Decoding the Disciplines DIGITAL

Software-Architektur als ganzheitlicher Prozess in der Hochschulbildung

1 Problemstellung

Was veranlasst Sie zu der geplanten Lehrinnovation? Welches Problem soll bearbeitet werden? Inwieweit handelt es sich dabei um ein zentrales Problem in der Lehre im jeweiligen Studienfach?

Die fortschreitende Digitalisierung wird Gesellschaft, Wirtschaft und Wissenschaft im Laufe der nächsten Dekaden nachhaltig umgestalten. Durch die ubiquitäre Verbreitung von IT-Technologie wird Software immer komplexer. Softwaresysteme der Zukunft werden noch stärker als heute zu großen IT-Landschaften vernetzt sein, deren Komplexität nach neuen, innovativen Softwarekonzepten ruft.

Für derartige Konzepte ist in der Informatik die *Software-Architektur* zuständig. Software-Architekten entwerfen Baupläne komplexer IT-Systeme und -Landschaften, die dann im Zusammenspiel verschiedener IT- und Fachexperten umgesetzt oder transformiert werden. Die Softwaretechnik hält für Software-Architekten einen umfangreichen „Methoden-Werkzeugkasten“ bereit, um einen solchen Systementwurf erstellen und beschreiben zu können. Dazu zählen Modellierungssprachen wie z.B. UML, Vorgehensmodelle, Entwurfsmuster, Architekturstile und vieles mehr. Diese Methoden beherrscht ein erfahrener Architekt so flüssig und intuitiv wie ein Konzertmusiker sein Instrument.

In der Berufswirklichkeit wird man erst nach einigen Jahren praktischer Erfahrung vom Software-Entwickler zum Architekten. In der Hochschullehre müssen die Grundlagen der oben beschriebenen Kernfähigkeiten im 3./4. Semester des Informatik-Bachelorstudiums vermittelt werden. Dies ist eine große Herausforderung für Studierende und Lehrende gleichermaßen. In der Berufspraxis begegnet man Fragen wie den folgenden:

(A) „Mein Projekt ist so umfangreich, dass wir parallel mit mehreren Teilteams daran arbeiten müssen. Mit welchem Ansatz lege ich am besten einen sinnvollen Systemschnitt für die Arbeitsverteilung fest?“

(B) „Wie dokumentiere ich meine Entwurfsentscheidungen? Welchen Modelltyp nutze ich in welcher Projektsituation?“

(C) „Sind alle Architektur-Diagrammtypen gleich wichtig? Und in welcher Detailtiefe sollte ich sie beim Entwurf einsetzen?“

Der klassische Lehransatz tendiert dazu, sich auf die reine Vermittlung von Werkzeugen und Methoden (UML-Modellierung, Liste der Entwurfsmuster, …) zu fokussieren. Dabei fallen die drängenden Fragen der Studierenden nach dem *Kontext* (was nutze ich wann wofür in welchem Detaillierungsgrad, welche Fallstricke gibt es, etc.) oft unter den Tisch.

Einen zentralen Fokus der Kompetenzorientierung deckt diese Art von Lehre damit gerade nicht ab. Architekturkenntnissen werden in diesem Fall als *Inselwissen* erworben, mit ungenügender Anbindung an andere Studieninhalte des Informatikstudiums. Software-Architektur wird von den Studierenden als mühselig, komplex und ohne praktischen Nutzwert erlebt. Ein so geprägtes Entwicklungsteam neigt dazu, ohne hinreichende Entwurfsphase direkt in die Umsetzung einzusteigen. Negativeffekte wie der einer *Zufallsarchitektur* („Accidental Architecture“, [5]) sind die Folge.

Die Fokussierung auf projektorientierte Lehre kann (nach eigener Beobachtung verschiedener Lehrveranstaltungen) paradoxerweise sogar zu einer Verstärkung dieses Effektes führen, falls nicht die Software-Architektur oder andere größere, verbindende Kontexte ausdrücklich im Fokus stehen.

Studienprojekte sind in der Regel *kleine* Softwaresysteme, die von den Studierenden unter hohem Zeitdruck realisiert werden. Entwicklungsphasen ohne sofort erkennbaren Nutzwert, wie etwa der Architektur-Entwurf oder das systematische Testen, werden dann von den Studierenden gern weggelassen oder zumindest sehr oberflächlich behandelt.

Die Frage (A) aus obiger Liste (nach einem ersten Systemschnitt für die Arbeitsverteilung) stellt sich bei kleinen Studienprojekten schlichtweg nicht – oder sie wird intuitiv durch einen Schnitt entlang der verwendeten Haupttechnologien (meist User Interface und Backend/Datenbank) beantwortet. Dieser Effekt lässt sich in Studentenprojekten immer wieder beobachten. Gemäß Conway's Law [7] prägt die Organisationsstruktur des Entwicklungsteams maßgeblich die Architektur des Systems. Für ein *größeres* produktives IT-System lässt sich nachweisen, dass dieser Arbeitsansatz zu einer sogenannten Monolithen-Architektur führt [15], die langfristig eine Weiterentwicklung der Software stark erschwert.

So erhalten Studierende ungewollt einen falschen Anreiz in Richtung von schnellen, aber wenig nachhaltigen Ergebnisse bei der Softwareentwicklung.

1.1 Ziele von *Decoding the Disciplines DIGITAL*

Welche Ziele verfolgen Sie mit der geplanten Lehrinnovation?

