Qualitative Inhaltsanalyse (nach Mayring) von 7 Leitfaden-gestützten Experten-Interviews

Stand: 07.04.2020

Inhaltsverzeichnis

1. Übersichten

1.1 Dokumentenliste[1]

Nr.	Dokument
1	Expertinnen-Interviews\EI00
2	Expertinnen-Interviews\EI01
3	Expertinnen-Interviews\EI02
4	Expertinnen-Interviews\EI03
5	Expertinnen-Interviews\EI04
6	Expertinnen-Interviews\EI05
7	Expertinnen-Interviews\EI06

1.2 Überblick nicht-kognitiver Kompetenzen

Haltung & Einstellung (insgesamt 89)[2]
- Begeisterung & Spaß am Problemlösen/Informatik (je 24)
- Durchhaltevermögen trotz Frust (je 23)
- Lernbereitschaft, Offenheit & Flexibilität (je 20)
- freiwillige Mitarbeit (je 10)
- aktive, regelmäßige Mitarbeit (je 9)
- Freude und Stolz über eigene Ergebnisse (je 3)

Selbstkompetenz (insgesamt 43)
- Vorkenntnisse durch Schule, Ausbildung, etc. (je 23)
- Angemessenes Selbstvertrauen in eigene Fähigkeiten (je 14)
- Kreativität (je 6)

Programmiererfahrung sammeln durch zeitintensives Üben (38)

Sozial-kommunikative Kompetenzen (insgesamt 17)
- Kooperieren und Kollaborieren (je 10)
- Kommunizieren (je 7)

[1] Liste der aus den Interviews resultierenden Transkripte.

[2] In den Klammern werden die Häufigkeiten im Material angezeigt.

1.3 Überblick kognitiver Kompetenzen

Kognitive Prozess-dimension	*Wissens-Dimensionen*			
	Faktenwissen	*Konzeptionelles Wissen*	*Prozedurales Wissen*	*Meta-kognitives Wissen*
Stufe 1 Erinnern	Elementare Programmiersprachliche Konstrukte kennen (3)		Objektorientierte Programmierung kennen (1) Technologische Grundlagen kennen (1)	
Stufe 2 Verstehen		Konzepte der Programmierung in eigenen Worten erklären (2)		
Stufe 3 Anwenden			Werkzeuge zur Software-Entwicklung nutzen (5) Qualitätskriterien auf Quellcode anwenden (3)	
Stufe 4 Analysieren			Gegebene Problemstellung in einzelne Teile zerlegen (11) Compiler & Interpreter-Meldungen lesen können (5) Eigenen Code erklären können (4) Fremden Code lesen können (1) Ausgabe von fremdem Code bestimmen (1)	
Stufe 5 Bewerten			Suchen von Fehlern in Programmen (4) Angemessenes Vorgehen zur Problemlösung auswählen (2) Angemessenheit programmiersprachlicher Lösungen beurteilen (2) Testen von Programmen (1)	Verantwortung übernehmen für Lernerfolg (10) Externe Ressourcen zum Lernen nutzen (9) Selbstreflexion (6)
Stufe 6 Erzeugen			Programmiersprachliche, lauffähige Lösungen für Probleme schreiben (17) Logische Ausdrücke und Operatoren zur Problemlösung benutzen (9) Software(-projekte) strukturiert entwerfen (7) Selbstständig Methoden schreiben (4) Einzelne Klassen schreiben (3) Modellieren von Problemen und Prozessen (3) Vorgegebenen Programmcode erweitern (3) Programm mit mehreren Klassen schreiben (2) In vorgegebene Methodensignatur Code schreiben (2) Selbstständig Code schreiben (2) (Programmier-)Sprachkonstrukte korrekt verwenden (2) Schleifen schreiben (1) Algorithmen mit Hilfe von Entwurfsmustern problemadäquat entwerfen (1) Selbstständig Klasse mit Methode schreiben (1)	Systematik beim Problemlösen entwickeln (26) Selbstständige Organisation des Lernprozesses (18) Abstraktion von Handlungsvorschriften (Rekursion) (10) Transfer (3)

2. nicht-kognitive Kompetenzen

2.1. Haltung & Einstellung

2.1.1. Begeisterung & Spaß am Problemlösen/Informatik

1.

"Unter den Informatik-Studierenden sind immer weniger Nerds. Unsere gestalterisch fokussierten Leute sind sehr selten Nerds. Die interessiert die Informatik nur so ein bisschen, aber nicht so, dass sie da total begeistert sind vom Programmieren. Die Leute, die Computer toll finden, an Computern herumschrauben, Computerspiele spielen und komische T-Shirts tragen, denen kann ich erklären, warum irgendetwas es toll ist, wenn irgendetwas static ist."

[Expertinnen-Interviews\EI00; Position: 68 - 68]

2.

"Das verstehen die und dafür kann ich sie begeistern. Wenn sich Leute nicht dafür begeistern, dann ist Programmieren einfach nur ein Fach, das schwierig ist. Man kann die Leute nur so und so weit antreiben. Wenn sie das nicht selbst begeistert, dann kommt man nur bis zu einem gewissen Punkt mit dem größten möglichen Zwang."

[Expertinnen-Interviews\EI00; Position: 69 - 69]

3.

"Die richtigen Nerds sind normalerweise immer gute Programmierer, weil die setzen sich dann am Abend selbst hin und denen macht es Spaß. Aber das sind nicht die Leute, die ich erreichen will mit meiner Programmierung-Vorlesung, denn die können es schon."

[Expertinnen-Interviews\EI00; Position: 70 - 70]

4.

"Ich will die erreichen, die lieber etwas anderes machen wollen. Von denen haben wir viele. Oder auch Wirtschaftsinformatiker, die sind auch nicht so richtig heiß darauf zu lernen, wie man programmiert. Weil die das in SAP nicht brauchen. Die benötigen jedoch auch die minimalen Kompetenzen und das ist die Herausforderung."

[Expertinnen-Interviews\EI00; Position: 71 - 71]

5.

"Und Bonusaufgaben machen die Leute nur, wenn es sie interessiert. Wenn sie sagen, Programmieren ist toll, also irgendwie macht es Spaß und das kann man nur zu einem gewissen Punkt motivieren. Ich kann versuchen zu zeigen, dass es interessant ist. Aber davon werde ich nicht alle überzeugen."

[Expertinnen-Interviews\EI00; Position: 78 - 78]

6.

"Diejenigen, denen es keinen Spaß macht, müssen es auch verstehen. Und die sind eine Herausforderung."

[Expertinnen-Interviews\EI00; Position: 79 - 79]

7.

"Die Informatiker, die Leute, die schon seit zehn Jahren LAN-Partys machen und irgendwelche Sachen programmieren, die haben sowieso ihre Eins oder Zwei. Aber diejenigen, die keine Lust auf sowas haben, an die muss man sich hauptsächlich richten."

[Expertinnen-Interviews\EI00; Position: 80 - 80]

8.

"Viele wollen nur bestehen, eine 4 reicht aus."

[Expertinnen-Interviews\EI00; Position: 99 - 99]

9.

"B: Programmieren ist in der ganzen Informatik das Wichtigste. Es macht am meisten Spaß und wer das nicht mag, ist meines Erachtens in der Informatik falsch."

[Expertinnen-Interviews\EI01; Position: 1 - 1]

10.

"Es gibt auch Studierende, die entdecken den Spaß am Programmieren. Die lassen sich auch helfen."

[Expertinnen-Interviews\EI01; Position: 14 - 14]

11.

"Generell auch Interessen spielen eine Rolle. Wer Sci-Fi mag, ist eher Programmier-affin, als Leute, die sich vielleicht die Lindenstraße anschauen. Das ist teilweise schon eine Entscheidung, die im frühen Kindesalter gefallen ist. Wenn ich an mich denke, ich fand schon bevor ich in die Schule kam, Science Fiction interessant. Wenn die Entscheidung erst später gefällt wird, dann lockt vielleicht bei vielen noch die Idee, dass man Geld verdienen kann, aber wenn es dann einem keinen Spaß macht, ist halt auch die Frage, wie lange bleibt man in dem Job? Oder wie viel Lust hat man darauf, sich in den Veranstaltungen durchzubeißen?"

[Expertinnen-Interviews\EI01; Position: 28 - 28]

12.

"Es ist die Fähigkeit, Probleme zu lösen, oder sich damit überhaupt beschäftigen zu wollen, sagen wir es mal so."

[Expertinnen-Interviews\EI03; Position: 23 - 23]

13.

"Ich erzähle immer, wie ich angefangen habe, zu studieren. Das erste Mal in einer Analysis Vorlesung dachte ich, das ist toll und hier will ich sein. Das will ich lernen. Hier finde ich es spannend. Und dieses Gefühl beim Studium sollte man haben. Und wer irgendwann diesen Effekt hatte, der schafft es. Ich habe nach drei Wochen von der Analysis nichts mehr verstanden. Ich bin da mit Ach und Krach mit 4 durchgekommen. Aber ich hatte am Anfang dieses Gefühl, das zu wollen. Und das sehe ich bei den Studenten, wenn diese Lust, dieser Initialeffekt irgendwann da war."

[Expertinnen-Interviews\EI03; Position: 39 - 39]

14.

"Wir haben einmal angefangen was für den Girls Day vorbereitet und eigentlich habe ich ein Thema mit denen gemacht, wo ich gedacht habe, dass sich überhaupt niemand anmeldet. Unter dem Motto: Computer aus, wir machen Informatik. Wir haben auf dem Papier mit denen Sachen Informatik durchgespielt. Und da war ein Mädchen, bei der war mir völlig klar, die war so begeistert, die hat das alles mitgemacht. Da war vollkommen klar, die wird irgendwann so etwas machen. Am nächsten Tag schrieb mir die Mutter von dem einen Mädchen zurück, dass das Kind den ganzen Abend der Familie alles erklärt hat, wofür sie sich bedankte. Und da sieht man schon was."

[Expertinnen-Interviews\EI03; Position: 40 - 40]

15.

"Das heißt, irgendwann von Alter 0 bis Alter 18 muss dieser Initialpunkt gewesen sein, Informatik machen zu wollen. Und dann klappt es auch."

[Expertinnen-Interviews\EI03; Position: 41 - 41]

16.

"Und das problematischste, der problematischste Studiengang bei uns sind irgendwie die Wirtschaftsinformatiker."

[Expertinnen-Interviews\EI03; Position: 42 - 42]

17.

"B: Naja, wer kommt zur Wirtschaftsinformatik? Programmierer? Nein, das weiß ich nicht. Ja aber nur so BWL studieren, weiß nicht, da muss ich immer im Anzug rumlaufen. Wirtschaftsinformatik ist so beides nicht. So, ist auch beides nicht in den Köpfen. Es ist beides. Und da gibt es zumindest einen Prozentsatz an Leuten, die das studieren, die es machen, weil sie für keins von beiden brennen. Und dann wird es schwierig. Es gibt die, die mehr für die Wirtschaft brennen und dann ungern programmieren lernen wollen. Die hätten dann doch BWL studieren sollen. Und es gibt die, die für die Informatik brennen. Ich sage denen auch, es ist mehr Informatik, weil Hundekuchen ist Kuchen und kein Hund. Wirtschaftsinformatik ist Informatik und weniger Wirtschaft. Sondern Informatik für die

Wirtschaft, wie der Hundekuchen Kuchen für den Hund ist. Aber es gibt da so die dazwischen, die da sagen, "ich wusste nicht so richtig. Lege ich mich nicht so fest, tut es wahrscheinlich auch nicht so weh", und das ist so die Gefahr bei diesen Zwischen/ oder gemischten Studiengängen."

[Expertinnen-Interviews\EI03; Position: 43 - 43]

18.

"In dem Moment, wenn die Leute loslegen und sich ein Rasperry Pi holen, und irgendwas mit machen. Ja, es klingt schlimm, das sind dann wirklich die Nerds."

[Expertinnen-Interviews\EI03; Position: 48 - 48]

19.

"B: Es gibt die Leute, so wie die rangehen, die wollen auch gar nicht. Da hat es gar keinen Zweck, dass sie sich so quälen. Und vierzig Jahre was machen, an dem sie keinen Spaß haben, wird nicht klappen. Aber es gibt durchaus diesen Bereich an Leuten, die sich reinhängen, viel ausprobieren, es wollen und dann schon wieder durchfallen."

[Expertinnen-Interviews\EI03; Position: 50 - 50]

20.

"B: Man braucht ein bisschen Spaß an kniffligeren Dingen."

[Expertinnen-Interviews\EI06; Position: 14 - 14]

21.

"Wenn man mathematische Sachen gut kann, kann man auch Programmieraufgaben relativ gut."

[Expertinnen-Interviews\EI06; Position: 17 - 17]

22.

"Spaß an abstrakten und komplexen Vorgängen und den Überblick dabei behalten."

[Expertinnen-Interviews\EI06; Position: 18 - 18]

23.

"Es ist förderlich, Spaß dran zu haben, etwas auszuprobieren. Spaß an kniffligen Sachen ist sinnvoll."

[Expertinnen-Interviews\EI06; Position: 60 - 60]

24.

"Programmieren ist ähnlich dem Puzzeln. Man benötigt Geduld und muss umdenken, nicht aufgeben, und Spaß haben, wenn etwas kniffliger wird."

[Expertinnen-Interviews\EI06; Position: 62 - 62]

2.1.2. Durchhaltevermögen trotz Frust

1.

"Aber selbstständig zu versuchen, die Lösung zu erreichen und nicht vorher aufzugeben, das ist das Wichtige."

[Expertinnen-Interviews\EI01; Position: 6 - 6]

2.

"Diese Studierenden wissen vielleicht auch nicht, wie es weitergeht, lassen sich aber helfen, versuchen, weiterzumachen, und schaffen die Aufgaben in der nächsten oder übernächsten Übungsstunde. Die sind auf einem guten Weg."

[Expertinnen-Interviews\EI01; Position: 11 - 11]

3.

"Das Problem ist nur, die Leute müssen für sich erkennen, wenn sie irgendwas mit Informatik machen, dass sie eine gewisse Durchbeißfähigkeit haben müssen."

[Expertinnen-Interviews\EI01; Position: 16 - 16]

4.

"Also wenn ein Problem sofort total langweilt, wenn es nicht sofort gelöst werden kann, dann kann man nie ein guter Programmierer werden - selbst wenn man noch nicht richtig programmieren kann, und mit der Syntax hadert. Aber wenn man ein Problem lösen will, und nicht aufhören will, bevor es gelöst ist, dann kann man ein guter Programmierer werden."

[Expertinnen-Interviews\EI01; Position: 17 - 17]

5.

"B: Mir ist bis jetzt kein Fall bekannt, in dem es die Leute sofort hinkriegen, zu programmieren, ohne es vorher probiert zu haben. Das kann nicht ausgeschlossen werden, aber ich kenne niemanden, der es nicht mal probiert und wo es sofort klappt."

[Expertinnen-Interviews\EI01; Position: 20 - 20]

6.

"Als junger, unerfahrener Mensch wissen Sie das alles nicht und fangen einfach an. Und dann laufen Sie in tausende Fettnäpfchen rein, aber dann hat man das gemacht. Aber wenn Sie es nochmal machen müssen, wissen Sie schon, es gibt das Fettnäpfchen. Bias würde ich auf Englisch sagen."

[Expertinnen-Interviews\EI02; Position: 44 - 44]

7.

"Wie stark ist die Disziplin? Man muss sich zum Teil zwingen im Programmierbereich. Da ändert sich ständig so viel. Allein in der Programmiersprache C++ hat sich in den Standards in den letzten Jahren viel geändert."

[Expertinnen-Interviews\EI02; Position: 47 - 47]

8.

"Eine Zweier-Gruppe hat ungefähr dreißig Sekunden nachgedacht, die Aufgabe in Google eingefügt, keine Antwort bekommen und aufgegeben. Das ist natürlich extrem."

[Expertinnen-Interviews\EI03; Position: 21 - 21]

9.

"Und das ist ein dunkler Tunnel, weil unsere Programmiersprachen abstrakte Sprachen sind und Fehler nicht verzeihen, weil es eindeutig sein muss. Das heißt, du kriegst sprichwörtlich dauernd auf die Finger gehauen, immer wieder und immer wieder, und das Ding sagt immer nur wieder, es ist falsch, falsch, falsch. Aber es sagt in der Regel nicht, warum etwas falsch ist. Und das frustriert."

[Expertinnen-Interviews\EI04; Position: 39 - 39]

10.

"B: Der dunkle Tunnel kann auf jeden Fall zu der Konsequenz führen, dass der Student abbricht."

[Expertinnen-Interviews\EI04; Position: 45 - 45]

11.

"Und wenn man ausschließlich über drei Stunden hört, dass alles falsch ist, muss man frustriert sein."

[Expertinnen-Interviews\EI04; Position: 52 - 52]

12.

"Weil das passiert natürlich beim Programmieren, dass man einen saublöden Fehler macht und den selber nicht findest."

[Expertinnen-Interviews\EI04; Position: 71 - 71]

13.

"Das zweite ist, ich will es nicht Durchsetzungsfähigkeit nennen, sondern Hartnäckigkeit. Wenn man ein Programm hat und das einfach nicht läuft. Das ist unbefriedigend und gräbt an einem. Das ist auch noch sowas erschreckendes, du kannst dich daran nicht vorbeilügen. Der Interpreter sagt, das Programm ist falsch. Und dann musst du gelernt haben, hartnäckig so lange zu verbessern, bis es geht und das kann eben Stunden dauern und das ist Frust."

[Expertinnen-Interviews\EI04; Position: 72 - 72]

14.

"Also Programmierversuche können zu echten Frust werden, wenn ich dieses Erfolgserlebnis nicht habe."

[Expertinnen-Interviews\EI04; Position: 81 - 81]

15.

"Aber nur mit einem sanften Schups, denn man muss sich selbst durchbeißen."

[Expertinnen-Interviews\EI05; Position: 22 - 22]

16.

"Leidensfähigkeit, nein sagen wir eine gewisse Persistenz. Also die Fähigkeit, auch Dinge, die einem nicht sofort zufliegen, sich zu erarbeiten und dranzubleiben. Das ist aber allgemein notwendig, das ist in jedem Fach so."

[Expertinnen-Interviews\EI05; Position: 47 - 47]

17.

"Aber da meine Studis alle Lehrer werden wollen, finde ich es eben auch wichtig, diese Art von Leidensfähigkeit - hier ist es vielleicht tatsächlich für einige Leidensfähigkeit - zu haben- Denn so eine Klasse mit dreißig Schülerinnen und Schülern, kriegt ganz schnell raus, wo die Lücken in den Kompetenzen sind und bohren da gnadenlos rein."

[Expertinnen-Interviews\EI05; Position: 66 - 66]

18.

"Man muss sich durchkämpfen wenn etwas nicht funktioniert."

[Expertinnen-Interviews\EI06; Position: 15 - 15]

19.

"Man muss Ehrgeiz haben um sich durchzukämpfen."

[Expertinnen-Interviews\EI06; Position: 16 - 16]

20.

"B: Die Hartnäckigkeit, sich bei Problemen durchzukämpfen hilft beim Programmieren. Wer gerne Knobelaufgaben oder Mathe-Olympiade macht, hat eher einen Draht dazu und sicherlich gewisse Vorteile."

[Expertinnen-Interviews\EI06; Position: 28 - 28]

21.

"Aber programmieren ist was Anderes als Mathe, wo man Beweise und hochabstrakten Dinge finden muss. Programmieren ist nicht so abstrakt. Manche Sachen sind knifflig, man

muss komplexe Sachen im Kopf behalten und sich drauf konzentrieren, ähnlich wie in der Mathematik darf man nicht gleich aufgeben."

