)**, *Anzahl Kinder*, *Verpflegungsarten* **(mindestens eine)** sowie *einen Zimmertyp* auswählt.

Anforderung 2: Der Benutzer kann ***optional die Hotelkategorie*** (Anzahl Sterne) mit in die Suche einbeziehen.

Anforderung 3: Benutzer kann auch **nur den Hinflug** buchen. Hierbei muss keine Eingabe in die Elemente der rechten Spalte gemacht werden.

Screenshot eines Web-Reise-Buchungssystems:



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Tabelle 1:

Testfall: Anforderung 1	Testfall: Anforderung 2	Testfall: Anforderung 3
Reiseziel: **Lanzarote**	Reiseziel: Lanzarote	Reiseziel:Palma de Mallorca
Abflughafen: **Paderborn**	Abflughafen: Paderborn	Abflughafen: Münster
Abflugdatum: **01.08.2010**	Abflugdatum: 01.08.2012	Abflugdatum: 02.08.2012
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ■
Rückflugdatum: **08.08.2010**	Rückflugdatum: 01.08.2012	Rückflugdatum:
Anzahl Erwachsene: **2**	Anzahl Erwachsene: 2	Anzahl Erwachsene:
Anzahl Kinder: **1**	Anzahl Kinder: 1	Anzahl Kinder:
Verpflegungsart: AI ❑; VP ■; OV ❑	Verpflegungsart: AI ■; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ❑
Zimmertyp: **Apartment**	Zimmertyp: Apartment	Zimmertyp:
Hotelkategorie: ❑	Hotelkategorie: ■ 4 Sterne	Hotelkategorie: ❑

ii) Überprüfen Sie die Robustheit der Anwendung, indem Sie nochmals den Auszug der **Anforderungsdefinition** betrachten und drei **unerwartete Testfälle** entwickeln, die die Anwendung zum Absturz bringen könnten. Ergänzen Sie diese Testfälle in **Tabelle 2**.

Tabelle 2:

Testfall: Fehleingabe 1	Testfall: Fehleingabe 2	Testfall: Fehleingabe 3
Reiseziel: **Lanzarote**	Reiseziel: Ibiza	Reiseziel: Menorca
Abflughafen: **Paderborn**	Abflughafen: Dortmund	Abflughafen: Hannover
Abflugdatum: **01.08.2010**	Abflugdatum: 21.05.2012	Abflugdatum: 21.06.2012
Nur Hinflug: ❑	Nur Hinflug: ❑	Nur Hinflug: ❑
Rückflugdatum: **25.07.2010**	Rückflugdatum: 15.05.2012	Rückflugdatum: 30.06.2012
Anzahl Erwachsene: **2**	Anzahl Erwachsene: 2	Anzahl Erwachsene: 0
Anzahl Kinder: **1**	Anzahl Kinder: 2	Anzahl Kinder: 2
Verpflegungsart: AI ■; VP ❑; OV ❑	Verpflegungsart: AI ❑; VP ■; OV ❑	Verpflegungsart: AI ❑; VP ❑; OV ■
Zimmertyp: **Apartment**	Zimmertyp: Einzelzimmer	Zimmertyp: Doppelzimmer
Hotelkategorie: ■ (3 Sterne)	Hotelkategorie: ❑	Hotelkategorie: ❑

Erläuterungen zu den Testfällen Fehleingabe 2, 3

- Fehleingabe 2: Trotz vier Personen ein Einzelzimmer →es sollte eine Fehlermeldung angezeigt werden
- Fehleingabe 3: Kinder buchen ohne Erwachsene

C)

i) Sie entwickeln eine Webseite für ein Reisebüro und befinden sich nach Abschluss der Implementierung in der Testphase. Um die Robustheit Ihres Softwareprodukts zu verbessern, sollen Betatester in den Prozess mit einbezogen werden. Welche Personen würden Sie in die Testphase mit einbeziehen? **(Mehrfachnennungen möglich)**

□ Die Entwickler des Reisebuchungssystem
√ Erfahrene Benutzer anderer Reisebuchungssysteme
√ Benutzer, die Grundkenntnisse in der Benutzung des Internets haben
□ Grundschüler, die gerade das Lesen gelernt haben

ii) Viele Betatester haben über Abstürze der Webseite berichtet. Wie gehen Sie vor, um die Eingaben in das System, die zum Absturz geführt haben, herauszufinden? Wie ermitteln Sie, was der Benutzer mit seinen Eingaben (die zum Absturz geführt haben)bezwecken wollte?