An dieser Stelle versucht mein Lehrkonzept einen praxisnahen Gegenentwurf, mit drei wesentlichen Zielen:

1. **Feingranulare Kompetenzbeschreibung**: Als Lehrender möchte ich die Lernräume der Software-Architektur in Form einer angemessen *feingranularen*, *hierarchisch strukturierten* und *nutzwertorientiert-kontextualisierten* Form beschreiben, um dem Anti-Pattern „Software-Architektur als zweckfreies Inselwissen" zu begegnen.
2. **Kompetenzdiagnostisches Werkzeug**: Als Studierende(r) wünsche ich mir ein *kompetenzdiagnostisches Werkzeug*, das mir aufzeigt, wo ich im Lernprozess stehe und wo meine individuellen Stärken und Schwächen liegen.
   Als Lehrender möchte ich dasselbe Werkzeug nutzen, um einzuschätzen, wo die *Gesamtheit der Studierenden* Schwächen hat und damit vermutlich entweder meine Ziele überambitioniert sind, oder meine Veranstaltung nicht gut ist. (Und umgekehrt interessiert mich natürlich auch, wo meine Lehre gut funktioniert.)
3. **„Eat your own Dogfood"**: Als Studierende(r) möchte ich die gelernten Architekturprinzipien und -Methoden in Ruhe und mit inhaltlichem Feedback ausprobieren können – idealerweise, indem ich genau *das* Werkzeug als komplexes Softwareprojekt weiterentwickle, das ich in meiner Architektur-Lehrveranstaltung selbst genutzt habe.

2 *Decoding the Disciplines DIGITAL* als ganzheitlicher Lehransatz

Warum bewerben Sie sich um ein Fellowship? (persönliche Motivation)

Die überwiegende Zeit meines beruflichen Lebens, vor meiner Zeit als Hochschulprofessor, habe ich als Software-Architekt verbracht. Ich habe Teams von Softwarearchitekten geleitet, Architekten ausgebildet und Kunden beim Aufbau von Softwarearchitektur-Abteilungen unterstützt. Die Frage, was genau einen guten Software-Architekten ausmacht, und wie man die nötigen Fähigkeiten am besten vermittelt, beschäftigt mich also schon sehr lange.

An der TH Köln habe ich nach meiner Berufung unter anderem das Fach *Softwaretechnik* übernommen, das Informatik-Bachelorstudenten im 3./4. Semester in Software-Architektur ausbildet. Bei der

Neukonzeption des didaktischen Ansatzes für dieses Fach wollte ich unbedingt einen innovativen Weg gehen.

An der TH Köln durchläuft man als Neuberufener ein intensives hochschuldidaktisches Ausbildungs- und Coachingprogramm. Für mich war dies in gewisser Weise ein Augenöffner. Drei didaktische Kernkonzepte bilden das Fundament meines Lehrkonzeptes, die ich dafür kombiniert, praktisch angewendet und in Teilen erweitert habe:

- *Revised Bloom's Taxonomy* [1] als Methode zur Differenzierung von Inhalten,
- *Learning Outcomes* (LO) [18] in einer erweiterten Form als durchgehende Spezifikationssprache der Veranstaltungsinhalte, sowie
- *Decoding the Disciplines* [12] als Vorgehensmodell.

2.1 Taxonomiestufen 4 – 6 in der Softwarearchitektur

Revised Bloom's Taxonomy [1] liefert eine sehr anschauliche formale Erklärung für die oben beschriebenen Probleme traditioneller Software-Architekturausbildung. Eine auf Details von Modellierungsnotationen fokussierte „Inselwissen"-Lehre bewegt sich hauptsächlich auf der Stufe 3 (*Applying*) der Taxonomie. Dergestalt ausgebildet, ist ein Studierender in der Lage, auf eine vorgegebene Problemstellung ein Modellierungsmuster anzuwenden. Der Praxistransfer scheitert dann aber häufig daran, dass diese Problemstellung „in Reinform" im praktischen Projektkontext gar nicht erkennbar ist. Hierfür benötigt der angehende Software-Entwickler und -Architekt die weitergehenden Taxonomiestufen 4 - 6:

- **Stufe 4** (Analysing) ermöglicht es, ein konkretes Projekt auf die Anwendbarkeit gelernter Architektur-Muster und -Stile hin zu analysieren. Das erst die Voraussetzung für die Anwendung des eigenen Wissens.
- Hochschul-Lehrprojekte sind in aller Regel sog. *Greenfieldprojekte*, also komplette Neuentwicklungen eines Softwaresystems. In der Praxis überwiegen bei weitem aber *Brownfieldprojekte*, Weiterentwicklung bereits existierender Softwareprojekte. Hier ist die **Stufe 5** der Taxonomie (*Evaluating*) gefragt, um die Qualität der bestehenden Architektur zu bewerten und über das Maß an notwendiger Umgestaltung (das sogenannte *Refactoring* [9]) zu entscheiden.
- Bei Projekten, in denen technologisches Neuland betreten wird, genügen bekannte Architekturstile oft nicht. Hier muss ein Softwarearchitekt in der Lage sein, durch Synthese und Weiterentwicklung bekannter Ansätze neue, geeignete Architekturmuster zu entwerfen. Dies entspricht **Stufe 6** (*Creating*) der Taxonomie. Der Bedarf hierfür wird sich fortschreitender Digitalisierung unser Wirtschaft und Gesellschaft noch verstärken.

Mein Lehrkonzept fokussiert daher darauf, die Kompetenzstufen 4-6 für das Fach Software-Architektur mit einem detaillierten, praxisorientierten Schritt-für-Schritt-Vorgehensmodell zu spezifizieren und den Studierenden ausgiebig Möglichkeiten zum praktischen Einüben zu geben.