[Expertinnen-Interviews\EI06; Position: 29 - 29]

22.

"Und Durchhaltevermögen braucht man zum Erfolg."

[Expertinnen-Interviews\EI06; Position: 52 - 52]

23.

"Durchhaltevermögen ist wichtig, wenn irgendwas nicht funktioniert, und Geduld."

[Expertinnen-Interviews\EI06; Position: 61 - 61]

2.1.3. Lernbereitschaft, Offenheit & Flexibilität

1.

"Wenn Studierende Formeln sehen oder etwa UML-Diagramme, dann wird es schwierig in ihrer Wahrnehmung. Dann denken sie, das verstehen sie nicht."

[Expertinnen-Interviews\EI00; Position: 45 - 45]

2.

"B: Wenn jemand nur unkooperativ und unwillig ist, weil er schlecht gelaunt ist, heißt nicht, dass er nichts kann. Der macht dann vielleicht doch was."

[Expertinnen-Interviews\EI01; Position: 29 - 29]

3.

"Aber wenn jemand unwillig ist, weil er völlig überfordert ist, das ist dann eher das Problem. Wenn diejenigen ihre Einstellung nicht ändern, werden sie es auch nicht meistern."

[Expertinnen-Interviews\EI01; Position: 30 - 30]

4.

"B: Es ist auch eine Denkweise. Man muss sich darauf einlassen."

[Expertinnen-Interviews\EI01; Position: 65 - 65]

5.

"Man muss sich auch ein bisschen darauf einlassen und überlegen, wie funktioniert eigentlich ein Rechner. Es ist nicht so abstrakt, wie wenn man mit Menschen redet. Das ist ziemlich kontextbehaftet, breit, mehrdeutig, man kann schwafeln. Beim Computer funktioniert das nicht."

[Expertinnen-Interviews\EI01; Position: 67 - 67]

6.

"Die meisten Faktoren sind Charakter. Wie stark ist der Wille, sich neue Dinge beizubringen?"

[Expertinnen-Interviews\EI02; Position: 46 - 46]

7.

"Und dann gibt es jedes Jahr neue Entwicklungsumgebungen, oder alle zwei Jahre. Und dann ist immer die Frage, gehen Sie da mit, oder reicht es irgendwann. Das kostet Kraft."

[Expertinnen-Interviews\EI02; Position: 48 - 48]

8.

"Mir würde auch eine kreative Antwort auf das Rätsel genügen. Die ist zwar falsch, aber sie ist immerhin ein eigener Gedanke."

[Expertinnen-Interviews\EI03; Position: 24 - 24]

9.

"Das einzige, was gilt ist, wer will, kann auch. Dann ist Geschlecht, Herkunft und alles egal. Wer da sitzt um Informatik zu studieren, und das will, schafft das. Und das gilt nicht nur für Programmieren oder Informatik."

[Expertinnen-Interviews\EI03; Position: 38 - 38]

10.

"B: Ein Teil der Studierenden ist schon sehr introvertiert: Informatik-Studierende, Physik-Studierende auch, Mathematiker auch."

[Expertinnen-Interviews\EI04; Position: 11 - 11]

11.

"Also, Abhängigkeitsängste und alles Mögliche, was man so haben kann, warum man eben nicht offen in der Welt ist, und das ist auch so ein typisches, da würde ich jetzt das Vor- schon fast streichen, Vorurteil über Informatiker, dass sie eben nicht so offene Naturen sind, sondern eher so ein bisschen in sich gekehrt."

[Expertinnen-Interviews\EI04; Position: 13 - 13]

12.

"B: Ich habe das auch hinterfragt schon, wie das erklärt werden kann. Die Psychologen sagen, das sind schlechte Erfahrungen, die die Leute gemacht haben. Dann ziehen sie sich nach innen zurück. Also mal abgesehen von einem krankhaften Verhalten wie Autismus, oder Asperger. Aber das ist ja krankhaft, da haben wir die Schwelle überschritten. Aber dazwischen gibt es natürlich viele Stufen von Scheu. Sensibel ist eine andere Kategorie, also von scheu bis nicht offen. Da gibt es schon einige, nicht alle um Gottes Willen. Sonst würde

ich da ja auch gar nicht reinpassen. Aber es gibt einige, die sind sehr verschlossen. Also die sperren sich lieber, und wenn sie lachen müssen, gehen sie lieber in den Keller (lachend)."

[Expertinnen-Interviews\EI04; Position: 14 - 14]

13.

"B: Das sind im Grunde genommen Ängste, die dahinterstehen. Und das ist oft schlechte Erfahrungen also ab der Früh-Kind-Phase. Da wird man dann eher zurückhaltend und eher vorsichtig."

[Expertinnen-Interviews\EI04; Position: 15 - 15]

14.

"B: Diese Gruppe in der Mitte überlegt sich bei jedem Problem einzeln, was zu tun ist. Die sind nicht a priori entschieden, dass sie zuerst 30 Seiten Handbuch lesen oder direkt auszuprobieren ohne etwas zu lesen. Das dauert aber ggf. auch Stunden um zur richtigen Lösung zu gelangen. Also diese Flexibilität ist vielleicht ein grundsätzliches Kennzeichen, diese Flexibilität in der Suche nach Lösungen, dass ich da nicht festgelegt bin und nicht auf ein Kochrezept reagiere, sondern immer wieder im Einzelfall schaue. Und vielleicht zeichnet das wirklich die guten Informatiker aus."

[Expertinnen-Interviews\EI04; Position: 66 - 66]

15.

"B: Also ich würde sagen, auch wieder der schöne Begriff Offenheit. Ich versuche das Problem als Problem zu betrachten, ohne mich im ersten Moment auf eine Richtung oder ein Rezept festzulegen, sondern einfach zu schauen, welche Möglichkeiten es gäbe. Dazu muss man den ganzen Horizont absuchen und sich dann entscheiden, aber begründet."

[Expertinnen-Interviews\EI04; Position: 67 - 67]

16.

"Alle Leute, die das nicht machen, die stoßen früher oder später auf Grundeis. Also weil diese Art von Offenheit muss entweder anerzogen werden, was sehr schwer ist. Also wenn die Persönlichkeit anders gestrickt ist, schlechte Erfahrungen irgendwo gemacht wurden, dann ist das wirklich schwer. Eigentlich will ich sie ja auch nicht erziehen, aber sie müssen herangeführt werden an diese Art des Denkens."

[Expertinnen-Interviews\EI04; Position: 68 - 68]

17.

"Also erfolgreich können nur die in der Mitte sein, die abwägen können, ich gehe mal den Weg, und ich gehe auch mal den Weg. Und die eben auch akzeptieren, dass es für die große Klasse von Problemen keinen Königsweg gibt. Das gibt es nicht."

[Expertinnen-Interviews\EI04; Position: 75 - 75]

18.

"B: Offenheit befördert das Programmieren lernen."

[Expertinnen-Interviews\EI05; Position: 41 - 41]

19.

"Aber nicht jeder ist bereit, zehn Stunden pro Woche zu investieren."

[Expertinnen-Interviews\EI06; Position: 43 - 43]

20.

"Diese Studierenden sind nicht gewohnt, selber zu denken. Eine Bereitschaft zum selbstständigen Nachdenken wird benötigt."

[Expertinnen-Interviews\EI06; Position: 45 - 45]

2.1.4. freiwillige Mitarbeit

1.

"Bei unverbindlichen Anforderungen während des Semesters bleibt die reguläre Mitarbeit aus."

[Expertinnen-Interviews\EI00; Position: 27 - 27]

2.

"Es gibt Studierende, die sind so motiviert, aber diejenigen Studierenden, die freiwillig Aufgaben erledigen, sind diejenigen, die es nicht nötig hätten."

[Expertinnen-Interviews\EI00; Position: 58 - 58]

3.

"Dass die Aufgaben gemacht werden sogar dann, wenn man nicht muss."

[Expertinnen-Interviews\EI00; Position: 74 - 74]

4.

"Wenn Studierende mehr machen, als Sie müssen, oder sich selber Gedanken machen, ist der Übungseffekt ein viel größerer."

[Expertinnen-Interviews\EI00; Position: 75 - 75]

5.

"Wer die Bonusaufgaben macht, obwohl er nicht muss, der lernt auch mehr."

[Expertinnen-Interviews\EI00; Position: 77 - 77]

6.

"Bei den guten Studierenden ist kein Zwang nötig."

[Expertinnen-Interviews\EI00; Position: 94 - 94]

7.

"Es kommen viele Studierende in die Übungen. Aber diejenigen, die es am nötigsten hätten, sind oft nicht da."

[Expertinnen-Interviews\EI00; Position: 112 - 112]

8.

"Freiwillig heißt immer, ich muss es nicht machen und es gibt andere Fächer, die nicht freiwillig sind, also legen die Studierenden den Schwerpunkt dahin. Und da sage ich den Studierenden gleich am Anfang des Studiums, das sie jetzt nicht mehr in der Schule sind, sondern im Studium. Die Studierenden müssen dazu in der Lage sein, sich die Zeit selber einzuteilen. Und ich warne davor, aber es ist ihre eigene Entscheidung."

[Expertinnen-Interviews\EI02; Position: 52 - 52]

9.

"B: Ich muss auf Freiwilligkeit setzen, das ist klar."

[Expertinnen-Interviews\EI05; Position: 60 - 60]

10.

"In freiwilligen Übungen haben zwei von sechzig, siebzig etwas abgegeben."

[Expertinnen-Interviews\EI06; Position: 86 - 86]

2.1.5. aktive, regelmäßige Mitarbeit

1.

"Viele Studierende haben nicht die Einstellung zum regelmäßigen Mitarbeiten."

[Expertinnen-Interviews\EI00; Position: 29 - 29]

2.

"Grundsätzlich heißt das aber nicht unbedingt, dass ich etwas ändern muss. Es kann auch etwas zu schwer gewesen sein, weil sich die Studierenden vorher etwas schon hätten anschauen müssen. Aber das ist ein anderer Fall."

[Expertinnen-Interviews\EI00; Position: 97 - 97]

3.

"In der Programmierung ist das Üben so wichtig, dass man nicht zwei Wochen vor der Klausur zu Lernen anfangen kann."

[Expertinnen-Interviews\EI00; Position: 113 - 113]

4.

"B: Wenn jemand nicht programmieren kann, fällt das an verschiedenen Aspekten auf. Diejenigen kommen später in die Lehrveranstaltung, kommen im Team. Einer zeigt das ganz schnell, und danach verschwinden einzelne ganz schnell wieder. Man erreicht sie nicht mal. Sie wollen nicht über ihren Code reden."

[Expertinnen-Interviews\EI01; Position: 7 - 7]

5.

"Aber die, die es nicht können, versuchen sich um die Übungen zu drücken."

[Expertinnen-Interviews\EI01; Position: 9 - 9]

6.

"Ansonsten sollte jeder seine Übungen haben und möglichst selbst gemacht haben."

[Expertinnen-Interviews\EI01; Position: 38 - 38]

7.

"Effektiver ist es, wenn man selbst motiviert zu Hause sitzt."

[Expertinnen-Interviews\EI06; Position: 55 - 55]

8.

"Durch das kontinuierliche Üben während des Semesters, bleiben sie dran."

[Expertinnen-Interviews\EI06; Position: 88 - 88]

9.

"An der Punktzahl scheitert keiner. Die fehlende regelmäßige Abgabe führt zum Scheitern."

[Expertinnen-Interviews\EI06; Position: 97 - 97]

2.1.6. Freude und Stolz über eigene Ergebnisse

1.

"Leute, die selber programmiert haben reden zwar nicht unbedingt gerne viel darüber, die sind auch manchmal eher schüchtern. Aber man sieht den Studierenden an, dass sie ihren Code vorzeigen können. Die Studierenden wollen ihren Code vorzeigen. Sie sind stolz darauf, dass es auch funktioniert."

[Expertinnen-Interviews\EI01; Position: 8 - 8]

2.

"B: Was positiver ausfällt, ist die Freude, wenn ein Programm funktioniert. Ich glaube, das sollte jeder mal erlebt haben, was das für ein Gefühl das ist, wenn so ein Programm dann funktioniert (lachend). Immer hat der Compiler gesagt, es funktioniert nicht, und dann mit

einem Mal steht genau das da, was ich erwartet habe und was auch richtig ist. Das ist einfach so eine kindliche Freude."

[Expertinnen-Interviews\EI04; Position: 41 - 41]

3.

"Ich will lieber motivieren, dass selber programmiert wird, und dass Studierende stolz auf die Programme sind, die sie geschafft haben. Da will keiner sehen, wie es noch viel schöner und einfacher hätte gemacht werden können. Diese Erkenntnis erfolgt später von alleine."

[Expertinnen-Interviews\EI06; Position: 77 - 77]

2.2. Selbstkompetenz

2.2.1. Vorkenntnisse durch Schule, Ausbildung, etc.

1.

"Es gibt viele Studierende, die können schon gut programmieren. Diejenigen, die eine Lehre gemacht haben, wie etwa Fachinformatiker, können schon sehr gut programmieren."

[Expertinnen-Interviews\EI00; Position: 37 - 37]

2.

"B: Wenige Studierenden können schon gut Programmieren, etwa zehn, zwanzig Prozent haben schon eine Lehre gemacht haben."

[Expertinnen-Interviews\EI00; Position: 38 - 38]

3.

"Ansonsten gibt es nichts, was den Programmiererfolg besonders konditioniert. Wenn einer schon etwas gelernt hat in der Informatik, dann haben die schon die richtige Arbeitseinstellung."

[Expertinnen-Interviews\EI00; Position: 72 - 72]

4.

"Es gibt Studierende mit unterschiedlichen Vorkenntnissen. Manche hatten schon Klassen in der Oberstufe oder Informatik, und andere fangen erst neu an erst. Aber ich würde sagen, letztere sind dann auch noch auf einem guten Weg."

[Expertinnen-Interviews\EI01; Position: 12 - 12]

5.

"Es gibt im Bachelor extrem große Leistungsunterschiede. Es gibt Studierende, die haben schon vorher eine Lehre gemacht. Es gibt Leute, die haben eine eigene Firma und studieren bei uns. Es gibt Leute, die haben schon in ihrer Freizeit vorher programmiert und es gibt viele

Leute, ich sage mal der große Teil, hat im Studium mit dem Programmieren angefangen. Das heißt, die Leistungsunterschiede sind sehr groß."

[Expertinnen-Interviews\EI02; Position: 63 - 63]

6.

"Also die Leistungsunterschiede, grundsätzlich sind sehr groß."

[Expertinnen-Interviews\EI02; Position: 65 - 65]

7.

"Wenn da Leute sitzen, die auch noch total Schwierigkeiten haben, die noch nie programmiert haben und den Lakmus-Test schaffen, ist klar, die kommen durch das Studium, selbst wenn sie am Anfang Schwierigkeiten haben sollten und keine Vorbildung haben."

[Expertinnen-Interviews\EI03; Position: 4 - 4]

8.

"Bei anderen, die schon ein bisschen was können aus der Schule, die dann ohne Nachzudenken das Programm konzipieren und irgendwas Ähnliches machen, fehlt so ein natürlicher Geistesblitz."

[Expertinnen-Interviews\EI03; Position: 5 - 5]

9.

"Manche, woher auch immer, haben die natürlichen Geistesblitze schon."

[Expertinnen-Interviews\EI03; Position: 6 - 6]

10.

"B: Direkt gute Programmierer zu erkennen, ist ein schwieriger Bereich. Da muss man auch aufpassen. Weil nach fünfzehn Jahren oder länger mit Anfängern im Programmieren hat man manchmal das Gefühl nach der ersten Praktikumsstunde, die Studierenden direkt in Schubladen zu stecken. Das stimmt oft, aber man kann auch irren, deshalb muss man da vorsichtig sein, dass man sich da nicht dann so selber vorbelastet."

[Expertinnen-Interviews\EI03; Position: 18 - 18]

11.

"B: Als ich anfing, hatte ich immer die Hoffnung, dass die Studierenden möglichst wenig Vorwissen haben. Weil Vorwissen, schlechtes Vorwissen, oder dogmatisches Vorwissen kontraproduktiv sein kann und dogmatisches Vorwissen kann aus Schulen oder Ausbildung kommen. Da muss erst mal viel erklärt werden, und zusätzlicher Kontext gegeben werden. Wenn Studierende wissen, wie sie mit Index und Länge über die Elemente eines Arrays gehen. Das wenden sie auf alles an, auf Listen, auf Mengen, auf verkettete Listen, und versperren sich gegen Iterator-Konzepte und andere Dinge, weil sie es schon auf eine Art und

Weise können, obwohl es besser geht. Also vor fünfzehn Jahren habe ich gesagt, lieber nicht."

[Expertinnen-Interviews\EI03; Position: 36 - 36]

12.

"Heute sage ich das nicht mehr, weil A, heute hat jeder mehr oder weniger irgendwas gesehen, die bei uns anfangen. Die Schulen in Hessen machen mittlerweile Java, also da ist schon was da. Dann trotzdem, man fängt immer bei null an, egal was mach macht, welches Fach, glaube ich. Also darf das auch nicht mich stören oder hinderlich sein, dass die das schon irgendwie anders gesehen haben und anders gehört haben. So das tut es auch nicht. Und pauschale Aussagen bezüglich Leistungskurs kann ich nicht treffen, dafür habe ich zu wenig Einblick. Da habe ich keine objektiven Zahlen, da habe ich vielleicht mal subjektive, ja und Vorurteile oder sonst was. Das funktioniert nicht, dass ich da was Stichhaltiges sage."

[Expertinnen-Interviews\EI03; Position: 37 - 37]

13.

"Die Dualstudierenden sind natürlich auch besser, weil die von den Firmen schon ausgesucht wurden. Also die haben schon eine Bewerbung durchgenommen und der ganze Studiengang ist zulassungsbeschränkt."

[Expertinnen-Interviews\EI03; Position: 45 - 45]

14.

"Aber durchaus sind das die, die schon parallel im Studium in einer Firma angestellt sind und die arbeiten dort auch einen Tag in der Woche fest, und in der vorlesungsfreien Zeit und zwar nicht als Kopier-Hilfskräfte, sondern als solche, die programmieren und das begünstigt die Sache ungemein."

[Expertinnen-Interviews\EI03; Position: 46 - 46]

15.

"Ich weiß auch noch nicht genau, wie ich das mache. Also gut, ich weiß, wie ich es im Moment mache. Aber es nützt nichts, dass ich nur in dem Licht handle, was ich nachher erreichen will. Ich muss die Leute da abholen, wo sie sind."

[Expertinnen-Interviews\EI04; Position: 46 - 46]

16.

"B: Man muss die Studierenden da abholen, wo sie sind. Und das ist sehr unterschiedlich. Also sagen wir mal so einfach oberflächliche Betrachtung hat bei uns einfach gezeigt, dass das erfolgreiche Bestehen der Eingangsveranstaltung entscheidend vom Vorwissen abhängt. Also Vorwissen ist die wichtigste Größe, die das beeinflusst. Gut, und dass man in der Programmierung jetzt ganz andere, neue Fertigkeiten braucht, macht es schwierig."

[Expertinnen-Interviews\EI04; Position: 85 - 85]

17.

"Und auf der anderen Seite gibt es Leute, die schon recht gut programmieren können, die jetzt nur auf eine andere Sprache umsteigen müssen, aber die Prinzipien verstanden haben und das jetzt nur ergänzend machen. Und das ist ein riesen Spektrum. Das ist die Schwierigkeit für die Lehrenden. Und deshalb können wir auch nur so reagieren, dass wir adaptieren je nachdem, welche Kenntnisstände die Kohorten haben."