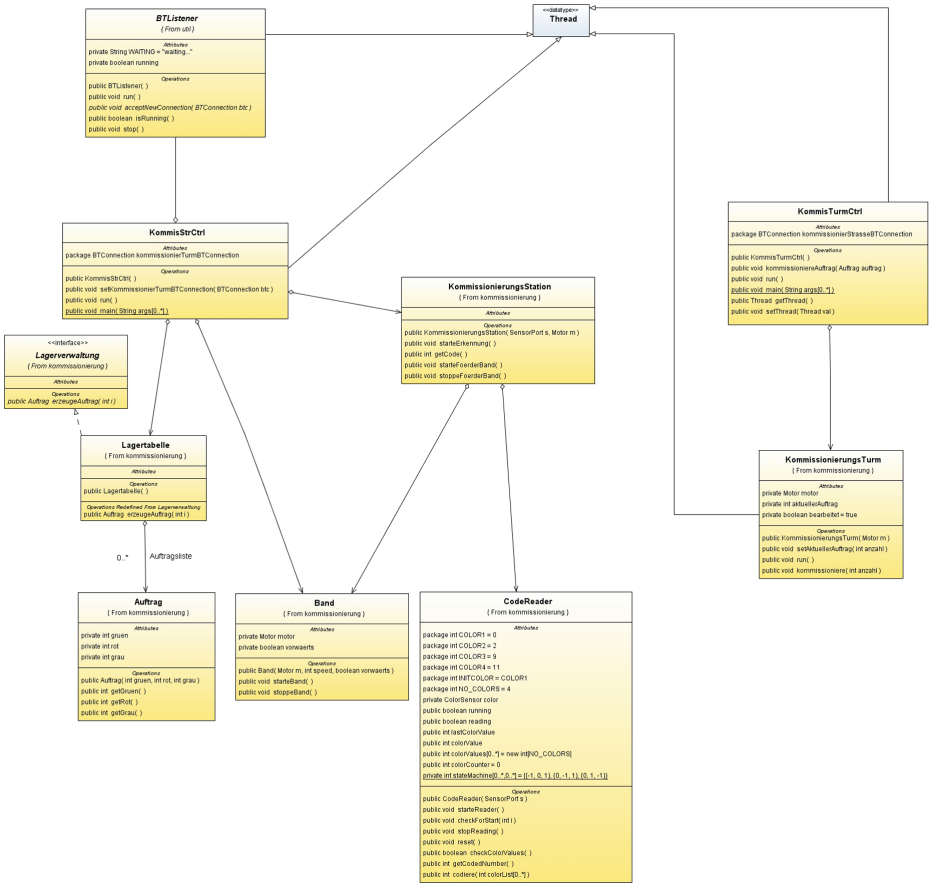
- Testprotokolle anfertigen lassen
- Dokumentation der Fehler mit Screenshots

...

A.3. Material zur Unterrichtserprobung

A.3.1. Klassendiagramm der Ausgangsversion der Kommissionierstation

Die folgende Abbildung zeigt das Klassendiagramm der Ausgangsversion (vor der Dekonstruktionsphase) der *Kommissionierstation*.



Klassendiagramm der *ILL-Kommissionierstation*

A.3.2. Ausbaustufen der Kommissionierstation im Quellcode

Die folgenden Tabelle zeigt die erforderlichen Anpassungen im Quellcode der *Java-Klassen*, um die Ausgangsversion der *Kommissionierstation* (vor der Dekonstruktionsphase) über die Ausbaustufe 1 (nach der Dekonstruktionsphase und vor der Konstruktionsphase) bis hin zur finalen Ausbaustufe 2 (nach der Konstruktionsphase) zu erweitern.