2.2 *Hierarchische Learning Outcomes* als Erweiterung von *WAS-WOMIT-WOZU*

Die *WAS-WOMIT-WOZU*-Form von Learning Outcomes [18] ist eine unmittelbar einleuchtende Form der Kompetenzspezifikation. Für die Ziele (1) und (2) aus Kap. 1.1 – feingranulare Kompetenzbeschreibung und eine darauf abgebildete Kompetenzdiagnostik – wurde dieser Ansatz erweitert. Inspiriert durch die Form der *User Story* [6] (ein Beschreibungselement für Anforderungen aus der agilen Softwareentwicklung), wird die gesamte Veranstaltung schrittweise in (Teil-)Kompetenzen heruntergebrochen, die dann – je nach Bedarf – wiederum hierarchisch feiner aufgegliedert werden können.

Der gesamte erste Veranstaltungsteil von Softwaretechnik liegt mittlerweile vollständig spezifiziert vor [2], in 24 Teil-LOs aufgegliedert. Eines davon zeigt Tabelle 1. Das dargestellte LO ist im Übrigen gerade dasjenige, die die Frage von Studierenden nach einem ersten Systemschnitt (siehe (A) auf Seite

1) adressiert. Die Zahlen stehen für die entsprechende Taxonomie-Stufe. Der „*vorausgesetzt, ich kann*“-Teil fasst notwendiges Wissen zusammen, das der Studierende zunächst in Selbststudium und Präsenzveranstaltung erwerben muss, bevor die im „*indem ich*“-Teil aufgelisteten Teilkompetenzen sinnvoll erarbeitet und eingeübt werden können.

Tabelle 1: Beispiel für ein *Hierachisches Learning Outcome*

Als	Softwarearchitekt oder Softwareentwickler		
kann ich *(WAS)*	durch Use Cases, fachliches Datenmodell und Clustering eine erste Komponentenstruktur	definieren	5
vorausgesetzt, ich kann	den Nutzen einer initialen Grobstruktur zur Verhinderung einer Zufallsarchitektur für mein Team und mich	begründen	2
indem ich *(WOMIT)*	Use Cases aus der Aufgabenstellung	ableite	4
	das fachliches Datenmodell und das Glossar aus der Aufgabenstellung	ableite	4
	durch Clustering Subkomponenten	bilde	3
	die Cluster in eine erste Komponentenstruktur	transformiere	3
und abschließend	mit Hilfe der 7-Fragen-Methode die Unklarheiten / Aspekte zur Weiterentwicklung meines Entwurfs	analysiere	5
so dass ich *(WOZU)*	einen Startpunkt für die weitere Entwurfs- und Entwicklungsarbeit habe.		

Die Kompetenzen aus dem „*vorausgesetzt, ich kann*“-Teil sind wissens- und verständnisorientiert und bewegen sich auf Taxonomiestufe 1 und 2. Das Beispiel in Tabelle 1 macht exemplarisch deutlich, dass auch diese Stufen in der späteren Berufspraxis durchaus unmittelbar nutzbar sein können. Wenn ich als Software-Architekt gut argumentieren kann, warum eine initiale Investition in Architektur sich später für das Team auszahlen wird, habe ich es leichter, mit meinen Entwurfsideen akzeptiert und verstanden zu werden.

Das LO aus Tabelle 1 stellt eine zusammenfassende Kompetenz auf relativ hoher Ebene dar. Das erkennt man daran, dass die Teilfähigkeiten im „*indem ich*“-Teil sämtlich Links auf detailliertere LOs sind, in denen die methodischen Schritte genau spezifiziert sind. Diese sind als Hyperlinks blau unterstrichenen dargestellt. Diese Teilffähigkeiten bewegen sich auf Taxonomiestufe 3-6.

2.3 Die Softwareplattform von *Decoding the Disciplines DIGITAL*

Decoding the Disciplines DIGITAL ist sowohl Lehrkonzept wie auch (inhaltlich eng angekoppelte) Software-Plattform. Letztere Plattform, die aktuell in Entwicklung befindlich ist, bietet als zentrales Modul die Erfassung und Verwaltung der hierarchischen Learning Outcomes an (siehe mittlerer Kasten in Abbildung 1).

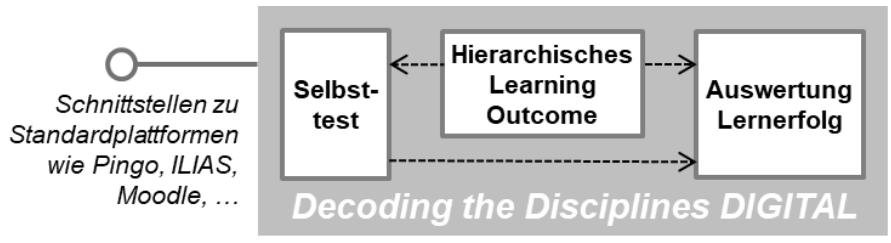

Abbildung 1: Bestandteile der *Decoding the Disciplines DIGITAL* Plattform

Diese hierarchischen LOs erfüllen in dem Lehrkonzept allgemein, und auch in der digitalen Plattform, eine zentrale, doppelte Funktion. Einerseits strukturieren sie alle Arten von Lehrveranstaltungen – Vorlesung, Übung, Praktikum, Klausur und auch weitergehende projektorientierte Lehre. Für jeden inhaltlichen Bestandteil der Lehrveranstaltungen wird auf feingranularer Ebene (noch unterhalb der Lernräume) nachgehalten, auf welches (Teil)-LO es einzahlt. Damit wird Ziel (1) aus Kap. 1.1 realisiert.