[Expertinnen-Interviews\EI04; Position: 86 - 86]

18.

"B: Wir haben hier eine heterogene, diverse Mischung an Studierenden. Man kann vorher nicht wissen, wer es schafft. Nein, das ist aber immer ganz überraschend und spannend."

[Expertinnen-Interviews\EI05; Position: 48 - 48]

19.

"B: Studierende mit Expertise sind schneller im Programme schreiben und stellen weniger Nachfragen."

[Expertinnen-Interviews\EI06; Position: 8 - 8]

20.

"Bei Studierenden mit Expertise funktioniert das Programm funktioniert hinterher, sodass auf eine Eingabe das richtige Ergebnis rauskommt."

[Expertinnen-Interviews\EI06; Position: 9 - 9]

21.

"B: Vorwissen ist förderlich. Zum Beispiel wenn man mit einem programmierbaren Roboter gespielt hat."

[Expertinnen-Interviews\EI06; Position: 56 - 56]

22.

"B: Viele von den Schulfächern sind nutzlos fürs programmieren lernen. Außer Mathe oder Physik, weil man Abstraktion behandelt, könnte helfen. Schule hat mit Programmieren lernen nichts zu tun."

[Expertinnen-Interviews\EI06; Position: 59 - 59]

23.

"Die Studierenden haben sehr unterschiedliche Vorkenntnisse. Es gibt relativ viele, die schon gut programmieren können."

[Expertinnen-Interviews\EI06; Position: 70 - 70]

2.2.2. Angemessenes Selbstvertrauen in eigene Fähigkeiten

1.

"Auch, wenn man von einer Aufgabenstellung überhaupt keine Ahnung hat, muss man erst einmal das Selbstvertrauen in sich selbst haben."

[Expertinnen-Interviews\EI02; Position: 23 - 23]

2.

"Und man muss zuversichtlich sein, dass man, auch wenn ich davon keine Ahnung habe, zum Problem hinkomme, oder zu einer Lösung komme. Und das ist, was die guten Leute im Master im Wesentlichen unterscheidet."

[Expertinnen-Interviews\EI02; Position: 25 - 25]

3.

"Studierende müssen auch lernen, zum Beispiel wirklich selbstbewusst zu werden und zu weiterzumachen, wenn sie etwas gar nicht verstehen."

[Expertinnen-Interviews\EI04; Position: 57 - 57]

4.

"Und sich auch was zuzutrauen."

[Expertinnen-Interviews\EI05; Position: 14 - 14]

5.

"Und indem er selber zuversichtlicher ist, dass er das auch schafft."

[Expertinnen-Interviews\EI05; Position: 16 - 16]

6.

"B: Die Zuversichtlichkeit bei den Studierenden kann man erreichen, indem man ihnen möglichst viele Werkzeuge an die Hand gibt, weil ganz oft ist es einfach tatsächlich in der Informatik so, dass es nicht an den Fähigkeiten, sondern am Zutrauen scheitert."

[Expertinnen-Interviews\EI05; Position: 17 - 17]

7.

"Es gibt eine Untersuchung von der Uni Potsdam, die vergleicht sogenannte Hochleister, also Endrundenteilnehmer am Bundeswettbewerb Informatik, mit ganz normalen Zweitsemester-Studierenden. Das sind die dreißig Jahrgangsbesten, die sich bemühen, die auch an so einem Wettbewerb teilgenommen haben. Und denen haben sie Aufgaben gegeben und sie mit einer Kamera beim Lauten Denken beobachtet. Dabei wurde festgestellt, eigentlich sind die Kompetenzen von den normalen Studierenden genauso groß. Nur die Hochleister trauen sich

zu, die Aufgabe zu lösen, während die anderen so abgeschreckt davon sind, dass sie gar nicht erst anfangen."

[Expertinnen-Interviews\EI05; Position: 19 - 19]

8.

"Es gibt manchmal diesen Durchbruch-Moment. Es gibt Studierende, die drei Mal durch die Prüfung gefallen sind und nach dem 15. Semester völlig frustriert sind. Wenn sie ein Jahr später zum allerletzten Versuch antreten und dann wirklich toll sind, und hart gearbeitet haben, strahlen sie dieses Zutrauen zu sich aus."

[Expertinnen-Interviews\EI05; Position: 23 - 23]

9.

"Und gute Programmierer erkennt man an dem Zutrauen, das auch zu machen."

[Expertinnen-Interviews\EI05; Position: 26 - 26]

10.

"Viele kommen mit der Fehlvorstellung kommen, dass die ziemlich erfolgreich sind beim Programmieren, und dass es ausreicht, wenn Sie für sich kleine und sogar mittlere Aufgabenstellungen lösen können. Die müssen erst mal ins kalte Wasser fallen."

[Expertinnen-Interviews\EI05; Position: 28 - 28]

11.

"B: Es gibt Leute, die ganz hervorragend sind. Die kommen zur mündlichen Prüfung, ohne dass ich sie schon einmal gesehen habe. Das war dann aber auch nicht nötig."

[Expertinnen-Interviews\EI05; Position: 57 - 57]

12.

"Und es gibt die anderen, die fünf Mal in der Vorlesung sitzen und die Prüfung nicht bestehen."

[Expertinnen-Interviews\EI05; Position: 58 - 58]

13.

"Es gibt eine Gruppe von Studenten, die sehr viel Zeit reinsteckt, aber permanent nur Hilfe für jede Zeile möchte. Diese Studierenden möchten für jede Zeile Feedback zur Richtigkeit bekommen, und was als nächstes zu tun ist."

[Expertinnen-Interviews\EI06; Position: 44 - 44]

14.

"Das Vertrauen in einem selbst, sprich ein gewisses Selbstvertrauen, dass man das hinkriegen wird hilft, das auch hinzukriegen."

[Expertinnen-Interviews\EI06; Position: 58 - 58]

2.2.3. Kreativität

1.

"Und dann gibt es noch Punkte, da braucht man Kreativität. Und diese Kreativität hat einmal was mit dieser Intuition zu tun und sehr viel mit logischem Denkvermögen."

[Expertinnen-Interviews\EI02; Position: 6 - 6]

2.

"B: Also das Problem ist eine Mischung von allem. Es ist die Kreativität definitiv. Da fehlt es an Kreativität."

[Expertinnen-Interviews\EI02; Position: 16 - 16]

3.

"Das Modellieren ist ein kreativer Prozess."

[Expertinnen-Interviews\EI05; Position: 7 - 7]

4.

"Aber da lege ich gar nicht so viel Wert drauf, dass man so einen bestimmten formalen Weg nimmt. Da sind die meisten guten Studierenden zumindest selbst sehr kreativ, dann einen eigenen Weg zu finden."

[Expertinnen-Interviews\EI05; Position: 12 - 12]

5.

"B: Also Kreativität ist jetzt keine Kompetenz, die ich probiere jemandem beizubringen, sondern eher die eigene Kreativität mit bestimmten Werkzeugen auszulegen."

[Expertinnen-Interviews\EI05; Position: 13 - 13]

6.

"Das ist sozusagen die Komponente von den Hochleistern und die Komponente auf der anderen Seite ist tatsächlich die Einsicht, dass das formale nur die oberen fünf Prozent sind, oder zehn Prozent sind, und dass es tatsächlich ganz viel Kreativität ist. Die Kreativität ist das Entscheidende."

[Expertinnen-Interviews\EI05; Position: 31 - 31]

2.3. Programmiererfahrung sammeln durch zeitintensives Üben

1.

"B: Die ganze Informatik und die Programmierung insbesondere lebt davon, dass man selbst aktiv die Übungen macht."

[Expertinnen-Interviews\EI00; Position: 73 - 73]

2.

"Mit Erfahrung von 20 Jahren, oder seit 30 Jahren, fällt das leichter."

[Expertinnen-Interviews\EI00; Position: 82 - 82]

3.

"Da muss man viel geübt haben, und deswegen ist Programmierung schwierig. Ich würde sagen, Programmierung ist anspruchsvoller als die meisten anderen Sachen, weil man diese Art von Denken nirgendwo beigebracht bekommt."

[Expertinnen-Interviews\EI00; Position: 87 - 87]

4.

"B: Man muss die jeweiligen Probleme oft genug durcharbeiten, um von der Erfahrung bei einem Problem darauf rückschließen zu können und diese Erfahrung bei der Lösung des neuen Problems nutzen zu können als Lösungsansatz."

[Expertinnen-Interviews\EI00; Position: 88 - 88]

5.

"Programmieren ist eine Fertigkeit, das ist ein Handwerk. Das kann man lernen, wenn man es auf kleine Pakete herunterbricht und jedes einzelne kleine Paket genügend übt. Der Schlüssel zum Programmieren, im Vergleich vielleicht zu anderen Vorlesungen, ist man muss es wirklich ausreichend üben muss."

[Expertinnen-Interviews\EI00; Position: 89 - 89]

6.

"Man muss nicht viel verstehen. Die einzelnen Konstrukte der Programmiersprachen sind nicht schwer. Aber deren Nutzung zur Lösung von Problemen ist schwer und das muss man üben."

[Expertinnen-Interviews\EI00; Position: 90 - 90]

7.

"Und dann lernt man, seinen Code zusammenzusetzen zu größeren Einheiten. Aber auch da ist der Schlüssel die Übung. Da gibt es nicht viel zu verstehen, das muss man einfach kennen, im Sinne von oft gemacht haben. Deswegen ist die Programmierung so Übungs-intensiv. Da gibt es nichts zu verstehen. Die Studierenden die sich die Vorlesung anhören, aber nicht üben, werden in der Klausur keine Aufgaben lösen können. Das liegt immer nur am Üben, das ist vielleicht in anderen Fächern nicht ganz so extrem."

[Expertinnen-Interviews\EI00; Position: 92 - 92]

8.

"Vielleicht gibt es noch einen dritten Kern Erfahrung. Mit Erfahrung können Sie die Intuition kompensieren."

[Expertinnen-Interviews\EI02; Position: 3 - 3]

9.

"Und eine Möglichkeit, Intuition zu kompensieren, ist Erfahrung. Und damit das mit der Erfahrung klappt, gibt es systematische Ansätze, wie man zu einer Lösung kommen kann. Die klappen dann bis zu einem gewissen Punkt."

[Expertinnen-Interviews\EI02; Position: 5 - 5]

10.

"Da fehlt es aber auch an Erfahrung."

[Expertinnen-Interviews\EI02; Position: 17 - 17]

11.

"B: Vorerfahrung halte ich mit für das Wichtigste. Auf der einen Seite ist das frustrierend am Programmieren. Jedes Mal ist was anders als vorher. Das heißt, es ist ein ständiger Lernprozess. Das Schöne ist, dass es nie langweilig wird."

[Expertinnen-Interviews\EI02; Position: 42 - 42]

12.

"B: Also Alter und Erfahrung hat definitiv was mit zu tun."

[Expertinnen-Interviews\EI02; Position: 45 - 45]

13.

"Und das ist dann Erfahrung, teilweise Kreativität, logisches Denkvermögen, aber viel Erfahrung. Allein zu wissen, dass es sowas gibt, ist schon mal viel Wert."

[Expertinnen-Interviews\EI02; Position: 59 - 59]

14.

"Andere müssen das erst erwerben durch Erfahrung."

[Expertinnen-Interviews\EI03; Position: 7 - 7]

15.

"B: Programmieren lernen ist ein bisschen wie Schwimmen lernen, und dazu ins Wasser zu gehen. Niemand lernt schwimmen, indem er in eine Vorlesung geht, es erklärt bekommt, indem er ein Buch liest, wo genau die Bewegungen erklärt werden. Wenn ich ihn ins Wasser

schmeiße, geht er unter. Programmieren ist genauso. Das ist vielfach ein Handwerk. Und das ist auch beim Tischler, viele, viele Regale fallen um und sind schief, bis man es kann. Und das gilt auch für das Programmieren, auch wenn das ja eine abstrakte Sache ist. Aber trotzdem, weil wir es nicht mit dem Körper machen, sondern mit dem Kopf. Deshalb ist es natürlich schon was Anderes. Oder anders gesagt, warum kann man Schwimmen nicht aus dem Buch lernen? Weil es auch im Kopf ist. Oder ein Instrument lernen, das ist zwar alles was Körperliches, aber es geht darum, zu trainieren, dass bestimmte Muster im Kopf automatisiert werden, dass man immer wieder gleiche Dinge erkennt, intuitiv sagt man dann aber, das ist erübt. Und das ist beim Programmieren dasselbe, deshalb führt einfach auch halt kein Weg in der Programmierausbildung an viel Machen vorbei."

[Expertinnen-Interviews\EI03; Position: 8 - 8]

16.

"Aber ich glaube zentral ist erst mal, dass man diese Praxis hat, dass man selber Programme schreiben kann."

[Expertinnen-Interviews\EI03; Position: 15 - 15]

17.

"B: Praxis fördert den Zugewinn an Programmierkompetenz. Wir haben jetzt seit einem Jahr die Dualstudierenden. Die sind signifikant besser als die anderen. Liegt daran, dass die in einer Firma programmieren. Sie sehen einfach, das ist mein Berufsbild, dass ich diese Fähigkeit können muss. Die verbessern sich."

[Expertinnen-Interviews\EI03; Position: 44 - 44]

18.

"Es muss aktiv verfügbar sein und das brauch viel Übung. Und das verstehen nicht alle Studierenden auf Anhieb, dass man durch diesen dunklen Tunnel gehen muss und all diese Aufgaben selber programmieren muss. Wenn sie es lesen ist es kein Problem und sie denken, sie haben es verstanden. Aber genau das reicht nicht."

[Expertinnen-Interviews\EI04; Position: 29 - 29]

19.

"Und das ist ganz viel Gewöhnung."

[Expertinnen-Interviews\EI04; Position: 31 - 31]

20.

"B: Noch nie hat jemand in der Vorlesung programmieren gelernt. Er lernt es nur durch selber machen, dadurch, dass er es wiederholt, schwieriger werdende Probleme umzusetzen in die Programmiersprache."

[Expertinnen-Interviews\EI04; Position: 38 - 38]

21.

"Also dieser Frust kommt dazu, und der Zeitverbraucht. Man hat dann das Gefühl, dass man viel zu viel Zeit da investiert, und das sind einfach Schwierigkeiten, mit denen alle Unerfahrenen konfrontiert sind."

[Expertinnen-Interviews\EI04; Position: 40 - 40]

22.

"Das Problem ist, programmieren lernen erfordert sehr viel Zeit."

[Expertinnen-Interviews\EI04; Position: 47 - 47]

23.

"Und das können die Leute am Anfang nicht einschätzen und verbrauchen dann ihre Zeit an der falschen Stelle. So und Zeit ist natürlich knapp und dann haben sie Schwierigkeiten."

[Expertinnen-Interviews\EI04; Position: 65 - 65]

24.

"B: Ich kenne keinen anderen Weg, als viel Zeit zu investieren. Niemand lernt in meiner Vorlesung programmieren. Er lernt es nur dadurch, dass er es selber macht. Und das kostet Zeit."

[Expertinnen-Interviews\EI04; Position: 69 - 69]

25.

"Die meisten Leute werden dann ein anderes Fach abwählen müssen, weil man kann nicht drei solche Fächer nebeneinander mal gerade ziehen. Wenn ich in einem Fach echte Defizite habe, das ist so."

[Expertinnen-Interviews\EI04; Position: 99 - 99]

26.

"Man kann das nur lernen durch Wiederholung und Betrachten aus vielen Seiten und so weiter und so weiter."

[Expertinnen-Interviews\EI04; Position: 114 - 114]

27.

"B: Dass sich jemand beim Programmieren verbessert, sieht man daran, dass jemand die Aufgabenstellung schneller und effizienter umsetzen kann."

[Expertinnen-Interviews\EI05; Position: 15 - 15]

28.

"Und natürlich gehört da dazu, sich damit zu beschäftigen und auch lange damit zu beschäftigen. Es gibt Überflieger, die gucken einmal drauf können alles. Und die anderen müssen sich das hart erarbeiten."

[Expertinnen-Interviews\EI05; Position: 24 - 24]

29.

"B: Für die einen ist dieser Erkenntnisprozess schwieriger und für die anderen weniger. Also es ist was Menschliches. Es ist auch schwierig, den zu analysieren, aber den muss jeder für sich selber durchlaufen und man kann dann letztlich an den Ergebnissen schauen, ob man den Eindruck hat, dass sie erfolgreich waren oder nicht. Aber das ist jetzt nichts, wo ich sagen würde das lässt sich direkt messen."

[Expertinnen-Interviews\EI05; Position: 33 - 33]

30.

"Aber letztlich ist das natürlich eine Sache, die alle selber machen müssen. Das kriegt man nicht durch Vorlesung, sondern man kriegt es beim Durchmachen."

[Expertinnen-Interviews\EI05; Position: 36 - 36]

31.

"Programmieren ist Übungssache. Als Anfänger mit ganz viel Zeit kann auf eine einfache Frage ein Programm geschrieben werden. Mit Übung hat, kann man schnell Programme schreiben."

[Expertinnen-Interviews\EI06; Position: 10 - 10]

32.

"B: Übung erreicht man, indem man Zeit reinsteckt."

[Expertinnen-Interviews\EI06; Position: 11 - 11]

33.

"Mit mehr oder weniger viel Übung kann man auf jeden Fall programmieren lernen."

[Expertinnen-Interviews\EI06; Position: 13 - 13]

34.

"Einfache Programme von ein paar Zeilen zu schreiben kriegt jeder hin mit entsprechendem Zeitaufwand."

[Expertinnen-Interviews\EI06; Position: 26 - 26]

35.

"B: Bestimmte Themen, die einfach sind, gibt es nicht. Mit mehr Übung können früher oder später Programme geschrieben werden."

[Expertinnen-Interviews\EI06; Position: 39 - 39]

36.

"B: Programmieren ist nicht das Fach, weswegen das Studium scheitert. Trotzdem fallen relativ viele durch, die noch nicht genug Übung haben."

[Expertinnen-Interviews\EI06; Position: 41 - 41]

37.

"Programmieren ist für jeden mit mehr oder weniger viel Aufwand machbar."

[Expertinnen-Interviews\EI06; Position: 42 - 42]

38.

"Programmieren ist eine Übungssache und jeder muss üben."

[Expertinnen-Interviews\EI06; Position: 72 - 72]

2.4. Sozial-kommunikative Kompetenzen

2.4.1. Kooperieren und Kollaborieren

1.

"Die Studierenden sind ja in einer Gruppe und es soll passieren, dass sie sich in der Form untereinander helfen."

[Expertinnen-Interviews\EI03; Position: 79 - 79]

2.

"Übrigens, Programmieren ist heute Teamarbeit. Das ist nicht Programmieren im Kleinen, wenn man ein Programm schreibt, um ein paar Labordaten auszuwerten. Das hat fast nichts mit Programmieren zu tun. Sondern man überlegt sich, wenn man so ein System wie Google machen will, wie man es aufgestellt. Und das kriege ich nur im Team hin mit sehr vielen Skills und Fähigkeiten und da müssen die Studierenden herangeführt werden. Auch wenn sie das nicht immer sofort mögen."

[Expertinnen-Interviews\EI04; Position: 3 - 3]

3.

"Oder, dass sie Probleme haben, einen Anzug anzuziehen (lacht)."

[Expertinnen-Interviews\EI04; Position: 8 - 8]

4.

"Es gibt schon ein paar (lachend) Eigentümlichkeiten bei Informatikern, wo man den Leuten helfen muss, dass sie sich das entweder nicht angewöhnen, oder sie sich das abgewöhnen. Weil sonst müssen sie damit leben, dass die Betriebswirte sie einfach ausnutzen."