Klasse	Änderung
Auftrag	Zwei zusätzliche Objektvariablen(Integer) für die zusätzlichen Farben
alt: private int gruen; neu: private int gruen; **private int grau;** **private int rot;**	
Auftrag	Konstruktor um zusätzliche Objektvariablen erweitern.

alt:
public Auftrag(int gruen) {
this.gruen = gruen;
}
 	
neu:
public Auftrag(int gruen, **int grau, int rot**) {
this.gruen = gruen;
this.grau = grau;
this.rot = rot;
}	

Auftrag	get Methoden für zusätzliche Objektvariablen einfügen.
KommisStrCtrl	Objekt-konstante auftrag mit drei Integern konstruieren.

alt:
private static final Auftrag auftrag = new Auftrag(2);
 	
neu:
private static final Auftrag auftrag = new Auftrag(**1,2,3**);	

KommisStrCtrl	In der run-Methode dem DataOutputStream die zusätzlichen Farben hinzufügen

alt:
sender.sendeAuftrag(auftrag.getGruen());
 	
neu:
sender.sendeAuftrag(auftrag.getGruen(), **auftrag.getGrau(), auftrag.getRot()**);	

Klasse	Änderung
KommisTurmCtrl	Zusätzliche Kommissioniertürme als Objektvariablen hinzufügen.

alt:
KommissionierungsTurm k1; //Motor A
 	
neu:
KommissionierungsTurm k1; //Motor A
KommissionierungsTurm k2; //Motor B
KommissionierungsTurm k3; //Motor C	

KommisTurmCtrl	Im Konstruktor den Objektvariablen der zusätzlichen Kommissioniertürme, Objekte vom Typ KommissionierTurm zuweisen.

alt:
this.k1 = new KommissionierungsTurm(Motor.A); // gruen
 	
neu:
this.k1 = new KommissionierungsTurm(Motor.A); // gruen
this.k2 = new KommissionierungsTurm(Motor.B); // grau
this.k3 = new KommissionierungsTurm(Motor.C); // rot	

KommisTurmCtrl	Im Konstruktor die Threads der zusätzlichen Kommissioniertürme starten.

alt:
//start des Threads
k1.start();
 	
neu:
//start der Threads
k1.start();
k2.start();
k3.start();	

<table>
<tr><th>Klasse</th><th>Änderung</th></tr>
<tr><td>KommisTurmCtrl</td><td>Die Methode kommissioniereAuftrag muss an alle drei Türme die Anzahl der zu kommissionierenden Steine übermitteln.</td></tr>
<tr><td colspan="2">alt:
public void kommissioniereAuftrag(Auftrag auftrag) {
k1.setAktuellerAuftrag(auftrag.getGruen());
}

neu:
public void kommissioniereAuftrag(Auftrag auftrag) {
k1.setAktuellerAuftrag(auftrag.getGruen());
k2.setAktuellerAuftrag(auftrag.getGrau());
k3.setAktuellerAuftrag(auftrag.getRot());
}</td></tr>
<tr><td>KommisTurmCtrl</td><td>In der run-Methode aus dem DataInputStream die zusätzlichen Farben auslesen und in Auftragobjekt speichern.</td></tr>
<tr><td colspan="2">alt:
farben = bte.empfangeAuftrag(1);
System.out.println(""+ farben[0]);
Auftrag auftrag = new Auftrag(farben[0]);
kommissioniereAuftrag(auftrag);

neu:
farben = bte.empfangeAuftrag(3);
System.out.println(""+ farben[0] + “ : “ + farben[1] + “ : “ + farben[2]);
Auftrag auftrag = new Auftrag(farben[0], farben[1], farben[2]);
kommissioniereAuftrag(auftrag);</td></tr>
</table>

Klasse	Änderung
Auftragstabelle	Neu zu erstellende Klasse, die eine Hartgecodete Tabelle enthält, die jedem Bytecode eines RFID-Chips einen Auftrag zuweist. Dafür gibt es zwei Methoden. DIe Methode getAuftragNummer ermittelt anhand eines Bytecodes die Nummer des Auftrags. Die Methode getAuftrag gibt mithilfe der Methode getAuftragNummer den richtigen Auftrag zurück.
KommisStrgCtrl	import des Pakets nxt.addon
import lejos.nxt.addon.*;	
KommisStrgCtrl	Deklarieren und initialisieren des neuen RFID-Sensor und der Auftragstabelle

private BluetoothSender sender;
private Auftrag auftrag;
private RFIDSensor rfid;
private Auftragstabelle at;
 	
public KommisStrCtrl() {
super();
sender = null;
ks = new KommissionierungsStation(SensorPort.S1, Motor.B);
pb = new Band(Motor.A, 200, true);
rfid = new RFIDSensor(SensorPort.S2);
at = new Auftragstabelle();
}	

KommisStrgCtrl	In der Run-Methode wird der Transponder ausgelesen und aus der Auftragstabelle die Methode getAuftrag aufgerufen.