Die beiden weiteren Module *Selbsttest* und *Auswertung Lernerfolg* (linker und rechter Kasten in Abbildung 1) beziehen sich konzeptionell (und in der Softwareplattform als Verlinkung realisiert) auf die Teil-LOs. Jede Aufgabe aus dem *Selbsttest*-Modul und jedes Einzel- oder aggregiertes Gruppenergebnis aus *Auswertung Lernerfolg* ist damit direkt einer (Teil-)Fähigkeit aus einem LO zugeordnet. Abbildung 2 zeigt diese Verlinkung an einem Beispiel für das *Selbsttest*-Modul.

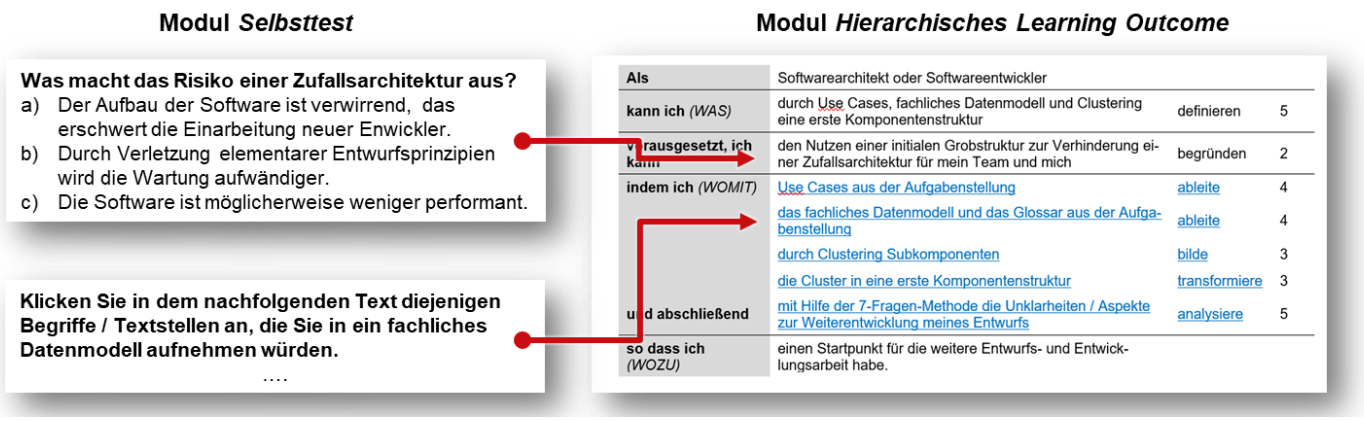

Abbildung 2: Verlinkung der Module *Hierarchisches Learning Outcome* und *Selbsttest* (Beispiel; in der dargestellten Quizfrage sind alle Antworten richtig.)

Durch diese Verlinkung auf konzeptioneller und softwaretechnischer Ebene wird das Ziel (2) des Lehrkonzepts erreicht. Sowohl als Studierender wie als Lehrender erhalte ich eine Art von „Heatmap“ der aktuell erreichten Kompetenzen. Es wird also unmittelbar klar, wo ich als Lernende(r) stehe, bzw. welche Teile der Veranstaltung gut (und welche weniger gut) funktionieren.

Damit wird auch deutlich, dass die Softwareplattform als Teil des Lehrkonzepts essentiell ist, und eben nicht nur „yet another e-learning platform“ ist. Ein Online-Selbsttest-Modul (das es – isoliert betrachtet – unbestritten schon vielfach gibt) ist im wesentlichen deshalb Teil der Plattform, um diese als Ganzes eigenständig betreiben zu können. Es sind auch, wie in Abbildung 1 dargestellt, Schnittstellen zu etablierten Plattformen wie Pingo [3], Moodle [16] und ILIAS [10] vorgesehen.

Ein weiterer Grund für genau diese Bestandteile der Softwareplattform ist das „Eat your own Dogfood“-Ziel (3) (siehe Kap. 1.1) des Lehrkonzepts. Die Plattform erfüllt eine Doppelfunktion: Die Studierenden nutzen die Software in den Lehrveranstaltungen, und gleichzeitig entwickeln ***die Studierenden selbst*** diese Plattform in nachfolgenden Studienprojekten als ein studentisches Langzeit-Softwareprojekt weiter.

Die Architektur der Plattform folgt dem Microservices-Architekturstil [17], einem noch relativ neuen Paradigma, das als besonders geeignet für dezentral strukturierte Systeme gilt. Es wird daher in der Fachwelt als interessanter Ansatz für die Herausforderungen der Digitalisierung betrachtet. Für die *Decoding the Disciplines DIGITAL* Softwareplattform stellt es eine anspruchsvolle Entwurfs- und Umsetzungsaufgabe dar, an der sich viele Herausforderungen der modernen Informatik sehr gut ausprobieren und üben lassen.

3 Einbettung in die Lehre gemäß *Decoding the Disciplines*

In welche Studiengänge und -abschnitte soll die geplante Lehrinnovation implementiert werden? Handelt es sich dabei um den Pflicht-, Wahlpflicht- oder Wahlbereich?

Die geplante Lehrinnovation setzt die Software-Architekturausbildung gemäß „*Decoding the Disciplines*“-Ansatz im Informatik-Bachelor- und -Master-Studiengang im Rahmen einer ganzen Anzahl von Pflicht- und Wahlpflichtfächern um, wie in Abbildung 3 dargestellt.