[Expertinnen-Interviews\EI04; Position: 10 - 10]

5.

"Und die mögen keine Gruppenarbeit. Also Sie sehen das ein, intellektuell können sie es verarbeiten. Also bei mir ist Teamarbeit, im Zweierteam, schon in der Einführungsveranstaltung Pflicht. Und es wird immer gefragt, ob man das alleine machen kann. Immer. Und das zeugt dann davon, dass es eben nicht krankhafter Autismus ist. Aber das ist eine Tendenz in die Richtung. Die Studierenden machen das lieber alleine, weil sonst sind sie abhängig von anderen."

[Expertinnen-Interviews\EI04; Position: 12 - 12]

6.

"B: Das tritt gehäuft in der Informatik auf. Aber das liegt nicht an der Informatik, sondern diese Art von Persönlichkeiten glauben in der Informatik in Ruhe gelassen zu werden. Ganz einfach, weil sie es in der Schule zum Beispiel nie erleben, diese Art von Gruppenarbeiten. Das empfinden sie eher als Zwang. Also bei mir muss man sogar dann den Partner oder die Partnerin wechseln. Was ja für die Leute eine Katastrophe ist."

[Expertinnen-Interviews\EI04; Position: 16 - 16]

7.

"Aber aus meiner Sicht gehört das dazu, weil ich kann nichts mit einem Einzelkämpfer anfangen in einem professionellen Team. Und ich kann nichts damit anfangen, wenn jemand nur mit einer anderen Person zusammenarbeiten kann. Also er muss im Team arbeiten können. Alle großen Projekte sind Teamarbeit, das geht gar nicht anders. Darum muss ich sie quälen (lachend)."

[Expertinnen-Interviews\EI04; Position: 17 - 17]

8.

"Aber Gruppenarbeit mögen offensichtlich nicht alle und ich kann auch nicht jedermanns Händchen halten. Aber ich lasse es nicht zu, dass das jemand dann alleine macht. Also das sind ganz besondere Ausnahmen, wo ich das mal zulasse. Aber es gehört dazu, sie müssen da durch."

[Expertinnen-Interviews\EI04; Position: 19 - 19]

9.

"Die Studierenden müssen erkennen, dass man größere Aufgabenstellungen nur im Team bewältigen kann, und dass diese insbesondere über die Zeit eine weitere Komponente haben."

[Expertinnen-Interviews\EI05; Position: 29 - 29]

10.

"Und entscheidend ist die Erkenntnis, dass es eben mehr um Menschen als um Computer geht."

[Expertinnen-Interviews\EI05; Position: 32 - 32]

2.4.2. Kommunizieren

1.

"Die Studierenden trauen sich auch gar nicht bei mir was zu sagen. Wenn da ein Tutor ist, ist es einfacher, dabei sagt der auch nichts anderes als ich. Da bräuchte man keine Angst haben, wir sind ja da zum Lernen."

[Expertinnen-Interviews\EI01; Position: 45 - 45]

2.

"Wichtig ist auch, in Gruppen über Aufgaben nachzudenken."

[Expertinnen-Interviews\EI03; Position: 20 - 20]

3.

"Wenn ich auf das Team anspreche, geht es um Skills wie Teamfähigkeit. Beginnt mit einfachen Sachen wie: Zuhören können, aktiv zuhören können, niemanden unterbrechen, Kritik äußern zu können, ohne verletzend zu sein, Kritik empfangen zu können, ohne verletzt zu sein (sachliche Kritik). Wenn es nötig ist, muss man sich auch mal aufregen zu können, ohne dass es einen persönlich wirklich möglichst belastet, damit der Gegenüber das mitkriegt. Weil das können viele Leute nicht."

[Expertinnen-Interviews\EI04; Position: 5 - 5]

4.

"Oder so andere kleine Sachen, wo ich erlebt habe, dass Informatiker daran versagen. Sie stellen ihr Programm vor, und erklären immer, was nicht geht. Sie erklären nicht das viele, was alles geht, sondern sie erklären die Probleme, die sie gerade haben."

[Expertinnen-Interviews\EI04; Position: 6 - 6]

5.

"Die Probleme interessieren den Zuhörer überhaupt nicht und gehen ihn eigentlich auch gar nichts an. Das sind klassische Kommunikationsfehler, die Informatiker machen."

[Expertinnen-Interviews\EI04; Position: 7 - 7]

6.

"Oder dass, wenn sie eine Zustimmung ausdrücken wollen, sie nur brummen (lachend)."

[Expertinnen-Interviews\EI04; Position: 9 - 9]

7.

"Diejenigen, die nichts abgeben bleiben mit ihren Vorstellungen alleine."

[Expertinnen-Interviews\EI05; Position: 56 - 56]

3. Wissensdimensionen[3]

3.1. Faktenwissen

1.

"B: Es gibt ein paar Auffälligkeiten beim Lernprozess. Auffällig ist zum Beispiel, dass es sehr viele Spracheigentümlichkeiten zu lernen gilt. Also, ich will nicht sagen Auswendiglernen, aber für einen Anfänger ist es natürlich viel, was er da alles sich behalten soll."

[Expertinnen-Interviews\EI04; Position: 27 - 27]

2.

"Es sind viele Sachen gar nicht so besonders schwer. Darum ist es wichtig, dass wir eine Programmiersprache haben, die sehr strikt ist und möglichst wenige Ausnahmen hat. Weil Ausnahmen müssen wirklich einzeln gelernt werden. Die anderen Ausdrücke sind unter den vielen Programmiersprachen sehr einheitlich. Da lernt man da nicht nur eine Programmiersprache, sondern man lernt eine ganze Gruppe von Programmiersprachen und das ist gut."

[Expertinnen-Interviews\EI04; Position: 32 - 32]

3.

"Aber sobald man Ausnahmen hat, geht gar nichts anderes, als das auswendig zu lernen. Und das ist am Anfang schlecht, weil ich dann noch zu den ganzen Begrifflichkeiten zusätzlich noch die Ausnahmen lernen muss."

[Expertinnen-Interviews\EI04; Position: 33 - 33]

3.2. Konzeptionelles Wissen

1.

"B: Meine Klausuren sind reine Programmierklausuren mit ein paar Theorie-Fragen. Letztere wurden die letzten Jahre reduziert. Oft ist es so, dass Studierende grundsätzlich eigentlich verstanden haben und erklären in ihren eigenen Worten, aber es fehlt ein Stichwort. Dann gibt man den Punkt nicht."

[Expertinnen-Interviews\EI02; Position: 36 - 36]

2.

"Wichtig ist mir diese fünfte Aufgabe, in der etwas erklärt werden muss. Ich finde das wichtig, dass ein Informatiker gezwungen wird, was in Deutsch aufzuschreiben. Das ist die

[3] Unter den nachfolgenden Überschriften werden alle Segmente der gleichnamigen Kodierung aufgelistet. Da kognitive Kompetenzen doppelt kodiert wurden (einmal pro Wissensdimension und einmal pro kognitiver Prozessdimension) finden sich alle kodierten Segmente der kognitiven Dimensionen genau zwei Mal im Sample.

komplexeste Sache zu korrigieren. Das muss man lesen, Textverständnis aufbringen und abgleichen mit dem Text, den man selber geschrieben hätte."

[Expertinnen-Interviews\EI03; Position: 91 - 91]

3.3. Prozedurales Wissen

1.

"Lernende sollen anhand einer vorgegebenen Methode mit vorgegebener Signatur, ein Code-Snippet schreiben."

[Expertinnen-Interviews\EI00; Position: 1 - 1]

2.

"Bei vorgegebenem Problem schreiben die Studierenden selbstständig eine Klasse mit Methode"

[Expertinnen-Interviews\EI00; Position: 2 - 2]

3.

"Lernende sollen Programme verstehen, indem sie erkennen, was ein fremder Code ausgibt."

[Expertinnen-Interviews\EI00; Position: 3 - 3]

4.

"Wenn jemand programmieren kann, versteht er zwar Programme, kann aber auch selber Programme schreiben."

[Expertinnen-Interviews\EI00; Position: 4 - 4]

5.

"Auf eine genaue Vorgabe hin ergänzen die Studierenden Code in einer Methode, damit diese eine bestimmte Sache ausführt."

[Expertinnen-Interviews\EI00; Position: 5 - 5]

6.

"Für eine gegebene Problembeschreibung sind die Studierenden in der Lage, eine Klasse zu schreiben."

[Expertinnen-Interviews\EI00; Position: 6 - 6]

7.

"Alternativ können die Studierenden ein kleines Projekt mit mehreren Klassen schreiben."

[Expertinnen-Interviews\EI00; Position: 7 - 7]

8.

"Das ganze Projekt muss richtig partitioniert werden, es müssen Files angelegt werden. Es muss bei einem Projekt überlegt werden, was in welcher Klasse vorkommt, um das Programm laufen zu lassen."

[Expertinnen-Interviews\EI00; Position: 9 - 9]

9.

"B: Die Kompetenz, selber Programme zu schreiben, soll gefördert werden."

[Expertinnen-Interviews\EI00; Position: 10 - 10]

10.

"Dabei gebe ich ein Problem und die Studierenden schreiben zumindest eine Klasse dazu."

[Expertinnen-Interviews\EI00; Position: 11 - 11]

11.

"Die Studierenden müssen selten Klassen schreiben."

[Expertinnen-Interviews\EI00; Position: 13 - 13]

12.

"Es ist ausreichend, wenn die Studierenden die richtige Methode schreiben, die eine Array-Liste erzeugt und zurückgibt. Dazu muss die Methodendefinition erfolgen mit korrekten Parametern und Rückgabewert."

[Expertinnen-Interviews\EI00; Position: 14 - 14]

13.

"Codeverständnis wird stark abgeprüft."

[Expertinnen-Interviews\EI00; Position: 15 - 15]

14.

"Die Studierenden sollen selbst Code erzeugen mit offenen Vorgaben bzw. Aufgabenstellung."

[Expertinnen-Interviews\EI00; Position: 16 - 16]

15.

"Die Studierende müssen selbst entscheiden, wie sie genau vorgehen, um ein Problem zu lösen. Der Lösungsweg ist nicht entscheidend, solange das Problem gelöst wird."

[Expertinnen-Interviews\EI00; Position: 17 - 17]

16.

"Aus dem vorgegebenen Problem entwickeln die Studierenden die Programm-Struktur. Das wäre die höchste Form des Programmierens."

[Expertinnen-Interviews\EI00; Position: 18 - 18]

17.

"Die Studierenden sollen bei vorgegebener Methode mit Header Code schreiben."

[Expertinnen-Interviews\EI00; Position: 19 - 19]

18.

"Die Studierenden sollen eine Methodensignatur schreiben und diese mit Code befüllen."

[Expertinnen-Interviews\EI00; Position: 20 - 20]

19.

"Es ist das Ziel, dass Studierende Code selber produzieren, ohne dass alles im Detail vorgegeben wird."

[Expertinnen-Interviews\EI00; Position: 21 - 21]

20.

"Im Sinne der Prüfung ist es schon in Ordnung, wenn die Studierenden die richtige Methodensignatur schreiben können und die Methode richtig mit Code ausfüllen."

[Expertinnen-Interviews\EI00; Position: 23 - 23]

21.

"Das selbstständige, aktive Schreiben von Code ohne genaue Mustervorgaben fällt den Studierenden schwer."

[Expertinnen-Interviews\EI00; Position: 24 - 24]

22.

"B: Ich will nicht, dass Studierende alle zwölf Methoden der Reflection-API aufsagen können. Das kann ich nachschauen. Aber die Studierenden müssen sie benutzen können, darum geht es."

[Expertinnen-Interviews\EI00; Position: 64 - 64]

23.

"Aber wenn man das Programmieren nicht prüft, bildet man die Leute nicht richtig aus. Die Leute müssen selber vom Problem ausgehend ein Programm schreiben. In der Praxis ist auch niemand, der die Main-Methode vorschreibt."

[Expertinnen-Interviews\EI00; Position: 66 - 66]

24.

"B: Logisches Denken ist schwierig. Bei der Aufgabe, alle Elemente eines Arrays um eine Position nach links zu verrutschen, entstehen zahlreiche Arbeitsschritte und viele einzelne Operationen."

[Expertinnen-Interviews\EI00; Position: 81 - 81]

25.

"Um vom Problem zum Programm zu gelangen muss ein in Worten beschriebenes abstraktes Problem auf präzise Einzelanweisungen heruntergebrochen werden."

[Expertinnen-Interviews\EI00; Position: 83 - 83]

26.

"Das Analysieren von Problemen und Herunterbrechen auf präzise Einzelanweisungen ist ein NP-hartes Problem und schwierig. Und das ist eine Art von Denken, die man auch normalerweise nicht lernt."

[Expertinnen-Interviews\EI00; Position: 84 - 84]

27.

"Auch Fehler suchen lernt man nicht. Was macht man, wenn das Programm nicht läuft? Ich würde sagen benutze Print-Anweisungen um zu sehen, bis wohin es läuft und dann schaue nach, ob die Variablen alle stimmen."

[Expertinnen-Interviews\EI00; Position: 85 - 85]

28.

"Aber das Suchen von Fehlern im Programm ist etwas, was man nicht a priori weiß, das muss man einmal gesagt bekommen oder einmal gemacht haben."

[Expertinnen-Interviews\EI00; Position: 86 - 86]

29.

"Sichtbar wird es, wenn Studierende bei der Abgabe Code vorführen sollen, den sie gemacht haben. Wenn sie nicht wissen, was die einzelnen Zeilen machen, ist das ein Problem."

[Expertinnen-Interviews\EI01; Position: 3 - 3]

30.

"Wer programmieren kann, da sieht man den Prozess im Praktikum. Die Studierenden sind in der Lage, während der Übungsstunde ein funktionierendes Programm fertigzustellen."

[Expertinnen-Interviews\EI01; Position: 4 - 4]

31.

"Problematisch sind diejenigen, die es gar nicht erst versuchen. Da weiß ich auch gar nicht, wie diese Studierenden erreicht werden können. Wer nicht weiß, wie man eine Entwicklungsumgebung aufmacht, oder kompiliert, keine Syntaxfehler sieht und beheben kann, und es nicht ausprobiert, kann kein guter Programmierer werden. Das beobachte ich in den älteren Semestern. Im ersten Semester kann das sicherlich noch passieren, dass man die Fehler nicht versteht. Aber wenn das im dritten, vierten oder fünften Semester auftritt, ist das ein echtes Problem. Diese Probleme treten auf bei Studierenden, die irgendwas mit Medien machen wollten, aber nicht programmieren wollten."

[Expertinnen-Interviews\EI01; Position: 13 - 13]

32.

"B: Für viele, die neu anfangen, ist das Formale eine Schwierigkeit. Die Studierenden haben vielleicht eine grobe Idee, wie sie Probleme lösen. In der Vorlesung wurde die Lösung in natürlicher Sprache gemeinsam an der Tafel besprochen. Dabei wurde ungefähr verstanden, worum es geht und was die Idee ist. Aber wenn das dann formal logisch in eine Programmiersprache gefasst werden soll, ist das sehr schwer."

[Expertinnen-Interviews\EI01; Position: 18 - 18]

33.

"Studierende, die wenig mit Mathematik zu tun hatten, haben es schwerer, zum Beispiel bei Bedingungen und deren Reihenfolge. Viele denken, die Reihenfolge ist völlig gleichgültig - ist sie natürlich nicht. Man muss das aber formaler ausdrücken zu können. An der Stelle würde es helfen, wenn die Mathematik nicht so stark vernachlässigt wird, dann könnte die Logik, die hinter Programmiersprachen und etwa Bedingungen formuliert wird, durchschaubarer werden für Studierende."

[Expertinnen-Interviews\EI01; Position: 19 - 19]

34.

"Für die Aufgaben haben die Studierenden erstmal den Code verstehen müssen und das Feature an der richtigen Stelle einbauen müssen."

[Expertinnen-Interviews\EI01; Position: 41 - 41]

35.

"Ganz viele scheitern schon daran richtig zu kompilieren. Da muss man von Anfang an helfen."

[Expertinnen-Interviews\EI01; Position: 42 - 42]

36.

"Allein die Entwicklungsumgebung sind heutzutage so überladen. Da muss man schon erklären, wie das funktioniert."

[Expertinnen-Interviews\EI01; Position: 43 - 43]

37.

"Manche Studierende sitzen da und verstehen einfach die Compiler-Meldungen nicht. Und dann reichen Hinweise zur Doppelklicken auf den Fehler und zum Anschauen der betreffenden Zeile."

[Expertinnen-Interviews\EI01; Position: 52 - 52]

38.

"Andere Studierende sitzen an irgendeinem Fehler und finden ihn einfach nicht und eigentlich sieht man da meistens sofort, da ist irgendwie der Ausdruck in der Bedingung vertauscht, oder es ist ein Syntaxfehler. Bevor die Studierenden an so einer Stelle ewig hängen, weil zugegebenermaßen gerade in C, oder C++ sieht man es erst mit ein bisschen Erfahrung schneller. Und das ist dann so frustrierend. Da sage ich dann doch lieber mal Bescheid. Und dann sehen die Studierenden, dass der Fehler unbedeutend war und es geht weiter."

[Expertinnen-Interviews\EI01; Position: 53 - 53]

39.

"Man muss mehr im mathematischen Formel-Kalkül denken. Es ist eine Sprache, aber anders als Englisch ist es doch sehr formal geprägt."

[Expertinnen-Interviews\EI01; Position: 66 - 66]

40.

"Das ist generell das mathematische Denken an der Stelle. Es müssen nicht Differenzialgleichungen hoch und runter sein. Aber ein bisschen das analytische Denken muss geschult werden. Und das ist das, was Programmieren vom Rest unterscheidet."

[Expertinnen-Interviews\EI01; Position: 68 - 68]

41.

"Und das zweite ist logisches Denkvermögen."

[Expertinnen-Interviews\EI02; Position: 2 - 2]

42.

"Zu Beginn überlegt man, was alles zum Fußballspiel dazugehört, ohne diese Machbarkeits-Kriterien zu berücksichtigen. Und dann kann man sich einen Kriterienkatalog überlegen, wie zum Beispiel: Machbarkeit, Zeitaufwand, Personalmöglichkeiten, Geldmöglichkeiten."

[Expertinnen-Interviews\EI02; Position: 8 - 8]

43.

"Dann kann man die Dinge sortieren. Die Fußballspielerin wird als sprachliches Konstrukt eine Klasse. Die Verben werden zu Methoden, und alles was so Substantive sind, werden

Attribute. Das heißt, Sie können sich, rein sprachlich an Ihren Lösungseinsatz annähernd und Software von vornherein strukturieren, ohne überhaupt irgendwas von Programmierung verstanden zu haben."

[Expertinnen-Interviews\EI02; Position: 9 - 9]

44.

"Studierende müssen vorm Programmieren Ihre Software schon gliedern, indem Sie sich einfach nur an der Realität orientieren und das sprachlich zerlegen."

[Expertinnen-Interviews\EI02; Position: 10 - 10]

45.

"Das heißt, die Studierenden haben zwar vorher schon mal was über Objektorientierung gehört, aber das sind eher grobe Techniken. Die kriegen einfach nur gesagt es gibt die Technik. Das heißt, es gibt Klassen. Aber die Ideen dahinter haben die noch nicht richtig verstanden. Sie wissen nur, es gibt Techniken, aber was die Philosophie dahinter ist, das verstehen die noch nicht. Und das wird Ihnen damit aber klarer."