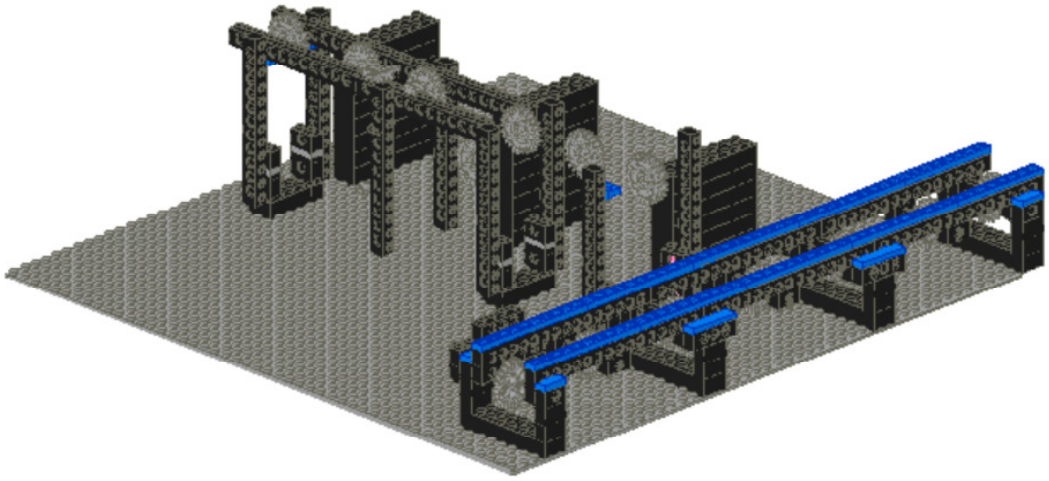
ks.starteErkennung();
pb.stoppeBand();
byte[] id = rfid.readTransponder(false);
auftrag = at.getAuftrag(id);	

KommisStrgCtrl	Die Konstante auftrag wird als variable deklariert.
private Auftrag auftrag;	

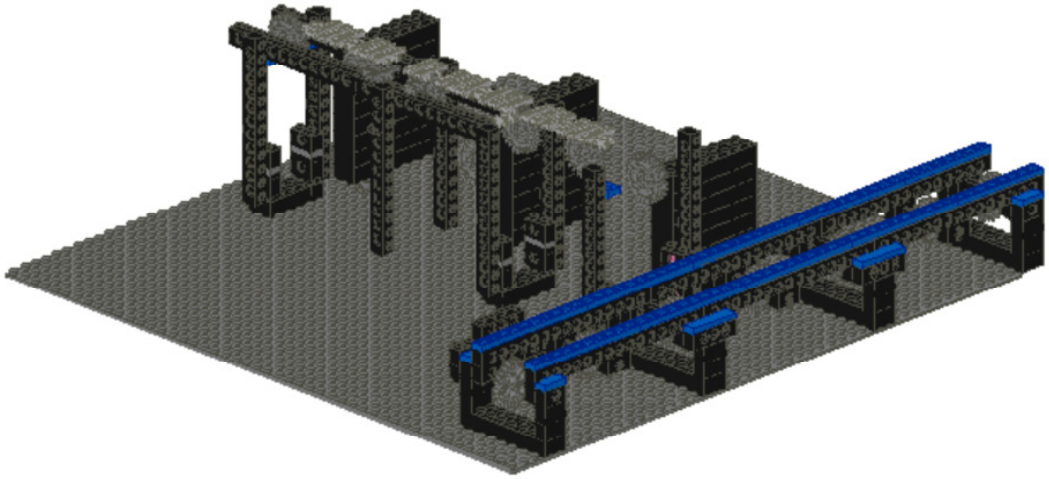
A.3.3. Bauanleitung der Kommissionierstation

Die folgende Bauanleitung für das LEGO-Modell der *Kommissionierstation* wurde von den Studierenden im Rahmen des Seminars *Informatik Lernlabor (SoSe 08)* erstellt.