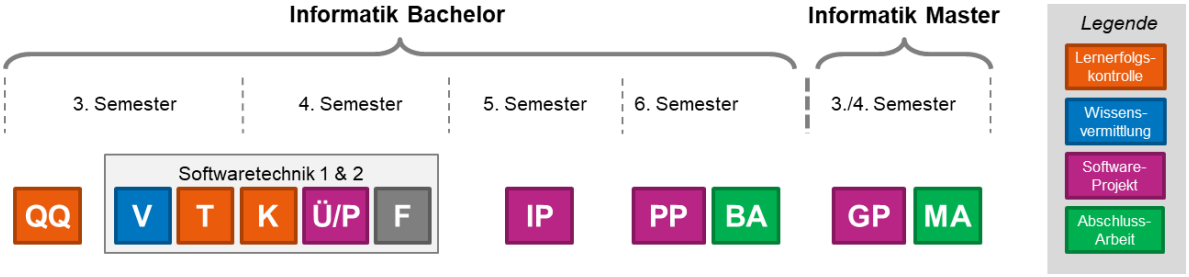

Abbildung 3: In das Lehrkonzept eingebundene Lehrveranstaltungen im Informatik-Bachelor- und -Master

- Zentrales Modul ist die Pflichtveranstaltung „Softwaretechnik“ mit 2 x 5 CP im 3. und 4. Semester des Bachelorstudiums. Sie setzt sich wie folgt zusammen:
  - Ein Blended-Learning-Vorlesungsteil (**V**) mit Diskussion komplexer Inhalte
  - Eine Möglichkeit von Selbsttests von Online-Architektur-Aufgaben (**T**); dies erfolgt zurzeit noch nicht, sondern ist Teil der geplanten Lehrinnovation
  - Eine (zurzeit papierbasierte) Klausur (**K**); mittel-/langfristig in Onlineform
  - Ein projektorientierter Übungs-/Praktikumsteil in Kleingruppen (**Ü/P**)
  - Verschiedene Feedback-Kanäle (**F**), darunter ein regelmäßiger TAP [14] und eigene Onlinebefragungen der Studierenden

Neben dieser Pflichtveranstaltung im 3./4. Semester nutze ich eine Anzahl von Wahlpflichtveranstaltungen, um den Studierenden ein aufeinander aufbauendes Angebot mit Bezug zu Architekturthemen zu machen.

- Begleitend zu ST1/2 biete ich jedes Semester ein „Querschnitts-Qualifikationsprojekt“ (**QQ**) an, bei dem Studierende Selbstorganisation und -management einüben sollen. Im Rahmen dieses QQ-Projekts erstellen die Studierenden begleitend zum Besuch der Softwaretechnik-Veranstaltung Architektur-Aufgaben für das Selbsttestmodul.
- Im Informatikprojekt (**IP**) im 5. Semester (10 CP) bearbeiten die Studierenden in Teams ein praktisches Softwareprojekt. Hier biete ich jedes Semester ein Projekt an, bei dem die Studierende die Softwareplattform von *Decoding the Disciplines DIGITAL* weiterentwickeln. Dieses Projekt wurde bis jetzt zweimal durchgeführt, mit großer Resonanz (fast die Hälfte der ST1/2-Absolventen haben sich für dieses Wahlpflichtprojekt entschieden).
- Praxisprojekt (**PP**) und Bachelorarbeit (**BA**) finden im 6. Semester statt. Auch hier biete ich forschungs- und entwicklungsorientierte Software-Architekturthemen am konkreten Beispiel der *Decoding the Disciplines DIGITAL* Plattform an.
- Auch im Master-Studiengang Informatik gibt es forschungsorientierte Projekte (Guided Projects, **GP**) sowie die Masterarbeit (**MA**), bei denen ich zusammen mit Mitarbeitern in analoger Weise Architekturthemen anbiete.

3.1 *Decoding the Disciplines* als Vorgehensmodell

Middendorf und Pace [12] liefern in ihrem Artikel *Decoding the Disciplines* eine Lösung für Vermittlung komplexen Expertenwissens vor (siehe Abbildung 4).

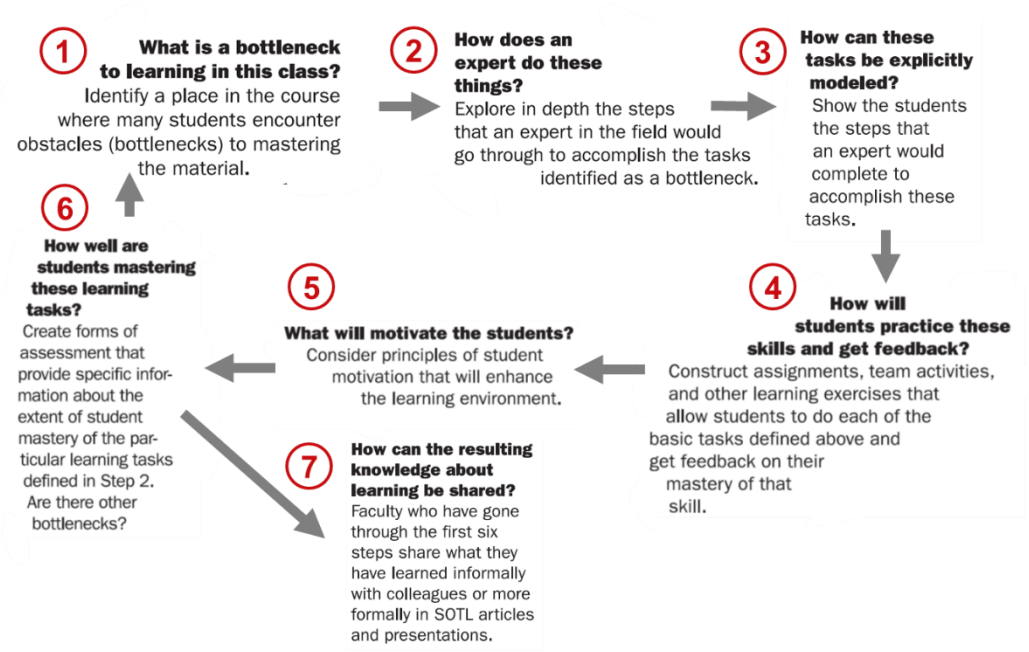

Abbildung 4: Die sieben Schritte von „Decoding the Disciplines“ [12] (graphisch modifiziert)