[Expertinnen-Interviews\EI02; Position: 12 - 12]

46.

"Das Problem ist, wenn sie jetzt eine Struktur haben, was haben Sie dann eigentlich erstmal gemacht? Sie haben eigentlich erstmal nur ein größeres Problem in kleinere Teile zerlegt. Sie haben dieses komplexe Problem Fußballspiel, und haben jetzt ein Element, den Fußballspieler, und müssen sich jetzt auf den konzentrieren. Aber das ist nur ein Teil von dem ganzen großen Problem. Das heißt, Sie haben dieses Teilproblem."

[Expertinnen-Interviews\EI02; Position: 13 - 13]

47.

"Was unsere Studierende eher als Problem haben, ist was machen sie in dem Fußballspieler? Das Programmieren der einzelnen Methoden ist schwierig. Das ist dann der Kern, diese Methoden mit Leben zu füllen."

[Expertinnen-Interviews\EI02; Position: 14 - 14]

48.

"Und um noch weiterzugehen, wie das dann vor allem alles zusammenspielt. Also wie die Algorithmen intern zusammenwirken."

[Expertinnen-Interviews\EI02; Position: 15 - 15]

49.

"Und zum Teil bei den schwersten Fälle, ist es das logische Denkvermögen. Wenn das logische Denkvermögen fehlt, können wir hier nicht mehr viel machen. Das ist in meinen Augen die grundsätzliche Voraussetzung."

[Expertinnen-Interviews\EI02; Position: 18 - 18]

50.

"Was Studierenden fehlt, ist das Gefühl, wie kann man eine Idee repräsentieren. Also zum Beispiel, will man einen Pass von A nach B spielen, muss man zuerst repräsentieren, dass ein Spieler einen Ball hat. Da gibt es viele ineffiziente Lösungsmöglichkeiten. Aber auf solche Ideen kommen die Studierenden. Es ist grundsätzlich in vielen Fällen nicht falsch, aber es ist umständlich."

[Expertinnen-Interviews\EI02; Position: 19 - 19]

51.

"Zu dem logischen Denkvermögen nochmal. Während meines Informatik-Studiums war die Informatik noch ziemlich jung, und die Professoren waren fast alle Mathematiker. Das war schwierig für mich, weil ich schon mit Programmiererfahrung hingekommen bin und gedacht habe, ich kann schon programmieren. Aus dem Studium habe ich inhaltlich nichts gebraucht. Aber in den Fächern Elektrotecknik, Mathematik und Informatik ging es nur im Denken und nichts Anderes. Und wenn Sie denken können, dann können Sie sich nach dem Studium an die speziellen Gegebenheiten in ihrem Job anpassen."

[Expertinnen-Interviews\EI02; Position: 20 - 20]

52.

"B: Das logische Denken kann man erlernen. Das Gehirn ist definitiv trainierbar, auf jeden Fall."

[Expertinnen-Interviews\EI02; Position: 21 - 21]

53.

"Und das andere ist, wenn es jetzt wirklich um das Programmieren geht, dass man wesentlich mehr Dinge weiß. Man hat ein Problem, und kennt genau die Vor- und die Nachteile. Und Programmieren ist eine Folge von Kompromissen. Die perfekte Lösung gibt es normalerweise nicht. Sie müssen immer Vor- und Nachteile abwägen und am Schluss einen Kompromiss schließen. Und zwar am besten den, der die meisten Vorteile bringt, und negativ ausgedrückt, der Ihnen die wenigsten Nachteile liefert. Und je komplexer eine Software wird, desto mehr Kompromisse haben Sie darin und desto mehr haben Sie manchmal kein gutes Gefühl mehr."

[Expertinnen-Interviews\EI02; Position: 43 - 43]

54.

"B: Ja das Feedback wirkt sich aus auf die Gliederung von komplexen Algorithmen und komplexen Programmen, das aus Teilproblemen besteht, und in dem Bausteine miteinander kommunizieren müssen. Das Problem zerlegen die Studierenden in Teilprobleme, da sieht man die Fortschritte."

[Expertinnen-Interviews\EI02; Position: 57 - 57]

55.

"B: Der große Vorteil von Objektorientierung ist die Realität als Vorbild. Die Studierenden können die Realität als Vorbild nehmen und daran ihren System-Entwurf machen. Der System-Entwurf klappt nicht bei Dingen, die da sind, Sie aber nicht sehen können. Und das abstrakte in einem Software-Entwurf zu integrieren ist ein bisschen schwierig."

[Expertinnen-Interviews\EI02; Position: 58 - 58]

56.

"Und die Umsetzung von abstrakten Dingen im Software-Entwurf zu hinterfragen, ist viel Wert."

[Expertinnen-Interviews\EI02; Position: 60 - 60]

57.

"Um die Kompetenz der Studierenden zu erkennen, dazu gibt es eine Art Lakmus-Test. Ich lasse auf Zeit und Wochentage programmieren. Dann lasse ich einen Vergleichsalgorithmus programmieren in den ersten paar Wochen, der zeigen soll, was es heißt, wenn ein Zeitpunkt früher ist, als der andere. Das ist eine Kleinigkeit."

[Expertinnen-Interviews\EI03; Position: 3 - 3]

58.

"Und was sichtbar ist, wenn Studierende Sachverhalte erkennen. Das hat auch nichts mit Programmieren zu tun, wenn ich weiß, dass Tag X vor Tag Y ist. Dann kann ich auch rechnen, dass ein Tag nach einem ist. Das ist die gleiche Aufgabe, ich muss nur die beiden Tage tauschen."

[Expertinnen-Interviews\EI03; Position: 25 - 25]

59.

"B: Also programmieren heißt ja erst mal ein Programm zu schreiben, das überhaupt übersetzt und läuft. Das ist schon eine Leistung."

[Expertinnen-Interviews\EI03; Position: 70 - 70]

60.

"Vorher braucht man gar nicht über den Inhalt nachdenken, wenn es noch daran liegt, dass eine Klammer nicht zugeht. Aber das sagt einem eigentlich schon der Compiler. Ich sage den

Studierenden, dass der Compiler hilfreich ist und wo man nachschauen muss, in welcher Zeile, an welcher Stelle, bei welcher Klammer, bei welchem Integer oder String."

[Expertinnen-Interviews\EI03; Position: 71 - 71]

61.

"Ich versuche auch beizubringen, dass man das Feedback vom Compiler lesen kann."

[Expertinnen-Interviews\EI03; Position: 74 - 74]

62.

"Dann geht es darum, es soll 42 rauskommen, und bei den Studierenden kommt siebzehn raus. Hier Da kann man selber über die Ursachen nachdenken. Meistens hilft das schon."

[Expertinnen-Interviews\EI03; Position: 75 - 75]

63.

"Und diese Fehler lassen sich alle durch das Tool (den Compiler) überprüfen."

[Expertinnen-Interviews\EI03; Position: 76 - 76]

64.

"B: Wer Sachverhalte über Programmierung ausformuliert aufschreiben kann, kann das mit Sicherheit auch programmieren. Weil er zeigt mit dieser Kompetenz, dass er weiß, wie es geht. Er kann das, indem er auf der Metaebene was darüber aufschreibt. Diejenigen mit einer hohen Punktzahl bei dieser Aufgabe haben bestanden, das weiß ich."

[Expertinnen-Interviews\EI03; Position: 93 - 93]

65.

"Es gibt manche, die haben da wenig gemacht, die können irgendwie intuitiv schon programmieren, und die kommen dann auf eine 3, oder 2 Minus. Aber die können noch nicht darüber reden. Es könnte auch sein, dass es bei manchen an Sprache und Motivation liegt. Aber viele, die da einen oder null Punkte von zwanzig haben in der Klausur, da bin ich ziemlich sicher, dass die Programme auch nichts sind."

[Expertinnen-Interviews\EI03; Position: 94 - 94]

66.

"B: Beim Programmieren gibt es sehr unterschiedliche Aufgabentypen. Eine ganz klassische Fähigkeit ist es, Programme systematisch anpassen zu können. Denken wir mal an das User-Interface, wo ich nicht nur einen neuen Stil reinbringe. Damit meine ich anpassen an das, was schon da ist, das verstehen, konzeptionalisieren, und ergänzen. Das ist so eine Fähigkeit, das kann man am Anfang nicht und muss sich im Laufe des Studiums entwickeln, dass bestimmte Sachen systematisch weiterentwickelt werden können."

[Expertinnen-Interviews\EI04; Position: 2 - 2]

67.

"B: Es gibt unterschiedliche Dimensionen. Da gibt es die Dimension Technik. Das sind zum Beispiel Datenbanken, Netzwerke, Kommunikationsteile, Internet, und so weiter. Das sind rein technologische Skills. Da kann man sicher noch Visualisierung drunter zählen, User-Interfaces, Human-Computer-Interaction. Alles das sollte man mal gelernt haben."

[Expertinnen-Interviews\EI04; Position: 4 - 4]

68.

"B: Es geht natürlich auch um das elementare Programmieren. Das heißt, das Zerlegen von Problemen in Einheiten, die man direkt in die Programmiersprache abbilden kann."

[Expertinnen-Interviews\EI04; Position: 20 - 20]

69.

"B: Kenntnis allein reicht natürlich nicht aus. Was sie lernen sollen, ist Problemlösungen mit der Programmiersprache."

[Expertinnen-Interviews\EI04; Position: 21 - 21]

70.

"Und dazu müssen sie das Problem analysieren können."

[Expertinnen-Interviews\EI04; Position: 22 - 22]

71.

"Das Problem muss aufbereitet werden, sodass es in eine Programmierung umsetzbar ist."

[Expertinnen-Interviews\EI04; Position: 23 - 23]

72.

"Dann müssen die Studierenden diese Umsetzung leisten können."

[Expertinnen-Interviews\EI04; Position: 24 - 24]

73.

"Und Studierende müssen den Test leisten können."

[Expertinnen-Interviews\EI04; Position: 25 - 25]

74.

"Also das ist als Ziel, Programmieren als Mittel zur Problemlösung kennenzulernen."

[Expertinnen-Interviews\EI04; Position: 26 - 26]

75.

"Und so ein passives Behalten reicht ja da nicht aus, sondern ich muss das ja aktiv als Sprachelement verfügbar haben, insofern hat es natürlich Ähnlichkeiten mit dem Lernen einer natürlichen Sprache."

[Expertinnen-Interviews\EI04; Position: 28 - 28]

76.

"Die Studierenden müssen die Sprachkonstrukte selber anwenden können. Und sie müssen die Programmiersprache aktiv benutzen können."

[Expertinnen-Interviews\EI04; Position: 30 - 30]

77.

"Das zweite ist ein Schritt, den viele Leute unterschätzen. Der zweite Schritt ist dieses Aufbrechen des Problems in Einheiten, die ich dann direkt programmieren kann. Also dieses Zerlegen des Problems."

[Expertinnen-Interviews\EI04; Position: 34 - 34]

78.

"Es geht um den Schritt, ein Problem umzusetzen in eine Lösung durch ein Programm. Und auch zu lernen, was geht und was geht nicht."

[Expertinnen-Interviews\EI04; Position: 37 - 37]

79.

"B: Der Interpreter sagt einfach, etwas ist falsch. Man muss erstmal lernen, seine Ausgabe zu interpretieren. Er sagt ja in der Regel ein bisschen mehr, aber das kommt bei vielen Leuten, wenn sie das noch nicht gelernt haben, diese Fehlerausgabe zu interpretieren, nicht an. Sie sehen nur, dass etwas falsch ist."

[Expertinnen-Interviews\EI04; Position: 51 - 51]

80.

"Also das kann auch nicht das Hauptproblem der professionellen Produkte sein, der Interpreter oder Compiler. Die sagen, was falsch ist, aber die erwarten schon, dass man gelernt hat, mit ihnen umzugehen. Und die erwarten schon, dass man die Sprache im Grunde genommen wirklich kennt, und helfen dann den Experten weiter."

[Expertinnen-Interviews\EI04; Position: 79 - 79]

81.

"B: Für mich heißt programmieren sich strukturiert über Abläufe Gedanken machen zu können. Sich dort insbesondere bewusst zu werden über bestimmte Modelle von Strukturen wie Schleifen, bedingte Anweisungen. Dementsprechend muss man einen ganzen Prozess mit

seinen Möglichkeiten im Blick haben und das in einer formalisierten Art und Weise in (ggf. grafischen) Programmiersprachen ausdrücken zu können."

[Expertinnen-Interviews\EI05; Position: 1 - 1]

82.

"B: Für mich wird Programmieren erkennbar, indem Probleme, die im akademischen Bereich manchmal künstlich sind, die aber auch immer eine Anlehnung an echte Aufgabenstellungen haben oder auch echte Aufgabenstellungen sind, in einer qualitativ hochwertigen Art durch Programme gelöst werden."

[Expertinnen-Interviews\EI05; Position: 2 - 2]

83.

"B: Qualität bedeutet, dass Portabilität und Nachhaltigkeit und Wartungsfreundlichkeit wichtig sind, aber auch die alten Kriterien, dass es überhaupt läuft sowie Benutzerfreundlichkeit und Effizienz."

[Expertinnen-Interviews\EI05; Position: 3 - 3]

84.

"B: Bei den Studierenden lässt sich an der Lösung und an der Präsentation erkennen, ob sie programmieren können. Effizient schaue ich mir nicht die Programme an. Ich glaube das ist so vielfältig und das ist so eine Art Prüfungssituation. Es geht um Vorstellungen, die man hat und ich glaube die Vorstellungen kommen in so einer Situation durch die Präsentation gut raus und man kann dann probieren, die in brauchbarere Vorstellungen umzuwandeln, wenn man denkt, dass die ungünstig sind. Früher hat man von Fehlvorstellungen geredet. Das macht man ja heute nicht mehr."

[Expertinnen-Interviews\EI05; Position: 4 - 4]

85.

"B: Als Lehr-/Lernziele werden Modellbildung und Problemlösen angestrebt, das heißt einen Ausschnitt der Welt zu modellieren und sich an Schemata zu orientieren."

[Expertinnen-Interviews\EI05; Position: 5 - 5]

86.

"Und Aufgabenstellung so herunter zu brechen, wie es für die aktuelle Problemstellung nötig ist."

[Expertinnen-Interviews\EI05; Position: 6 - 6]

87.

"Problemlösungen muss man dann so modellieren, dass es die Maschine versteht. Das Modellieren kommt also am Anfang und Ende."

[Expertinnen-Interviews\EI05; Position: 8 - 8]

88.

"B: Das kann man noch herunterbrechen. Beim Problemlösen gibt es die bottom-up, top-down und die greedy und die Varianten. Die können konkretisiert werden, aber den roten Faden zu behalten, ist wichtiger. So wird auf wenige Elemente reduziert, die immer wieder genutzt werden."

[Expertinnen-Interviews\EI05; Position: 9 - 9]

89.

"B: Beim Modellieren werden Prozesse ganz abstrakt heruntergebrochen, indem man ein Schema anwendet."

[Expertinnen-Interviews\EI05; Position: 10 - 10]

90.

"Und dann kann man ein Problem abbilden indem man UML-Notationen nutzt oder wie bei den agilen Prozessen User-Stories baut."

[Expertinnen-Interviews\EI05; Position: 11 - 11]

91.

"Und natürlich, die grundsätzliche Kompetenz ist, das Problem zu lösen. Das heißt natürlich nicht, dass die sich dafür lange damit beschäftigt haben."

[Expertinnen-Interviews\EI05; Position: 20 - 20]

92.

"B: Also gute Programmierer erkennt man an den Problemlösungen."

[Expertinnen-Interviews\EI05; Position: 25 - 25]

93.

"Man muss berücksichtigen, dass das Programm danach noch von vielen Leuten angeguckt werden muss. Und dass man die Aufgaben eben anders als nur durch seine eigene bloße Kreativität lösen kann. Oder wo noch eine andere kreative Komponente dazukommt, nämlich das Vorausschauende, das man einmal die Benutzer und andererseits die Programmierer, die nach einem kommen, im Blick hat."

[Expertinnen-Interviews\EI05; Position: 30 - 30]

94.

"B: Programmieren bedeutet ein Problem zu lesen."

[Expertinnen-Interviews\EI06; Position: 1 - 1]

95.

"Zu dem Problem wird ein funktionales Programm geschrieben, das getestet ist."

[Expertinnen-Interviews\EI06; Position: 2 - 2]

96.

"Das Programm sollte einen guten Stil haben, gut lesbar, wartbar und übersichtlich sein, sodass andere damit arbeiten können."

[Expertinnen-Interviews\EI06; Position: 3 - 3]

97.

"Man muss die Konstrukte der Programmiersprache kennen und die passenden einsetzen."

[Expertinnen-Interviews\EI06; Position: 5 - 5]

98.

"B: Schleifen schreiben am Anfang kriegen die Studierenden hin."

[Expertinnen-Interviews\EI06; Position: 30 - 30]

3.4. Meta-kognitives Wissen

1.

"Bei dem Projekt wird selbstständig gearbeitet."

[Expertinnen-Interviews\EI00; Position: 8 - 8]

2.

"Die Studierenden sind der Meinung, es ist die Verantwortung der Lehrperson, alles zu erklären was sie nicht verstehen."

[Expertinnen-Interviews\EI00; Position: 30 - 30]

3.

"Anstatt Literatur zu bearbeiten, weisen die Studierenden die Schuld der Lehrperson zu. Die Verantwortung, aktiv zu werden, verstehen viele im ersten Semester nicht."

[Expertinnen-Interviews\EI00; Position: 31 - 31]

4.

"Zu dieser Haltung und Einstellung zur fehlenden Selbstverantwortung beim Lernen muss Feedback gegeben werden, unter Umständen in Form der nicht bestandenen Klausur."

[Expertinnen-Interviews\EI00; Position: 32 - 32]

5.

"An den Hochschulen besteht unabhängig von der Programmierung das große Problem, dass viele Studierende im ersten Semester dieses universitäre Arbeiten noch nicht verinnerlicht haben. In der Schule habe ich etwas nicht verstanden, weil der Lehrer es nicht ordentlich erklärt hat. Wenn ich etwas an der Universität nicht verstehe, dann hätte ich selbst nachschauen müssen. Diese Beweislastumkehr haben viele noch nicht richtig verstanden."

[Expertinnen-Interviews\EI00; Position: 33 - 33]

6.

"Diejenigen, die schon eine Lehre gemacht haben, haben schon die Einstellung, nicht Kunde zu sein. Sie treten stattdessen in Vorleistung und haben diese Haltung verinnerlicht."

[Expertinnen-Interviews\EI00; Position: 39 - 39]

7.

"Studierende, die direkt nach dem Abitur mit dem Studium beginnen, haben diese Einstellung, nicht mehr Kunde zu sein, noch nicht."

[Expertinnen-Interviews\EI00; Position: 40 - 40]

8.

"B: Wer programmieren kann, versucht selbstständig, ein Ziel zu erreichen, die Aufgabe umzusetzen, ohne ständig zu schauen, die Lösung durch andere Kommilitonen zu erhalten, während man selbst nichts gemacht hat."

[Expertinnen-Interviews\EI01; Position: 2 - 2]

9.

"Wer das macht, versucht es auch selbstständig, evtl. auch durch googeln, auf Stack-Overflow schauen, etc."

[Expertinnen-Interviews\EI01; Position: 5 - 5]

10.

"B: Es gibt auch Studenten, da merkt man, die fangen überhaupt erst an, versuchen es aber trotzdem. Sie haben dann viele Zwischenfragen, nutzen ein Fachbuch zum Thema."

[Expertinnen-Interviews\EI01; Position: 10 - 10]

11.