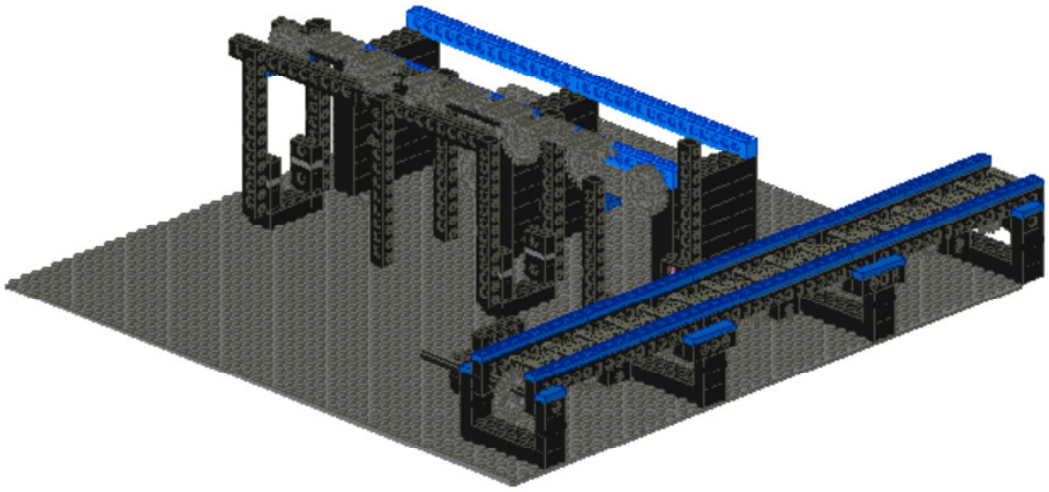
1



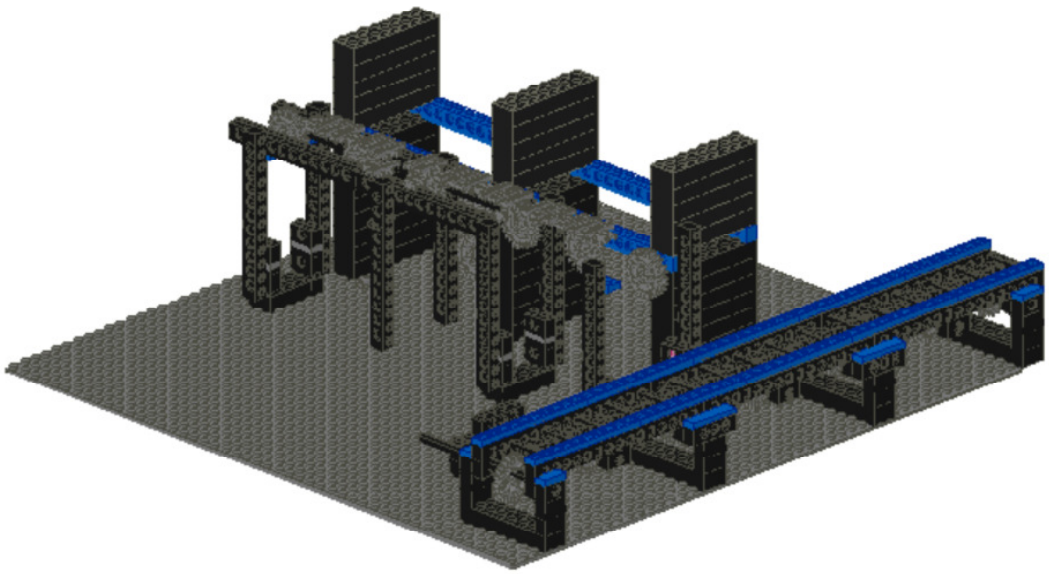