Das vorgeschlagene Lehrkonzept nutzt die Schritte aus *Decoding the Disciplines*, um den Studierende die Aneignung der „Experten“-Taxonomiestufen 4 – 6 zu erleichtern. Die Funktionsweise der einzelnen Schritte ist nachfolgend dargestellt. Abbildung 5 gibt eine Übersicht. Die in der Abbildung mit Nummern gekennzeichneten einzelnen Schritte werden nachfolgend erläutert.

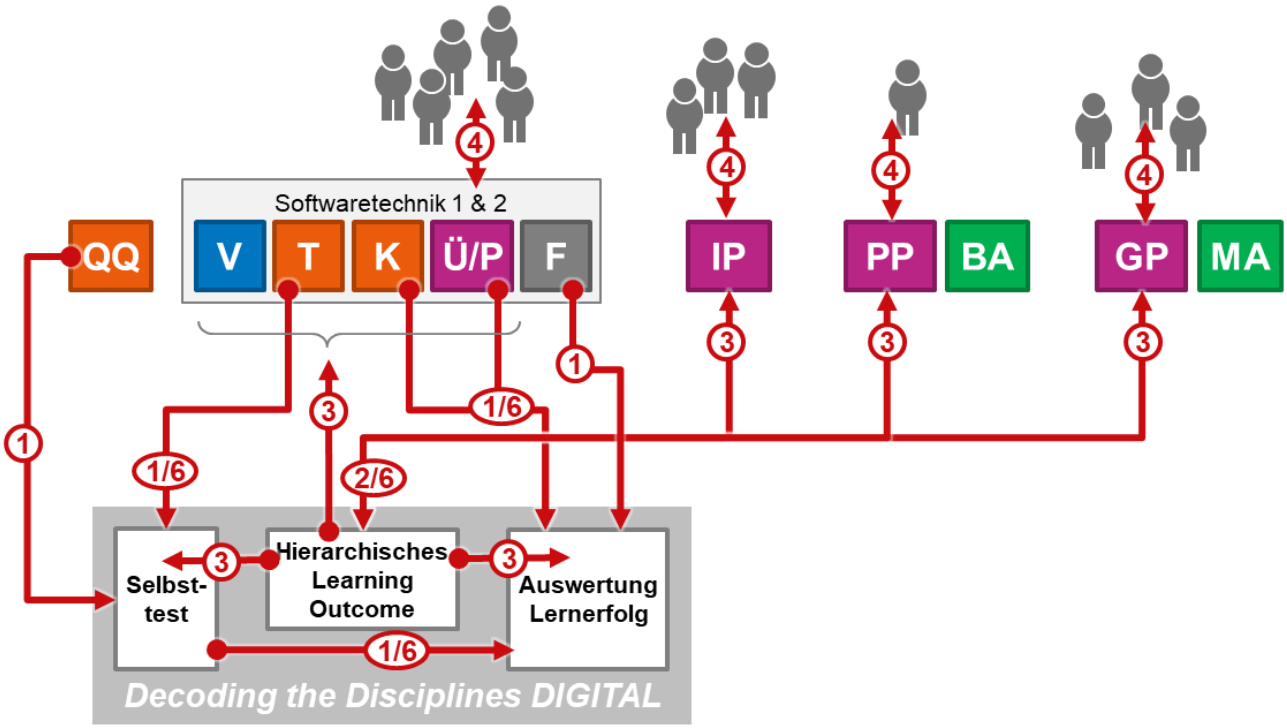

Abbildung 5: Transferschritte in *Decoding the Disciplines DIGITAL*

3.1.1 Schritt 1 – What is a bottleneck to learning in this class?

Die verschiedenen Kanäle für Lernstandskontrolle (**Ü/P**, **K**) sowie Feedback (**F**) liefern ein kontinuierliches Bild, an welchen Stellen die Studierenden sich mit dem Stoff schwertun. Sobald das *Selbsttest*-Onlinemodul produktiv verfügbar ist, liefern die aggregierten Ergebnisse der Studierenden (**T**) ein präzises Bild. Auch die in **QQ** gesammelten Fragen stellen ein Indiz dar: sinnvolle Fragen und Aufgaben deuten auf ein gutes Verständnis hin, unsinnige oder fehlende Fragen / Aufgaben auf Schwierigkeiten mit bestimmten Teilfähigkeiten.

3.1.2 Schritt 2 – How does an expert do these things?

Praxisprojekte (**PP**) und Guided Projects (**GP**) im Master sind softwaretechnisch anspruchsvoll, aufgrund der zur Verfügung stehenden Bearbeitungsdauer, dem fortgeschrittenen Ausbildungsstand der Studierenden und einer starken Forschungsorientierung. Darüber hinaus werden sie in der Regel zusammen mit Industriepartnern durchgeführt.