"B: Es gibt Studenten, die denken sie können es, weil sie Vorkenntnisse aus der Fachoberschule haben. Er hat dann aber festgestellt, er kann es doch nicht. Er hat sich dann hingesetzt und dann hat es geklappt. Es kommt auf den Einzelfall an."

[Expertinnen-Interviews\EI01; Position: 15 - 15]

12.

"B: Wenn die Studierenden selber etwas gemacht haben, ist es in Ordnung. Dann haben sie eine Lösung und selber was gemacht."

[Expertinnen-Interviews\EI01; Position: 40 - 40]

13.

"Wenn man selbst einschätzt, inwieweit man selber programmieren kann, ist das an der Stelle auch noch wichtig, gerade in den frühen Semestern."

[Expertinnen-Interviews\EI01; Position: 64 - 64]

14.

"B: In der Germanistik gibt es auch diese zwei Teile. Es gibt zum einen die Literaturwissenschaft, wo man interpretieren kann und zum anderen die Sprachwissenschaft. Und da gibt es diejenigen, die eher mit der Literatur gut klarkommen. Und dann gibt es die Studierenden, denen Sprachwissenschaft eher lag. Und Letzteres ist die Denkrichtung, die Richtung Programmieren geht. Das sind einfach vielleicht einfach zwei Arten zu Denken. Kann man vielleicht lernen, weiß ich nicht, kenne ich mich nicht genügend aus. Das ist in anderen Bereichen vermutlich ähnlich."

[Expertinnen-Interviews\EI01; Position: 69 - 69]

15.

"B: Programmieren setzt sich aus zwei wesentlichen Dingen zusammen. Das eine ist Intuition. Intuition ist etwas, was sich nur schwer lernen lässt. Das ist der knifflige Teil. Das ist oft so: hat man früh genug damit angefangen, oder nicht. Wenn Sie zum Beispiel als Kind etwas gelernt haben, eine Sprache, ein Musikinstrument, dann ist das so ähnlich. Wenn Sie etwas als Kind so gelernt haben, ist es etwas völlig Selbstverständliches."

[Expertinnen-Interviews\EI02; Position: 1 - 1]

16.

"Das Kernproblem, was die Studierenden bei uns haben, ist von der Aufgabenstellung ausgehend eine Lösung zu finden. Das hat etwas mit Intuition zu tun. Das ist etwas, was sich nur schwer lernen lässt, oder sich im späteren Alter mit viel Aufwand kompensieren lässt."

[Expertinnen-Interviews\EI02; Position: 4 - 4]

17.

"Ich lasse die Studierenden zum Beispiel ein Fußballspiel programmieren, damit sie diese Herangehensweise lernen."

[Expertinnen-Interviews\EI02; Position: 7 - 7]

18.

"B: Das kriegen die Studierenden soweit hin. Da kommen die von selber drauf. Es ist oft intuitiv klar, weil ich setze im dritten, vierten Semester an."

[Expertinnen-Interviews\EI02; Position: 11 - 11]

19.

"B: Das ist immer das Schöne, wenn man sieht, wie Studierende dazulernen. Das wird sichtbar, indem Studierende systematischer an Problemlösungen rangehen. Das ist das Entscheidende, was ich für mich im Studium mitgenommen habe."

[Expertinnen-Interviews\EI02; Position: 22 - 22]

20.

"Und man muss die systematische Vorgehensweise haben, wie man da rangeht."

[Expertinnen-Interviews\EI02; Position: 24 - 24]

21.

"B: Es hat sehr viel mit Charakter zu tun, und damit, seine eigenen Grenzen zu erkennen. Irgendwann muss man lernen, oder akzeptieren, dass es auch andere Leute gibt, die das vielleicht viel toller können, als man das selbst kann und dass man von denen viel lernen kann. Und das Entscheidende ist, nicht gleich loszulegen und irgendetwas zu versuchen, weil man denkt, man kann alles."

[Expertinnen-Interviews\EI02; Position: 26 - 26]

22.

"Es macht mehr Sinn macht, erst mal lange und gründlich zu recherchieren und von anderen zu lernen."

[Expertinnen-Interviews\EI02; Position: 27 - 27]

23.

"Und das meine ich mit eigene Grenzen kennen, das Rad nicht tausend Mal neu erfinden, sondern von anderen lernen."

[Expertinnen-Interviews\EI02; Position: 28 - 28]

24.

"B: Man darf nicht vergessen, dass die Studierenden junge Menschen sind. Es gibt Studierende, die sind so alt wie ich, das kann passieren. Das sind aber Ausnahmen. Es gibt so Menschen, die bauen sich ein Eigenuniversum auf, das sie niemals durchbrechen werden. Die haben eine Selbstwahrnehmung, die dem objektiven Eindruck entgegenspricht. Aber bei den meisten, das ist einfach eine Erfahrung, die sie machen müssen, Und wenn die diese Erfahrung mal gemacht haben, dann klappt das auch."

[Expertinnen-Interviews\EI02; Position: 29 - 29]

25.

"Und ich bin auch immer der Meinung, es gibt so ein paar einschneidende Erlebnisse. Wir haben zum Beispiel Studierende, die haben Kinder. Wenn man ein Kind hat, muss man mit ganz anderen Verantwortungen darangehen. Und das merkt man im Studium. Die haben eine andere Herangehensweise. Die wissen auch, dass es wichtigere Schwerpunkte gibt, als nur zu zocken, oder Spaß zu haben. Und das ist ein permanenter Lernprozess. Also die Hoffnung muss man da nie aufgeben."

[Expertinnen-Interviews\EI02; Position: 30 - 30]

26.

"B: Die Studierenden bekommen genau gesagt, was in den Klausuren drankommt, sie kennen die Aufgabenstellungen, die kommen können. Sie kennen alten Klausuren, auch wenn ich die regelmäßig ändere. Von den Ideen her, ist es doch ähnlich. Wenn man sich vernünftig vorbereitet, kann man die Klausuren ordentlich bestehen."

[Expertinnen-Interviews\EI02; Position: 35 - 35]

27.

"Bei einer Programmieraufgabe ist es ähnlich wie in einer Mathematik-Aufgabe, da gibt es wenige Grauzonen. Entweder jemand macht was, und das kann er auch auf verschiedene Wege machen. Es gibt da nicht unbedingt den einen Ziel-Weg. Aber es gibt keine Ungewissheit."

[Expertinnen-Interviews\EI02; Position: 37 - 37]

28.

"Also es sind reine Programmieraufgabe, die in irgendeiner Form in den Vorlesungen oder in den Übungen gemacht wurden. Wenn Studierende die Übungen, die Vorlesung mitgemacht hat, wenn er sich auf die Klausur vorbereitet hat, dann kann zumindest diese Aufgabenstellung bewältigt werden."

[Expertinnen-Interviews\EI02; Position: 38 - 38]

29.

"Man muss den Transfer schaffen von Übungsaufgabe zu Prüfungsaufgaben. Wobei ich zum Teil konkret Prüfungsaufgaben vorher übe."

[Expertinnen-Interviews\EI02; Position: 39 - 39]

30.

"B: Am schnellsten lernt man Programmieren, wenn man sich Code von anderen anschaut. Das ist sehr wichtig. Weil nur, wenn man sich Sachen von anderen anguckt, bekommt man neuen Input. Ansonsten bewegt man sich immer in seinen eigenen Bahnen. Man hat ein Problem, man stellt sich die Lösung dafür vor, sieht die Lösung von jemand anderem und merkt daran, was jemand anders besser gemacht hat. Davon lernt man sehr viel."

[Expertinnen-Interviews\EI02; Position: 40 - 40]

31.

"B: Beim Quellcode von Kommilitonen ist Vorsicht geboten. Das kann gut gehen, wenn Studierende wirklich wissen, was Sie tun. Beim Internet als Quelle müssen die Quellen ganz genau angeschaut werden. Es gibt ein paar gute Bücher, wo man sich sowas anschauen kann. Und dann, muss man eigentlich wirklich gezielt im Internet suchen. Beziehungsweise was es auch gibt, sind Open Source Bibliotheken und Produkte. Und da gibt es einige im Programmierbereich, die sind richtig gut, wie zum Beispiel die Boost Library im C++ Bereich. Das ist oft gereviewt worden, weltweit von vielen Leuten."

[Expertinnen-Interviews\EI02; Position: 41 - 41]

32.

"Normalerweise nehmen diejenigen, die Fragen stellen, die Hilfe ernst. In der Regel sieht man das in den nächsten Übungen. Eine zweite Lösung nach der Hilfestellung ist dann schon besser, aber es muss noch viel gemacht werden."

[Expertinnen-Interviews\EI02; Position: 55 - 55]

33.

"Das darüber nachdenken, was ich da eigentlich mache, also oder Reflektieren ist bei uns oft der Fall. Das muss ich begleiten in der Vorlesung. Aber wir haben dann zum Beispiel auch eine Vorlesung, die das dediziert zum Thema hat. In Algorithmen und Datenstrukturen ist das dediziert zum Thema. Dann über Aufwandsabschätzungen, O-Notation nachzudenken. Die Software-Technik hat das Thema, wie man es überhaupt, wie macht man das in Projekten macht, wie man es dokumentiert."

[Expertinnen-Interviews\EI03; Position: 14 - 14]

34.

"Man erkennt es, aber das gilt nicht nur für Programmieren. Man erkennt es eher an der Arbeitssystematik. In der ersten Praktikumsstunde im ersten Semester programmiere ich noch nicht, sondern habe Denksportaufgaben dabei, Knobelaufgaben, oder eine Aufgabe aus einem Kinderbuch. Der Punkt ist aber nicht, die Aufgabe sofort zu lösen."

[Expertinnen-Interviews\EI03; Position: 19 - 19]

35.

"Das Hauptproblem, wenn es mit der Programmierung nicht klappt, liegt bei den Studierenden, nicht an der Programmierung, sie würden auch bei Literaturwissenschaft scheitern, oder bei Maschinenbau, weil es an der Beschäftigung mit Problemen, oder der Herangehensweise mit Problemen liegt."

[Expertinnen-Interviews\EI03; Position: 22 - 22]

36.

"B: Also ein Knackpunkt, der schwer ist, ist vielleicht Rekursion. Wer Rekursion sofort schluckt und versteht, kann programmieren."

[Expertinnen-Interviews\EI03; Position: 29 - 29]

37.

"B: Rekursion enthält die Essenz dessen, was unsere gängige Abstraktion der Programmierung ist. Wer Rekursion verstanden hat, kann Funktionsaufrufe, kennt Funktionen. Wer Rekursion kann, hat zumindest intuitiv, technisch gesehen, das Konzept vom Stack verstanden. Wer Rekursion kann, kann Induktionsbeweise. Vollständige Induktion ist Rekursion in der Mathematik. Wer Rekursion kann, kann Mathematik dazu."

[Expertinnen-Interviews\EI03; Position: 30 - 30]

38.

"B: Man könnte sagen, ein Programm ist diskrete Mathematik in angewandter Form. Typen sind Mengen, Funktionen sind Programme, das Datenmodell ist Mathematik."

[Expertinnen-Interviews\EI03; Position: 31 - 31]

39.

"Ich mache das Internet auf zum Programmieren und da gibt es so viel Information. Und es gibt vielleicht Informationen, die den Anfänger verwirren. Wenn Studierende versuchen, sich Inhalte über andere Quellen anzueignen, anstatt mal meinem Kurs zu folgen, kann das problematisch werden."

[Expertinnen-Interviews\EI03; Position: 52 - 52]

40.

"Die Studierenden sollten sich eher meinen Kurs, mein Skript anschauen, nicht weil es besser ist, sondern weil ich zum Schluss prüfe. Und alles, was ich prüfe, steht in meinem Material. In den anderen Dingern steht ganz viel Information, ganz viel wichtige und gute Information. Aber die wird im ersten Semester verwirren."

[Expertinnen-Interviews\EI03; Position: 53 - 53]

41.

"Und das sieht man manchmal, wenn jemand anders lernt, oder versucht mit anderen Unterlagen zu lernen. Da muss er sehr gut sein, wenn er das kann, um das zu verstehen. Die Terminologie ist nicht einheitlich zwischen der Literatur. Auf Webseiten steht, wie man es machen kann, aber der Kontext oder Spezialfälle fehlen. Oder ich habe gerade erst mal dieses eine Konzept beigebracht. Also das ist hinderlich, was ich manchmal beobachte. Das ist auch nicht falsch, aber es passt gerade nicht zu dem, was sie gerade lernen sollen. Nach zwei Semestern, drei Semestern sollte sich das einspielen. Aber im ersten Semester kann das hinderlich sein."

[Expertinnen-Interviews\EI03; Position: 54 - 54]

42.

"B: Interessant ist immer, wenn man zeigen kann, eine individuelle Problemlösung lässt sich verallgemeinern. Weil das ist ja genau das, was wieder am Anfang ansetzt. Das macht die Erfahrung aus, wenn man sagt, Moment, ich habe doch mal ein ähnliches Programm gemacht, wann ist ein Wochentag vor einem anderen. Das geht natürlich mit Uhrzeiten genauso. Das geht aber auch mit Objekten im zweidimensionalen Raum und der Frage, welcher weiter links vom anderen liegt."

[Expertinnen-Interviews\EI03; Position: 67 - 67]

43.

"Das ist auch etwas, was man in der ganzen Softwaretechnik sieht, dass man individuelle Lösungskonzepte, Einzellösungskonzepte verallgemeinert zu allgemeinen Konzepten, mit der man eine ganze Gruppe von Problemen lösen kann."

[Expertinnen-Interviews\EI03; Position: 68 - 68]

44.

"Und das macht dann den Novizen und zum Schluss dann den Experten aus, ob man so seinen Baukasten an Strategien zur Problemlösung hat. Da sollen die Leute ja hinkommen."

[Expertinnen-Interviews\EI03; Position: 69 - 69]

45.

"B: Ja die Leute haben das Selbststudium nicht gelernt."

[Expertinnen-Interviews\EI04; Position: 55 - 55]

46.

"Viele Sachen sind gar nicht mehr im Lehrbuchstil verfügbar, sondern nur noch im Manual-Stil. Also in einer Sprachbeschreibung, die ganz anders strukturiert ist, viel schwieriger zu verstehen ist. Es werden Details erklärt, die Studierende überhaupt nicht verstehen, und noch nicht verstehen können, weil es kein Lehrbuch ist, das systematisch aufeinander aufbaut. Und trotz alledem müssen Studierende damit umgehen lernen."

[Expertinnen-Interviews\EI04; Position: 56 - 56]

47.

"B: Ich habe ziemlich viel nachgedacht über meine Rolle als Vorlesender, weil ich kann nicht jedes Detail der Programmiersprache in der Vorlesung vorstellen. Die Leute müssen einfach lernen, dass sie das sich aus eigener Kraft im Selbststudium teilweise erarbeiten müssen."

[Expertinnen-Interviews\EI04; Position: 88 - 88]

48.

"So und das Selbststudium ist schwer, weil das sind sie zum Beispiel von der Schule in der Regel nicht gewöhnt. Sondern da wird das geprüft, was in der Stunde besprochen wurde und alles andere wird so getan, als wenn es das nicht gibt. Und das ist hier natürlich gänzlich anders. Wichtig ist nicht das, was ich gesagt habe, sondern richtig ist das, was richtig belegbar, beweisbar richtig ist."

[Expertinnen-Interviews\EI04; Position: 89 - 89]

49.

"Meistens hilft es, wenn man sich an einem roten Faden festklammern kann. Bei Überforderung kann man dann Standardschema X ausprobieren. Wenn die eine Möglichkeit schiefgeht, kann man ich das nächste probieren."

[Expertinnen-Interviews\EI05; Position: 18 - 18]

50.

"B: Die Herausforderung in dem Lernprozess ist, das obere Ende zu erreichen, wo Studierende durch sehr einfache Schemata gute Erfolge bei Programmieraufgaben erzielen."

[Expertinnen-Interviews\EI05; Position: 27 - 27]

51.

"B: Neben Offenheit spielt strukturierte Denkfähigkeit eine Rolle."

[Expertinnen-Interviews\EI05; Position: 46 - 46]

52.

"An der Universität ist das Studium etwas Individuelles und kein verschultes System. Im Prinzip sind alle für sich selber verantwortlich. Man kann zwar Hilfestellung geben und sagen, aber letztlich ist jeder seines Glückes Schmied und ich zwinge niemanden, was abzugeben."

[Expertinnen-Interviews\EI05; Position: 59 - 59]

53.

"B: Die Lernerfolge sind völlig individuell. Bei einigen klappt der Transfer und bei anderen nicht. An der Uni hat man nicht den Anspruch, dass man auch die letzten mitnimmt. Manche Studierenden haben einen anderen Zugang und müssen nicht zwangsläufig eine Lernstrategie nutzen."

[Expertinnen-Interviews\EI05; Position: 63 - 63]

54.

"Man zeigt noch andere Möglichkeiten auf und nennt ein Lehrbuch, Literatur, etc."

[Expertinnen-Interviews\EI05; Position: 64 - 64]

55.

"Es ist außerdem eine sehr individuelle Sache, ob man diese eine Art Bloßstellung durch die Präsentation der eigenen Lösung persönlich aushält. Und wer das nicht aushält, hat den Nachteil, dass er das Feedback in dem Moment nicht kriegt. Das muss er sich eben anders holen."

[Expertinnen-Interviews\EI05; Position: 65 - 65]

56.

"B: Man muss algorithmisch denken, und sich überlegen, was in welcher Reihenfolge gemacht werden muss."

[Expertinnen-Interviews\EI06; Position: 4 - 4]

57.

"Man muss sich den gesamten Ablauf vorher überlegen."

[Expertinnen-Interviews\EI06; Position: 6 - 6]

58.

"Man muss gewohnt sein, an alle Randfälle zu denken, und diese berücksichtigen."

[Expertinnen-Interviews\EI06; Position: 7 - 7]

59.

"Die Studierenden haben mehr oder weniger viel Talent zum Programmieren. Manche haben schon eine Ablauf-orientierte Denkweise."

[Expertinnen-Interviews\EI06; Position: 12 - 12]

60.

"Viele Anfänger haben schon als Kind programmiert, wo man spielerisch anfängt, und das als ganz natürliche Herangehensweise betrachtet. Das ist sicherlich ein Vorteil. Wer eine Ablauf-orientierte Denkweise gewohnt ist, hat es beim Programmieren einfacher."

[Expertinnen-Interviews\EI06; Position: 19 - 19]

61.

"B: Wahrscheinlich ist das eine Denkweise, die in den Leuten drinsteckt oder nicht."

[Expertinnen-Interviews\EI06; Position: 20 - 20]

62.

"B: Weiß ich nicht, ob man diese Ablauf-orientierte Denkweise viel fördern kann. Natürlich indem man als Kind solche Sachen schon spielerisch macht, zum Beispiel mit einem programmierbaren Taschenrechner. Damit haben wir gespielt und in der Schule

verbotenerweise einen Wettbewerb daraus gemacht. Damit übt man das in einem Alter, wo man das noch mehr verinnerlicht."

[Expertinnen-Interviews\EI06; Position: 21 - 21]

63.

"Ich nehme an, dass das eher so ein bisschen was angeborenes ist. Oder man übernimmt diese Denkweise von den Eltern. Nicht überall im Leben ist das als sinnige Vorgehensweise zu gebrauchen, wenn man z. Bsp. nicht entscheidungsfreudig ist. Also ich würde da jetzt nicht unbedingt als ist gut oder ist schlecht betrachten."

[Expertinnen-Interviews\EI06; Position: 22 - 22]

64.