2



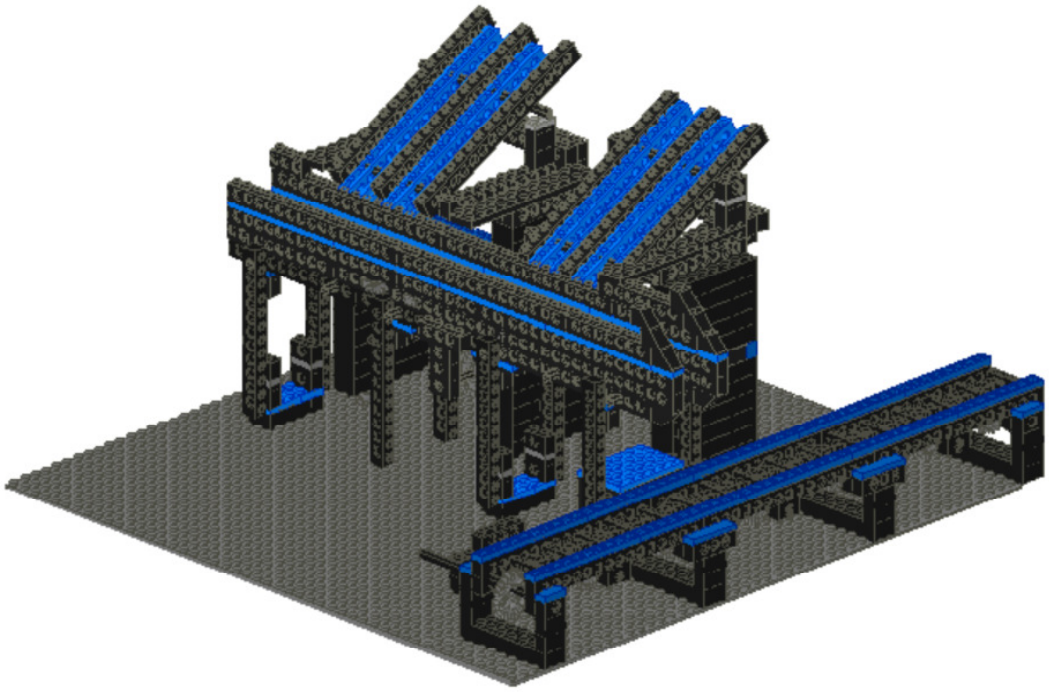
3



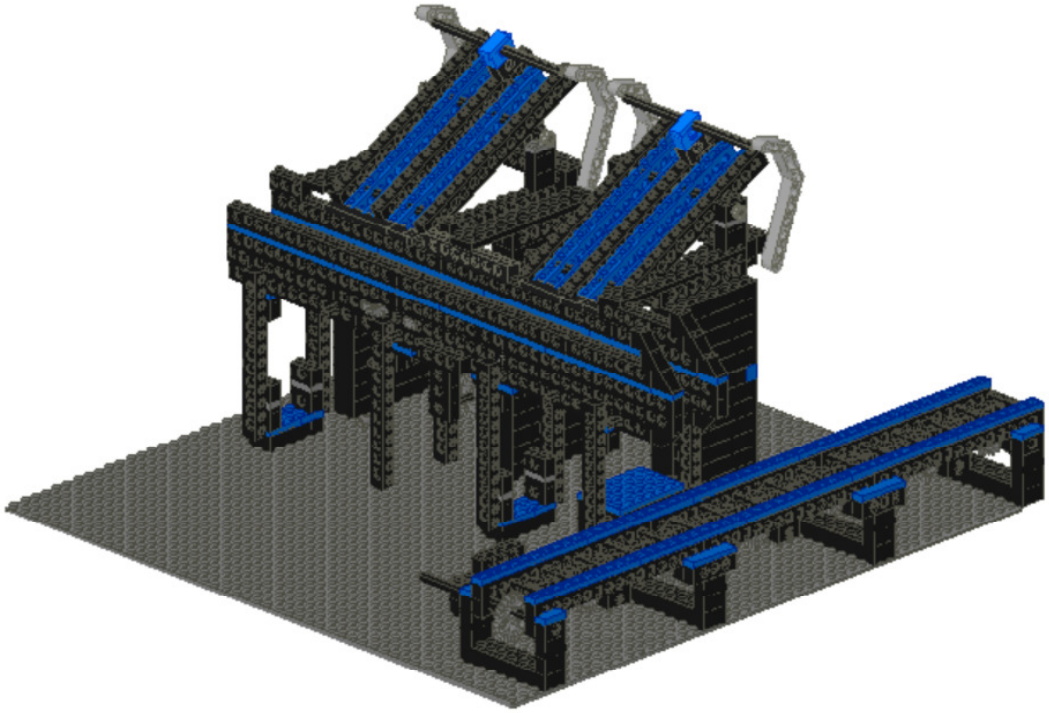