Für den Lehrenden sind dies Gelegenheiten, Projektpartnern (und letztlich auch sich selbst) „über die Schulter zu schauen“ und Herangehensweise zu analysieren und mit dem Stand der Forschung abzugleichen.

3.1.3 Schritt 3 – How can these tasks be explicitly modelled?

Aus den so gewonnenen Erkenntnissen werden die in Kap. 2.2 beschriebenen hierarchischen Learning Outcomes durch die Lehrenden spezifiziert, erweitert / überarbeitet und einem internen Review-Prozess unterworfen. Die so entstandenen LOs wiederum prägen dann die Inhalte der Software-Architektur-Ausbildung: Vorlesung (**V**), Lernstandskontrolle (**K**, **T**), Struktur von Übung und Praktikum (**Ü/P**) und nicht zuletzt auch wiederum zukünftige Projekte (**IP, PP, GP**).

3.1.4 Schritt 4 – How will students practice these skills and get feedback?

In den Projekten (**IP, PP, GP**) haben die Studierenden, wie schon in Kap. 2.3 beschrieben, die Möglichkeit, die erworbenen Expertenkompetenzen in Sachen Software-Architektur praktisch einzuüben. Studierende entwickeln und erweitern Module der *Decoding the Disciplines DIGITAL* Softwareplattform. Der Fokus liegt dabei auf einem bestimmten Softwarearchitektur-Aspekt. Es gilt „Entwurfsqualität vor Umfang“. Diese Projekte werden von Professor und Mitarbeitern intensiv betreut. Die Studierenden erhalten so ein entsprechendes Feedback.

Für die Gruppenprojekte (**IP, GP**) wurde dabei im Rahmen mehrerer Abschlussarbeiten [11] [8] ein agiler Entwicklungsprozesses entworfen, der für die speziellen Hochschulbelange optimiert ist[1]. Dieser Prozess ist besonders für unmittelbares Feedback geeignet.

3.1.5 Schritt 5 – What will motivate the students?

Laut Middendorf und Pace wird Motivation beispielsweise durch Transparenz des Lehrkonzepts erzeugt. Das konsequente befolgen des 4P’s-Prinzip „Projects, Peers, Play, Passion“ [13], zusammen mit dem „Eat you own dogfood“ Ansatz (Studierende gestalten ihre eigene Lehrumgebung mit, und wenden dabei konsequent das Erlernte an) führte nach den ersten (zunächst informellen) Feedbackauswertungen von Projekten zu einer wahrnehmbaren Motivationssteigerung gegenüber früheren Lehrprojekten.

[1] Eine große praktische Herausforderung für das agile Vorgehen ist beispielsweise, dass Studierende in der Regel mehrere Lehrveranstaltungen gleichzeitig absolvieren und nur einen Bruchteil ihrer Zeit für das Projekt aufwenden (können). Die für den agilen Ansatz essentiellen Teambildungsprozesse werden dadurch stark behindert. In der Berufspraxis ist dies anders, hier wird in agilen Teams möglichst mit Vollzeitallokation gearbeitet. Das angepasste Vorgehen für die Hochschule versucht diese Situation mit Blockphasen, größeren Sprintlängen, festen Vor-Ort-Tagen mit Anwesenheitspflicht, angepassten Besprechungsregeln etc. zu kompensieren.

3.1.6 Schritt 6 – How well are students mastering these learning tasks?

Die Lernstandskontrolle erfolgt auf ähnlichem Weg wie die Aufdeckung von Lernhindernissen in Schritt 1. Die zurzeit papierbasierten Klausuren (**K**) sind bereits jetzt für die Teilveranstaltung ST1 mit den hierarchischen Learning Outcomes verlinkt. Die Eingabe der Klausurergebnisse in die Plattform muss dabei allerdings von Hand oder durch Import von Excel-basierten Ergebnislisten erfolgen (dies ist geplant, aber noch nicht umgesetzt). Langfristig erscheint ein Umstieg auf Onlineklausuren, beispielsweise per Schnittstelle zu entsprechenden Prüfungssystemen, sinnvoll.

Weitere Rückmeldungen über den praktischen Lernerfolg liefert zukünftig das Selbsttest-Modul (**T**). Auch die Gruppenprojekte (**IP, GP**) werden im Rahmen des oben erwähnten agilen Prozesses in jedem Sprint einer Retrospektive unterworfen und am Ende u.a. nach Architektur-Qualität bewertet. Auch hier ist geplant, die Ergebnisse wieder in das Modul *Auswertung Lernerfolg* zurückzuspeisen.

4 Verstetigung der Lehrinnovation

Wie soll die geplante Lehrinnovation verstetigt werden?

Das hier beschriebene Lehrkonzept mit der damit verbundenen Softwareplattform ist auf eine Entwicklungszeit von mehreren Jahren ausgelegt. Dadurch ist sichergestellt, dass das Projekt durch meine eigene Person und das Team der Fachgruppe Systemgestaltung am Institut für Informatik der TH Köln nachhaltig vorangetrieben wird. Der aktuelle Projektstand ist in Tabelle 2 zusammengefasst.