"Manche Studierende haben eine Ablauf-orientierte Vorgehensweise/Denkweise. Logistik ist sicherlich ähnlich von der Vorgehensweise."

[Expertinnen-Interviews\EI06; Position: 23 - 23]

65.

"B: Ein Ablauf-orientiertes Vorgehen hilft den Studierenden, programmieren zu lernen."

[Expertinnen-Interviews\EI06; Position: 24 - 24]

66.

"Mit wenig Zeit komplexe Programme zu schreiben, erfordert abstraktes Denken."

[Expertinnen-Interviews\EI06; Position: 27 - 27]

67.

"Rekursion ist schwierig."

[Expertinnen-Interviews\EI06; Position: 31 - 31]

68.

"Dann kommt Rekursion, wo das, was man hinschreibt nicht mehr direkt am Programm beobachtet werden kann. Damit hat man einen riesen Sprung und braucht dieses mathematische, ähnlich einem Induktionsbeweis. Das ist für viele überhaupt nicht machbar."

[Expertinnen-Interviews\EI06; Position: 33 - 33]

69.

"Rekursion ist das haarige Thema, was in der Vorlesung immer länger behandelt wird, obwohl es eigentlich nicht viel zu sagen gibt. Eigentlich könnte man das in zwanzig Minuten erklären, aber man braucht viele Beispiele. Manche kriegen das nicht hin, weil das ein Sprung ist, etwas anderes hinzuschreiben, als das was in Wirklichkeit abläuft. Einige Studierende empfinden das als zu fremd, unabhängig davon, wie oft man diese Besonderheit erklärt."

[Expertinnen-Interviews\EI06; Position: 36 - 36]

70.

"B: Es fällt den Studierenden schwer, wenn man das, was passiert, nicht sieht."

[Expertinnen-Interviews\EI06; Position: 37 - 37]

71.

"B: Die Studierenden selbst teilen nicht mit, was Ihnen schwer fällt. Sie teilen nur mit, nicht zu wissen, wie etwas funktioniert. Trotz mehrfacher Hinweise können die Rekursionsvorschriften nicht aufgeschrieben werden. Die Studierenden finden das merkwürdig."

[Expertinnen-Interviews\EI06; Position: 38 - 38]

72.

"Alles außer Rekursion kann mit mehr oder weniger viel Übung gemeistert werden, bei dem ein Aha-Erlebnis auftritt, wenn etwas entsprechend funktioniert."

[Expertinnen-Interviews\EI06; Position: 40 - 40]

73.

"B: Diejenigen Studierenden, die immer nachfragen, haben das das Vorgehen nicht verinnerlicht. Zu viel Unterstützung zu geben, ist kontraproduktiv. Studierende müssen sich alleine da durchkämpfen, dann ist der Lerneffekt höher, als mit viel Unterstützung."

[Expertinnen-Interviews\EI06; Position: 46 - 46]

74.

"Andere führen in Dreier-, Vierer-Gruppen ihre eigene Diskussion und fühlen sich genervt, wenn man fragt, ob man helfen kann."

[Expertinnen-Interviews\EI06; Position: 48 - 48]

75.

"Und es gibt einzelne, die alle fünf Minuten fragen. Bei denen ist Hopfen und Malz verloren. Die haben einen falschen Ansatz, wenn sie permanent fragen müssen."

[Expertinnen-Interviews\EI06; Position: 49 - 49]

76.

"Selbstständig denken und rumprobieren ist nützlich beim Programmieren, obwohl ich selbst nicht gerne probiere."

[Expertinnen-Interviews\EI06; Position: 50 - 50]

77.

"Selbstständiges Denken braucht man zum Erfolg."

[Expertinnen-Interviews\EI06; Position: 51 - 51]

78.

"Was behindert, ist Unselbstständigkeit. Man muss selbstständig denken und man darf nicht immerzu andere fragen."

[Expertinnen-Interviews\EI06; Position: 57 - 57]

79.

"B: Ich beobachte manchmal eine Gruppe von Studierenden, die sich während des gesamten Semesters permanent meldet und dann in der Klausur durchfällt. Die sind dann nicht selbstständig. Die selbstständigen sehe ich nicht."

[Expertinnen-Interviews\EI06; Position: 63 - 63]

80.

"Wer am Angang viel Hilfe sucht, tut das in der Regel kontinuierlich. Es kann sein, dass manche Studierende aufgehört haben, oder selbstständig geworden sind. Das kann ich nicht beurteilen."

[Expertinnen-Interviews\EI06; Position: 64 - 64]

81.

"Ich habe eine Studentin im Kopf, die permanent gefragt hat und durch Ausdauer den Bachelor-Abschluss geschafft hat. Aber selbstständig war sie nicht."

[Expertinnen-Interviews\EI06; Position: 65 - 65]

82.

"B: Ein wenig selbstständiger wird man mit zunehmendem Alter. Aber ich glaube es ist relativ fest gegeben, wie selbstständig man ist."

[Expertinnen-Interviews\EI06; Position: 66 - 66]

4. kognitive Prozessdimensionen

4.1. Erinnern

4.1.1. Elementare Programmiersprachliche Konstrukte kennen

1.

"B: Es gibt ein paar Auffälligkeiten beim Lernprozess. Auffällig ist zum Beispiel, dass es sehr viele Spracheigentümlichkeiten zu lernen gilt. Also, ich will nicht sagen Auswendiglernen, aber für einen Anfänger ist es natürlich viel, was er da alles sich behalten soll."

[Expertinnen-Interviews\EI04; Position: 27 - 27]

2.

"Es sind viele Sachen gar nicht so besonders schwer. Darum ist es wichtig, dass wir eine Programmiersprache haben, die sehr strikt ist und möglichst wenige Ausnahmen hat. Weil Ausnahmen müssen wirklich einzeln gelernt werden. Die anderen Ausdrücke sind unter den vielen Programmiersprachen sehr einheitlich. Da lernt man da nicht nur eine Programmiersprache, sondern man lernt eine ganze Gruppe von Programmiersprachen und das ist gut."

[Expertinnen-Interviews\EI04; Position: 32 - 32]

3.

"Aber sobald man Ausnahmen hat, geht gar nichts anderes, als das auswendig zu lernen. Und das ist am Anfang schlecht, weil ich dann noch zu den ganzen Begrifflichkeiten zusätzlich noch die Ausnahmen lernen muss."

[Expertinnen-Interviews\EI04; Position: 33 - 33]

4.1.2. Objektorientierte Programmierung kennen

1.

"Das heißt, die Studierenden haben zwar vorher schon mal was über Objektorientierung gehört, aber das sind eher grobe Techniken. Die kriegen einfach nur gesagt es gibt die Technik. Das heißt, es gibt Klassen. Aber die Ideen dahinter haben die noch nicht richtig verstanden. Sie wissen nur, es gibt Techniken, aber was die Philosophie dahinter ist, das verstehen die noch nicht. Und das wird Ihnen damit aber klarer."

[Expertinnen-Interviews\EI02; Position: 12 - 12]

4.1.3. Technologische Grundlagen kennen

1.

"B: Es gibt unterschiedliche Dimensionen. Da gibt es die Dimension Technik. Das sind zum Beispiel Datenbanken, Netzwerke, Kommunikationsteile, Internet, und so weiter. Das sind

rein technologische Skills. Da kann man sicher noch Visualisierung drunter zählen, User-Interfaces, Human-Computer-Interaction. Alles das sollte man mal gelernt haben."

[Expertinnen-Interviews\EI04; Position: 4 - 4]

4.2. Verstehen

4.2.1. Konzepte der Programmierung in eigenen Worten erklären

1.

"B: Meine Klausuren sind reine Programmierklausuren mit ein paar Theorie-Fragen. Letztere wurden die letzten Jahre reduziert. Oft ist es so, dass Studierende grundsätzlich eigentlich verstanden haben und erklären das in ihren eigenen Worten, aber es fehlt ein Stichwort. Dann gibt man den Punkt nicht."

[Expertinnen-Interviews\EI02; Position: 36 - 36]

2.

"Wichtig ist mir diese fünfte Aufgabe, in der etwas erklärt werden muss. Ich finde das wichtig, dass ein Informatiker gezwungen wird, was in Deutsch aufzuschreiben. Das ist die komplexeste Sache zu korrigieren. Das muss man lesen, Textverständnis aufbringen und abgleichen mit dem Text, den man selber geschrieben hätte."

[Expertinnen-Interviews\EI03; Position: 91 - 91]

4.3. Anwenden

4.3.1. Werkzeuge zur Software-Entwicklung nutzen

1.

"B: Ich will nicht, dass Studierende alle zwölf Methoden der Reflection-API aufsagen können. Das kann ich nachschauen. Aber die Studierenden müssen sie benutzen können, darum geht es."

[Expertinnen-Interviews\EI00; Position: 64 - 64]

2.

"Problematisch sind diejenigen, die es gar nicht erst versuchen. Da weiß ich auch gar nicht, wie diese Studierenden erreicht werden können. Wer nicht weiß, wie man eine Entwicklungsumgebung aufmacht, oder kompiliert, keine Syntaxfehler sieht und beheben kann, und es nicht ausprobiert, kann kein guter Programmierer werden. Das beobachte ich in den älteren Semestern. Im ersten Semester kann das sicherlich noch passieren, dass man die Fehler nicht versteht. Aber wenn das im dritten, vierten oder fünften Semester auftritt, ist das ein echtes Problem. Diese Probleme treten auf bei Studierenden, die irgendwas mit Medien machen wollten, aber nicht programmieren wollten."

[Expertinnen-Interviews\EI01; Position: 13 - 13]

3.

"Ganz viele scheitern schon daran richtig zu kompilieren. Da muss man von Anfang an helfen."

[Expertinnen-Interviews\EI01; Position: 42 - 42]

4.

"Allein die Entwicklungsumgebung sind heutzutage so überladen. Da muss man schon erklären, wie das funktioniert."

[Expertinnen-Interviews\EI01; Position: 43 - 43]

5.

"Und diese Fehler lassen sich alle durch das Tool (den Compiler) überprüfen."

[Expertinnen-Interviews\EI03; Position: 76 - 76]

4.3.2. Qualitätskriterien auf Quellcode anwenden

1.

"B: Qualität bedeutet, dass Portabilität und Nachhaltigkeit und Wartungsfreundlichkeit wichtig sind, aber auch die alten Kriterien, dass es überhaupt läuft sowie Benutzerfreundlichkeit und Effizienz."

[Expertinnen-Interviews\EI05; Position: 3 - 3]

2.

"Man muss berücksichtigen, dass das Programm danach noch von vielen Leuten angeguckt werden muss. Und dass man die Aufgaben eben anders als nur durch seine eigene bloße Kreativität lösen kann. Oder wo noch eine andere kreative Komponente dazukommt, nämlich das Vorausschauende, das man einmal die Benutzer und andererseits die Programmierer, die nach einem kommen, im Blick hat."

[Expertinnen-Interviews\EI05; Position: 30 - 30]

3.

"Das Programm sollte einen guten Stil haben, gut lesbar, wartbar und übersichtlich sein, sodass andere damit arbeiten können."

[Expertinnen-Interviews\EI06; Position: 3 - 3]

4.4. Analysieren

4.4.1. gegebene Problemstellungen in einzelne Teile zerlegen

1.

"Um vom Problem zum Programm zu gelangen muss ein in Worten beschriebenes abstraktes Problem auf präzise Einzelanweisungen heruntergebrochen werden."

[Expertinnen-Interviews\EI00; Position: 83 - 83]

2.

"Das Analysieren von Problemen und Herunterbrechen auf präzise Einzelanweisungen ist ein NP-hartes Problem und schwierig. Und das ist eine Art von Denken, die man auch normalerweise nicht lernt."

[Expertinnen-Interviews\EI00; Position: 84 - 84]

3.

"Zu Beginn überlegt man, was alles zum Fußballspiel dazugehört, ohne diese Machbarkeits-Kriterien zu berücksichtigen. Und dann kann man sich einen Kriterienkatalog überlegen, wie zum Beispiel: Machbarkeit, Zeitaufwand, Personalmöglichkeiten, Geldmöglichkeiten."

[Expertinnen-Interviews\EI02; Position: 8 - 8]

4.

"Das Problem ist, wenn sie jetzt eine Struktur haben, was haben Sie dann eigentlich erstmal gemacht? Sie haben eigentlich erstmal nur ein größeres Problem in kleinere Teile zerlegt. Sie haben dieses komplexe Problem Fußballspiel, und haben jetzt ein Element, den Fußballspieler, und müssen sich jetzt auf den konzentrieren. Aber das ist nur ein Teil von dem ganzen großen Problem. Das heißt, Sie haben dieses Teilproblem."

[Expertinnen-Interviews\EI02; Position: 13 - 13]

5.

"B: Ja das Feedback wirkt sich aus auf die Gliederung von komplexen Algorithmen und komplexen Programmen, das aus Teilproblemen besteht, und in dem Bausteine miteinander kommunizieren müssen. Das Problem zerlegen die Studierenden in Teilprobleme, da sieht man die Fortschritte."

[Expertinnen-Interviews\EI02; Position: 57 - 57]

6.

"Und was sichtbar ist, wenn Studierende Sachverhalte erkennen. Das hat auch nichts mit Programmieren zu tun, wenn ich weiß, dass Tag X vor Tag Y ist. Dann kann ich auch rechnen, dass ein Tag nach einem ist. Das ist die gleiche Aufgabe, ich muss nur die beiden Tage tauschen."

[Expertinnen-Interviews\EI03; Position: 25 - 25]

7.

"B: Es geht natürlich auch um das elementare Programmieren. Das heißt, das Zerlegen von Problemen in Einheiten, die man direkt in die Programmiersprache abbilden kann."

[Expertinnen-Interviews\EI04; Position: 20 - 20]

8.

"Und dazu müssen sie das Problem analysieren können."

[Expertinnen-Interviews\EI04; Position: 22 - 22]

9.

"Das zweite ist ein Schritt, den viele Leute unterschätzen. Der zweite Schritt ist dieses Aufbrechen des Problems in Einheiten, die ich dann direkt programmieren kann. Also dieses Zerlegen des Problems."

[Expertinnen-Interviews\EI04; Position: 34 - 34]

10.

"Und Aufgabenstellung so herunter zu brechen, wie es für die aktuelle Problemstellung nötig ist."

[Expertinnen-Interviews\EI05; Position: 6 - 6]

11.

"B: Programmieren bedeutet ein Problem zu lesen."

[Expertinnen-Interviews\EI06; Position: 1 - 1]

4.4.2. Compiler & Interpreter-Meldungen verstehen

1.

"Manche Studierende sitzen da und verstehen einfach die Compiler-Meldungen nicht. Und dann reichen Hinweise zur Doppelklicken auf den Fehler und zum Anschauen der betreffenden Zeile."

[Expertinnen-Interviews\EI01; Position: 52 - 52]

2.

"Vorher braucht man gar nicht über den Inhalt nachdenken, wenn es noch daran liegt, dass eine Klammer nicht zugeht. Aber das sagt einem eigentlich schon der Compiler. Ich sage den Studierenden, dass der Compiler hilfreich ist und wo man nachschauen muss, in welcher Zeile, an welcher Stelle, bei welcher Klammer, bei welchem Integer oder String."

[Expertinnen-Interviews\EI03; Position: 71 - 71]

3.

"Ich versuche auch beizubringen, dass man das Feedback vom Compiler lesen kann."

[Expertinnen-Interviews\EI03; Position: 74 - 74]

4.

"B: Der Interpreter sagt einfach, etwas ist falsch. Man muss erstmal lernen, seine Ausgabe zu interpretieren. Er sagt ja in der Regel ein bisschen mehr, aber das kommt bei vielen Leuten, wenn sie das noch nicht gelernt haben, diese Fehlerausgabe zu interpretieren, nicht an. Sie sehen nur, dass etwas falsch ist."

[Expertinnen-Interviews\EI04; Position: 51 - 51]

5.

"Also das kann auch nicht das Hauptproblem der professionellen Produkte sein, der Interpreter oder Compiler. Die sagen, was falsch ist, aber die erwarten schon, dass man gelernt hat, mit ihnen umzugehen. Und die erwarten schon, dass man die Sprache im Grunde genommen wirklich kennt, und helfen dann den Experten weiter."

[Expertinnen-Interviews\EI04; Position: 79 - 79]

4.4.3. eigenen Code erklären können

1.

"Sichtbar wird es, wenn Studierende bei der Abgabe Code vorführen sollen, den sie gemacht haben. Wenn sie nicht wissen, was die einzelnen Zeilen machen, ist das ein Problem."

[Expertinnen-Interviews\EI01; Position: 3 - 3]

2.

"B: Wer Sachverhalte über Programmierung ausformuliert aufschreiben kann, kann das mit Sicherheit auch programmieren. Weil er zeigt mit dieser Kompetenz, dass er weiß, wie es geht. Er kann das, indem er auf der Metaebene was darüber aufschreibt. Diejenigen mit einer hohen Punktzahl bei dieser Aufgabe haben bestanden, das weiß ich."

[Expertinnen-Interviews\EI03; Position: 93 - 93]

3.

"Es gibt manche, die haben da wenig gemacht, die können irgendwie intuitiv schon programmieren, und die kommen dann auf eine 3, oder 2 Minus. Aber die können noch nicht darüber reden. Es könnte auch sein, dass es bei manchen an Sprache und Motivation liegt. Aber viele, die da einen oder null Punkte von zwanzig haben in der Klausur, da bin ich ziemlich sicher, dass die Programme auch nichts sind."

[Expertinnen-Interviews\EI03; Position: 94 - 94]

4.

"B: Bei den Studierenden lässt sich an der Lösung und an der Präsentation erkennen, ob sie programmieren können. Effizient schaue ich mir nicht die Programme an. Ich glaube das ist

so vielfältig und das ist so eine Art Prüfungssituation. Es geht um Vorstellungen, die man hat und ich glaube die Vorstellungen kommen in so einer Situation durch die Präsentation gut raus und man kann dann probieren, die in brauchbarere Vorstellungen umzuwandeln, wenn man denkt, dass die ungünstig sind. Früher hat man von Fehlvorstellungen geredet. Das macht man ja heute nicht mehr."

[Expertinnen-Interviews\EI05; Position: 4 - 4]

4.4.4. fremden Code lesen können

1.

"Codeverständnis wird stark abgeprüft."

[Expertinnen-Interviews\EI00; Position: 15 - 15]

4.4.5. Ausgabe von fremdem Code bestimmen

1.

"Lernende sollen Programme verstehen, indem sie erkennen, was ein fremder Code ausgibt."

[Expertinnen-Interviews\EI00; Position: 3 - 3]

4.5. Bewerten

4.5.1. Suchen von Fehlern in Programmen

1.

"Auch Fehler suchen lernt man nicht. Was macht man, wenn das Programm nicht läuft? Ich würde sagen benutze Print-Anweisungen um zu sehen, bis wohin es läuft und dann schaue nach, ob die Variablen alle stimmen."

[Expertinnen-Interviews\EI00; Position: 85 - 85]

2.

"Aber das Suchen von Fehlern im Programm ist etwas, was man nicht a priori weiß, das muss man einmal gesagt bekommen oder einmal gemacht haben."

[Expertinnen-Interviews\EI00; Position: 86 - 86]

3.

"Andere Studierende sitzen an irgendeinem Fehler und finden ihn einfach nicht und eigentlich sieht man da meistens sofort, da ist irgendwie der Ausdruck in der Bedingung vertauscht, oder es ist ein Syntaxfehler. Bevor die Studierenden an so einer Stelle ewig hängen, weil zugegebenermaßen gerade in C, oder C++ sieht man es erst mit ein bisschen Erfahrung schneller. Und das ist dann so frustrierend. Da sage ich dann doch lieber mal Bescheid. Und dann sehen die Studierenden, dass der Fehler unbedeutend war und es geht weiter."