4



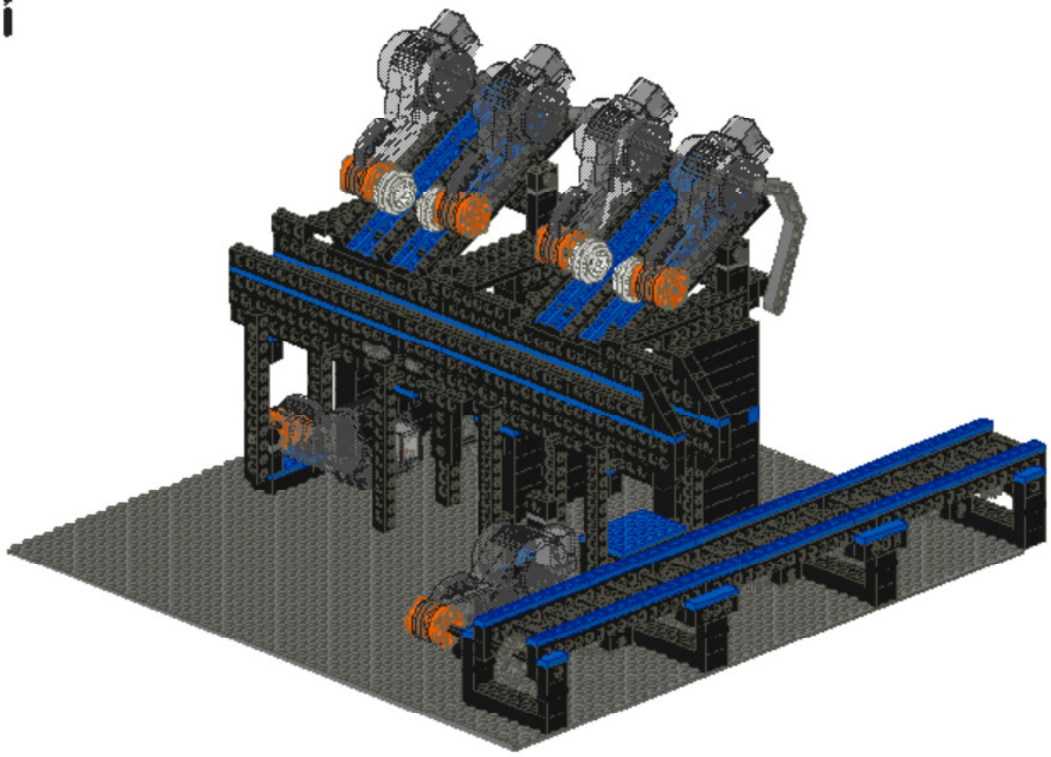
5