Tabelle 2: Aktueller Entwicklungsstand des Lehrkonzepts

Bestandteil des Lehrkonzepts	Status zum Zeitpunkt der Antragstellung
Plattformmodul Hierarchische Learning Outcomes	in Entwicklung (laufendes **IP**)
Spezifikation der LOs	ST1: liegt vor [2], ST2: in Arbeit
Plattformmodul *Selbsttest*	Prototyp existiert
Inhalte Selbsttest	Ca. 200 Software-Architekturfragen, allerdings i.d.R. auf Taxonomiestufe 1-2; redaktionelle Bearbeitung steht noch aus
Plattformmodul *Auswertung*	Erster Prototyp existiert, noch keine Daten erfasst
Schnittstellen, Erweiterungen	in Planung (als Abschlussarbeiten)
Integration in die Lehre	Findet wie im Text beschrieben statt, seit 2015

Eine Förderung wie Fellowship für Innovationen in der digitalen Hochschullehre ermöglicht eine schnellere, effektivere Produktifizierung der Software – und damit auch erst den praktischen Einsatz in der Lehre. Bei Robustheit, Tests, Dokumentation und Wartung benötigen reine Studentenprojekten erfahrungsgemäß zusätzliche Unterstützung. Eine Betreuung durch fest angestellte Kräfte ist nötig, um das Konzept auch tatsächlich effektiv im Lehralltag zu verankern.

4.1 Schritt 7 von *Decoding the Disciplines* – How can the resulting knowledge about learning be shared? – Austausch mit anderen Lehrenden

Was versprechen Sie sich vom Austausch mit anderen Fellows des Programms für sich persönlich und für Ihr Projekt?

Das Fellowship erlaubt es mir, mich zu den hier realisierten Ideen mit Fachkolleg*innen auszutauschen, dabei neue Impulse aufzunehmen und diese im Lehrkonzept umzusetzen. Durch den Austausch mit anderen Lehrenden erhoffe ich mir Anregungen und konstruktive Kritik für meine Vorhaben.

Die Erfahrung meines Berufslebens zeigt, dass Treffen mit Fachkollegen immer außerordentlich inspirierend wirken. Gleichzeitig hoffe ich, dass ich selbst auch zu einem solchen Effekt bei anderen Teilnehmern beitragen kann.

Auf den Austausch mit anderen Fellows des Programms freue ich mich ganz besonders, weil hier innovative Köpfe mit hohem Engagement für eine gute Hochschullehre zusammengebracht werden. Es ist damit zu rechnen, dass sich eine lebhafte, fruchtbare Arbeitsatmosphäre einstellt.

Schon jetzt bin ich in mehreren Gremien innerhalb der Hochschule organisatorisch eingebunden und vernetzt, was jeweils auch Auswirkungen auf die Planungen für die Plattform hat:

- Als Studiengangsbeauftragter für den Informatik-Masterstudiengang an der TH Köln und Modulverantwortlicher für mehrere Veranstaltungen gestalte ich die Organisation der Lehre aktiv mit. Weiterhin wirke ich zurzeit an der Neukonzeption mehrerer Bachelor-Studiengänge mit. Dadurch habe ich vielfältigen Einblick in die Herausforderungen der Lehre, auch über meine eigenen Veranstaltungen hinaus.
- Mit den Lehrenden der weiteren Informatik-Kernfächer Mensch-Computer-Interaktion und Datenbanken arbeite ich bei der Praktikumsgestaltung des Informatik-Bachelors eng zusammen.
- Ich bin Mitglied der Fachgruppe Systemgestaltung, die sich mit Methoden der Softwareerstellung auseinandersetzt. ArchiLab wird von den Professoren und Mitarbeitern der Fachgruppe aktiv mitgetragen und unterstützt.
- Ich bin Mitglied des Forschungsschwerpunkts *Digitale Technologien und Soziale Dienste* (DiTeS, www.th-koeln.de/dites), der sich mit der Anwendung von IT-Technologie im sozialen Kontext auseinandersetzt und auch das Thema des Kompetenzerwerbs komplexer Methoden aufgreift.
- 2017 wurde ich für außergewöhnliches Engagement in der Lehre an der TH Köln im Rahmen der 46. Jahrestagung der *dghd* geehrt.

4.2 Übertragung auf andere Fächer und Disziplinen

Auf welche Lehr-Lern-Situationen – auch in anderen Disziplinen – kann die geplante Lehrinnovation übertragen werden?

Zunächst einmal sind Synergien innerhalb der Informatik-Studiengänge der TH Köln möglich. Erste Annäherungen in Form einer gemeinsamen Praktikums-Fallstudie, die über drei Informatikfächer hinweg aus verschiedenen Blickwinkeln bearbeitet wird, finden auf mein Betreiben jetzt, im SoSe 2017, statt.

Darüber hinaus ist – vorausgesetzt das Konzept wird erfolgreich umgesetzt und produktifiziert – ein Einsatz auch in der Informatik-Ausbildung außerhalb der Hochschule sinnvoll. Die Plattform wird, sobald eine hinreichende Testabdeckung vorliegt, in GitHub als Open Source veröffentlicht.

Eine Adaptierung auch auf andere komplexe Fachdisziplinen (z.B. Ingenieurswissenschaften) scheint grundsätzlich möglich. Dies wäre im Rahmen von fachbereichsübergreifenden Projekt- und Abschlussarbeiten zu prüfen.

4.3 Erfolgs- und Risikobewertung in der Erprobung

Wie lassen sich nach Erprobung der Lehrinnovation Erfolg und eventuelle Risiken beurteilen?

Durch die Bindung an das Learning Outcome ist Feedback und Erfolgskontrolle ein essentielles Element des Konzepts, wie in den obigen Kapiteln dargestellt. Darüber hinaus werden werde ich – wie auch schon jetzt – Evaluationsbögen und andere Feedbackformen wie etwa den Teaching Analysis Poll (TAP) [14] nutzen, um Erfolg und Risiken des Einsatzes in der Lehre zu bewerten.