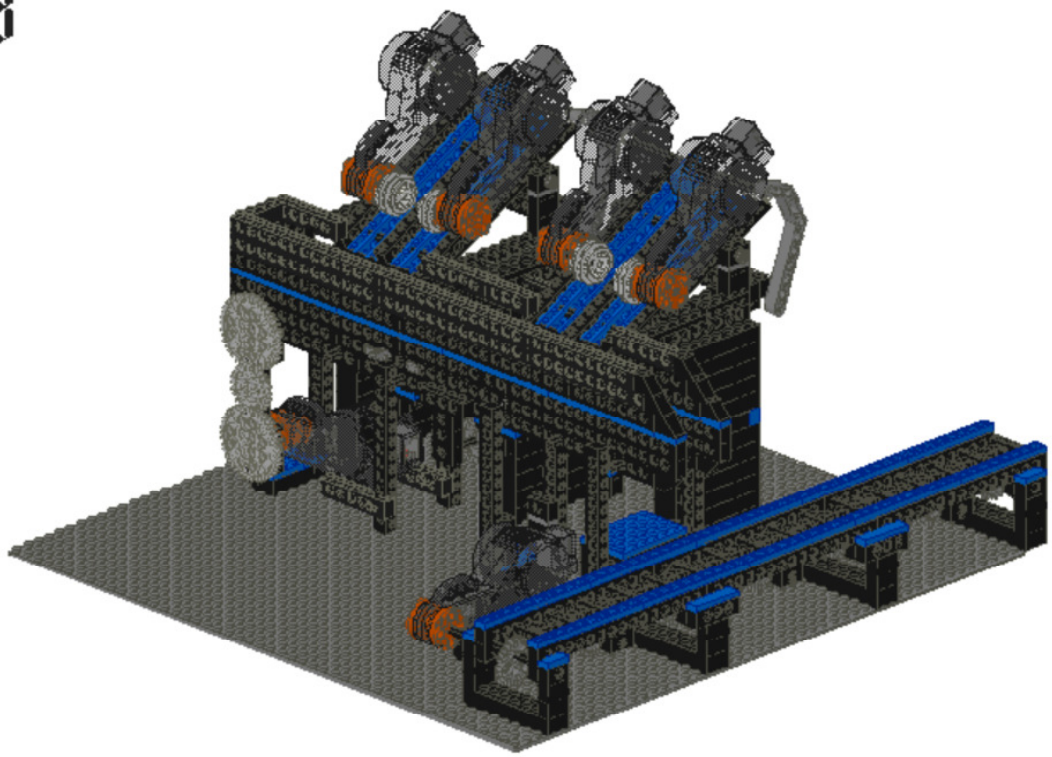
6



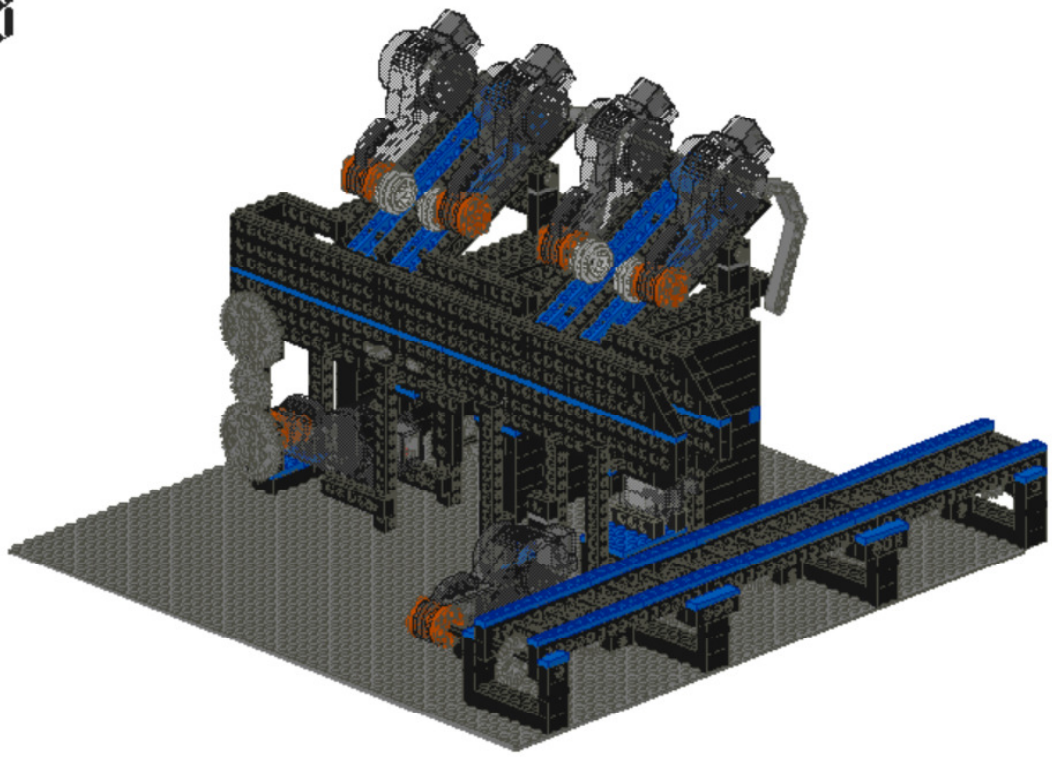
7



8



9



10