[Expertinnen-Interviews\EI01; Position: 53 - 53]

4.

"Dann geht es darum, es soll 42 rauskommen, und bei den Studierenden kommt siebzehn raus. Hier Da kann man selber über die Ursachen nachdenken. Meistens hilft das schon."

[Expertinnen-Interviews\EI03; Position: 75 - 75]

4.5.2. Angemessenes Vorgehen zur Problemlösung auswählen

1.

"Die Studierende müssen selbst entscheiden, wie sie genau vorgehen, um ein Problem zu lösen. Der Lösungsweg ist nicht entscheidend, solange das Problem gelöst wird."

[Expertinnen-Interviews\EI00; Position: 17 - 17]

2.

"Und das andere ist, wenn es jetzt wirklich um das Programmieren geht, dass man wesentlich mehr Dinge weiß. Man hat ein Problem, und kennt genau die Vor- und die Nachteile. Und Programmieren ist eine Folge von Kompromissen. Die perfekte Lösung gibt es normalerweise nicht. Sie müssen immer Vor- und Nachteile abwägen und am Schluss einen Kompromiss schließen. Und zwar am besten den, der die meisten Vorteile bringt, und negativ ausgedrückt, der Ihnen die wenigsten Nachteile liefert. Und je komplexer eine Software wird, desto mehr Kompromisse haben Sie darin und desto mehr haben Sie manchmal kein gutes Gefühl mehr."

[Expertinnen-Interviews\EI02; Position: 43 - 43]

4.5.3. Angemessenheit programmiersprachlicher Lösungen beurteilen

1.

"Und die Umsetzung von abstrakten Dingen im Software-Entwurf zu hinterfragen, ist viel Wert."

[Expertinnen-Interviews\EI02; Position: 60 - 60]

2.

"Man muss die Konstrukte der Programmiersprache kennen und die passenden einsetzen."

[Expertinnen-Interviews\EI06; Position: 5 - 5]

4.5.4. Testen von Programmen

1.

"Und Studierende müssen den Test leisten können."

[Expertinnen-Interviews\EI04; Position: 25 - 25]

4.5.5. Verantwortung übernehmen für Lernerfolg

1.

"Die Studierenden sind der Meinung, es ist die Verantwortung der Lehrperson, alles zu erklären was sie nicht verstehen."

[Expertinnen-Interviews\EI00; Position: 30 - 30]

2.

"Anstatt Literatur zu bearbeiten, weisen die Studierenden die Schuld der Lehrperson zu. Die Verantwortung, aktiv zu werden, verstehen viele im ersten Semester nicht."

[Expertinnen-Interviews\EI00; Position: 31 - 31]

3.

"Zu dieser Haltung und Einstellung zur fehlenden Selbstverantwortung beim Lernen muss Feedback gegeben werden, unter Umständen in Form der nicht bestandenen Klausur."

[Expertinnen-Interviews\EI00; Position: 32 - 32]

4.

"An den Hochschulen besteht unabhängig von der Programmierung das große Problem, dass viele Studierende im ersten Semester dieses universitäre Arbeiten noch nicht verinnerlicht haben. In der Schule habe ich etwas nicht verstanden, weil der Lehrer es nicht ordentlich erklärt hat. Wenn ich etwas an der Universität nicht verstehe, dann hätte ich selbst nachschauen müssen. Diese Beweislastumkehr haben viele noch nicht richtig verstanden."

[Expertinnen-Interviews\EI00; Position: 33 - 33]

5.

"Diejenigen, die schon eine Lehre gemacht haben, haben schon die Einstellung, nicht Kunde zu sein. Sie treten stattdessen in Vorleistung und haben diese Haltung verinnerlicht."

[Expertinnen-Interviews\EI00; Position: 39 - 39]

6.

"Studierende, die direkt nach dem Abitur mit dem Studium beginnen, haben diese Einstellung, nicht mehr Kunde zu sein, noch nicht."

[Expertinnen-Interviews\EI00; Position: 40 - 40]

7.

"Und ich bin auch immer der Meinung, es gibt so ein paar einschneidende Erlebnisse. Wir haben zum Beispiel Studierende, die haben Kinder. Wenn man ein Kind hat, muss man mit ganz anderen Verantwortungen darangehen. Und das merkt man im Studium. Die haben eine andere Herangehensweise. Die wissen auch, dass es wichtigere Schwerpunkte gibt, als nur zu

zocken, oder Spaß zu haben. Und das ist ein permanenter Lernprozess. Also die Hoffnung muss man da nie aufgeben."

[Expertinnen-Interviews\EI02; Position: 30 - 30]

8.

"Normalerweise nehmen diejenigen, die Fragen stellen, die Hilfe ernst. In der Regel sieht man das in den nächsten Übungen. Eine zweite Lösung nach der Hilfestellung ist dann schon besser, aber es muss noch viel gemacht werden."

[Expertinnen-Interviews\EI02; Position: 55 - 55]

9.

"B: Ja die Leute haben das Selbststudium nicht gelernt."

[Expertinnen-Interviews\EI04; Position: 55 - 55]

10.

"An der Universität ist das Studium etwas Individuelles und kein verschultes System. Im Prinzip sind alle für sich selber verantwortlich. Man kann zwar Hilfestellung geben und sagen, aber letztlich ist jeder seines Glückes Schmied und ich zwinge niemanden, was abzugeben."

[Expertinnen-Interviews\EI05; Position: 59 - 59]

4.5.6. Externe Ressourcen zum Lernen nutzen

1.

"B: Es gibt auch Studenten, da merkt man, die fangen überhaupt erst an, versuchen es aber trotzdem. Sie haben dann viele Zwischenfragen, nutzen ein Fachbuch zum Thema."

[Expertinnen-Interviews\EI01; Position: 10 - 10]

2.

"Es macht mehr Sinn macht, erst mal lange und gründlich zu recherchieren und von anderen zu lernen."

[Expertinnen-Interviews\EI02; Position: 27 - 27]

3.

"B: Am schnellsten lernt man Programmieren, wenn man sich Code von anderen anschaut. Das ist sehr wichtig. Weil nur, wenn man sich Sachen von anderen anguckt, bekommt man neuen Input. Ansonsten bewegt man sich immer in seinen eigenen Bahnen. Man hat ein Problem, man stellt sich die Lösung dafür vor, sieht die Lösung von jemand anderem und merkt daran, was jemand anders besser gemacht hat. Davon lernt man sehr viel."

[Expertinnen-Interviews\EI02; Position: 40 - 40]

4.

"B: Beim Quellcode von Kommilitonen ist Vorsicht geboten. Das kann gut gehen, wenn Studierende wirklich wissen, was Sie tun. Beim Internet als Quelle müssen die Quellen ganz genau angeschaut werden. Es gibt ein paar gute Bücher, wo man sich sowas anschauen kann. Und dann, muss man eigentlich wirklich gezielt im Internet suchen. Beziehungsweise was es auch gibt, sind Open Source Bibliotheken und Produkte. Und da gibt es einige im Programmierbereich, die sind richtig gut, wie zum Beispiel die Boost Library im C++ Bereich. Das ist oft gereviewt worden, weltweit von vielen Leuten."

[Expertinnen-Interviews\EI02; Position: 41 - 41]

5.

"Ich mache das Internet auf zum Programmieren und da gibt es so viel Information. Und es gibt vielleicht Informationen, die den Anfänger verwirren. Wenn Studierende versuchen, sich Inhalte über andere Quellen anzueignen, anstatt mal meinem Kurs zu folgen, kann das problematisch werden."

[Expertinnen-Interviews\EI03; Position: 52 - 52]

6.

"Die Studierenden sollten sich eher meinen Kurs, mein Skript anschauen, nicht weil es besser ist, sondern weil ich zum Schluss prüfe. Und alles, was ich prüfe, steht in meinem Material. In den anderen Dingern steht ganz viel Information, ganz viel wichtige und gute Information. Aber die wird im ersten Semester verwirren."

[Expertinnen-Interviews\EI03; Position: 53 - 53]

7.

"Und das sieht man manchmal, wenn jemand anders lernt, oder versucht mit anderen Unterlagen zu lernen. Da muss er sehr gut sein, wenn er das kann, um das zu verstehen. Die Terminologie ist nicht einheitlich zwischen der Literatur. Auf Webseiten steht, wie man es machen kann, aber der Kontext oder Spezialfälle fehlen. Oder ich habe gerade erst mal dieses eine Konzept beigebracht. Also das ist hinderlich, was ich manchmal beobachte. Das ist auch nicht falsch, aber es passt gerade nicht zu dem, was sie gerade lernen sollen. Nach zwei Semestern, drei Semestern sollte sich das einspielen. Aber im ersten Semester kann das hinderlich sein."

[Expertinnen-Interviews\EI03; Position: 54 - 54]

8.

"Viele Sachen sind gar nicht mehr im Lehrbuchstil verfügbar, sondern nur noch im Manual-Stil. Also in einer Sprachbeschreibung, die ganz anders strukturiert ist, viel schwieriger zu verstehen ist. Es werden Details erklärt, die Studierende überhaupt nicht verstehen, und noch nicht verstehen können, weil es kein Lehrbuch ist, das systematisch aufeinander aufbaut. Und trotz alledem müssen Studierende damit umgehen lernen."

[Expertinnen-Interviews\EI04; Position: 56 - 56]

9.

"Man zeigt noch andere Möglichkeiten auf und nennt ein Lehrbuch, Literatur, etc."

[Expertinnen-Interviews\EI05; Position: 64 - 64]

4.5.7. Selbstreflexion

1.

"B: Es gibt Studenten, die denken sie können es, weil sie Vorkenntnisse aus der Fachoberschule haben. Er hat dann aber festgestellt, er kann es doch nicht. Er hat sich dann hingesetzt und dann hat es geklappt. Es kommt auf den Einzelfall an."

[Expertinnen-Interviews\EI01; Position: 15 - 15]

2.

"Wenn man selbst einschätzt, inwieweit man selber programmieren kann, ist das an der Stelle auch noch wichtig, gerade in den frühen Semestern."

[Expertinnen-Interviews\EI01; Position: 64 - 64]

3.

"B: Es hat sehr viel mit Charakter zu tun, und damit, seine eigenen Grenzen zu erkennen. Irgendwann muss man lernen, oder akzeptieren, dass es auch andere Leute gibt, die das vielleicht viel toller können, als man das selbst kann und dass man von denen viel lernen kann. Und das Entscheidende ist, nicht gleich loszulegen und irgendetwas zu versuchen, weil man denkt, man kann alles."

[Expertinnen-Interviews\EI02; Position: 26 - 26]

4.

"Und das meine ich mit eigene Grenzen kennen, das Rad nicht tausend Mal neu erfinden, sondern von anderen lernen."

[Expertinnen-Interviews\EI02; Position: 28 - 28]

5.

"B: Man darf nicht vergessen, dass die Studierenden junge Menschen sind. Es gibt Studierende, die sind so alt wie ich, das kann passieren. Das sind aber Ausnahmen. Es gibt so Menschen, die bauen sich ein Eigenuniversum auf, das sie niemals durchbrechen werden. Die haben eine Selbstwahrnehmung, die dem objektiven Eindruck entgegenspricht. Aber bei den meisten, das ist einfach eine Erfahrung, die sie machen müssen, Und wenn die diese Erfahrung mal gemacht haben, dann klappt das auch."

[Expertinnen-Interviews\EI02; Position: 29 - 29]

6.

"Das darüber nachdenken, was ich da eigentlich mache, also oder Reflektieren ist bei uns oft der Fall. Das muss ich begleiten in der Vorlesung. Aber wir haben dann zum Beispiel auch eine Vorlesung, die das dediziert zum Thema hat. In Algorithmen und Datenstrukturen ist das dediziert zum Thema. Dann über Aufwandsabschätzungen, O-Notation nachzudenken. Die Software-Technik hat das Thema, wie man es überhaupt, wie macht man das in Projekten macht, wie man es dokumentiert."

[Expertinnen-Interviews\EI03; Position: 14 - 14]

4.6. Erzeugen

4.6.1. Programmiersprachliche, lauffähige Lösungen für Probleme schreiben

1.

"Wenn jemand programmieren kann, versteht er zwar Programme, kann aber auch selber Programme schreiben."

[Expertinnen-Interviews\EI00; Position: 4 - 4]

2.

"B: Die Kompetenz, selber Programme zu schreiben, soll gefördert werden."

[Expertinnen-Interviews\EI00; Position: 10 - 10]

3.

"Das selbstständige, aktive Schreiben von Code ohne genaue Mustervorgaben fällt den Studierenden schwer."

[Expertinnen-Interviews\EI00; Position: 24 - 24]

4.

"Aber wenn man das Programmieren nicht prüft, bildet man die Leute nicht richtig aus. Die Leute müssen selber vom Problem ausgehend ein Programm schreiben. In der Praxis ist auch niemand, der die Main-Methode vorschreibt."

[Expertinnen-Interviews\EI00; Position: 66 - 66]

5.

"Wer programmieren kann, da sieht man den Prozess im Praktikum. Die Studierenden sind in der Lage, während der Übungsstunde ein funktionierendes Programm fertigzustellen."

[Expertinnen-Interviews\EI01; Position: 4 - 4]

6.

"B: Für viele, die neu anfangen, ist das Formale eine Schwierigkeit. Die Studierenden haben vielleicht eine grobe Idee, wie sie Probleme lösen. In der Vorlesung wurde die Lösung in natürlicher Sprache gemeinsam an der Tafel besprochen. Dabei wurde ungefähr verstanden, worum es geht und was die Idee ist. Aber wenn das dann formal logisch in eine Programmiersprache gefasst werden soll, ist das sehr schwer."

[Expertinnen-Interviews\EI01; Position: 18 - 18]

7.

"B: Also programmieren heißt ja erst mal ein Programm zu schreiben, das überhaupt übersetzt und läuft. Das ist schon eine Leistung."

[Expertinnen-Interviews\EI03; Position: 70 - 70]

8.

"B: Kenntnis allein reicht natürlich nicht aus. Was sie lernen sollen, ist Problemlösungen mit der Programmiersprache."

[Expertinnen-Interviews\EI04; Position: 21 - 21]

9.

"Dann müssen die Studierenden diese Umsetzung leisten können."

[Expertinnen-Interviews\EI04; Position: 24 - 24]

10.

"Also das ist als Ziel, Programmieren als Mittel zur Problemlösung kennenzulernen."

[Expertinnen-Interviews\EI04; Position: 26 - 26]

11.

"Es geht um den Schritt, ein Problem umzusetzen in eine Lösung durch ein Programm. Und auch zu lernen, was geht und was geht nicht."

[Expertinnen-Interviews\EI04; Position: 37 - 37]

12.

"B: Für mich heißt programmieren sich strukturiert über Abläufe Gedanken machen zu können. Sich dort insbesondere bewusst zu werden über bestimmte Modelle von Strukturen wie Schleifen, bedingte Anweisungen. Dementsprechend muss man einen ganzen Prozess mit seinen Möglichkeiten im Blick haben und das in einer formalisierten Art und Weise in (ggf. grafischen) Programmiersprachen ausdrücken zu können."

[Expertinnen-Interviews\EI05; Position: 1 - 1]

13.

"B: Für mich wird Programmieren erkennbar, indem Probleme, die im akademischen Bereich manchmal künstlich sind, die aber auch immer eine Anlehnung an echte Aufgabenstellungen haben oder auch echte Aufgabenstellungen sind, in einer qualitativ hochwertigen Art durch Programme gelöst werden."

[Expertinnen-Interviews\EI05; Position: 2 - 2]

14.

"Problemlösungen muss man dann so modellieren, dass es die Maschine versteht. Das Modellieren kommt also am Anfang und Ende."

[Expertinnen-Interviews\EI05; Position: 8 - 8]

15.

"Und natürlich, die grundsätzliche Kompetenz ist, das Problem zu lösen. Das heißt natürlich nicht, dass die sich dafür lange damit beschäftigt haben."

[Expertinnen-Interviews\EI05; Position: 20 - 20]

16.

"B: Also gute Programmierer erkennt man an den Problemlösungen."

[Expertinnen-Interviews\EI05; Position: 25 - 25]

17.

"Zu dem Problem wird ein funktionales Programm geschrieben, das getestet ist."

[Expertinnen-Interviews\EI06; Position: 2 - 2]

4.6.2. Logische Ausdrücke und Operatoren zur Problemlösung benutzen

1.

"B: Logisches Denken ist schwierig. Bei der Aufgabe, alle Elemente eines Arrays um eine Position nach links zu verrutschen, entstehen zahlreiche Arbeitsschritte und viele einzelne Operationen."

[Expertinnen-Interviews\EI00; Position: 81 - 81]

2.

"Studierende, die wenig mit Mathematik zu tun hatten, haben es schwerer, zum Beispiel bei Bedingungen und deren Reihenfolge. Viele denken, die Reihenfolge ist völlig gleichgültig - ist sie natürlich nicht. Man muss das aber formaler ausdrücken zu können. An der Stelle würde es helfen, wenn die Mathematik nicht so stark vernachlässigt wird, dann könnte die Logik, die hinter Programmiersprachen und etwa Bedingungen formuliert wird, durchschaubarer werden für Studierende."

[Expertinnen-Interviews\EI01; Position: 19 - 19]

3.

"Man muss mehr im mathematischen Formel-Kalkül denken. Es ist eine Sprache, aber anders als Englisch ist es doch sehr formal geprägt."

[Expertinnen-Interviews\EI01; Position: 66 - 66]

4.

"Das ist generell das mathematische Denken an der Stelle. Es müssen nicht Differenzialgleichungen hoch und runter sein. Aber ein bisschen das analytische Denken muss geschult werden. Und das ist das, was Programmieren vom Rest unterscheidet."

[Expertinnen-Interviews\EI01; Position: 68 - 68]

5.

"Und das zweite ist logisches Denkvermögen."

[Expertinnen-Interviews\EI02; Position: 2 - 2]

6.

"Und zum Teil bei den schwersten Fälle, ist es das logische Denkvermögen. Wenn das logische Denkvermögen fehlt, können wir hier nicht mehr viel machen. Das ist in meinen Augen die grundsätzliche Voraussetzung."

[Expertinnen-Interviews\EI02; Position: 18 - 18]

7.

"Zu dem logischen Denkvermögen nochmal. Während meines Informatik-Studiums war die Informatik noch ziemlich jung, und die Professoren waren fast alle Mathematiker. Das war schwierig für mich, weil ich schon mit Programmiererfahrung hingekommen bin und gedacht habe, ich kann schon programmieren. Aus dem Studium habe ich inhaltlich nichts gebraucht. Aber in den Fächern Elektrotecknik, Mathematik und Informatik ging es nur im Denken und nichts Anderes. Und wenn Sie denken können, dann können Sie sich nach dem Studium an die speziellen Gegebenheiten in ihrem Job anpassen."

[Expertinnen-Interviews\EI02; Position: 20 - 20]

8.

"B: Das logische Denken kann man erlernen. Das Gehirn ist definitiv trainierbar, auf jeden Fall."

[Expertinnen-Interviews\EI02; Position: 21 - 21]

9.

"Um die Kompetenz der Studierenden zu erkennen, dazu gibt es eine Art Lakmus-Test. Ich lasse auf Zeit und Wochentage programmieren. Dann lasse ich einen Vergleichsalgorithmus

programmieren in den ersten paar Wochen, der zeigen soll, was es heißt, wenn ein Zeitpunkt früher ist, als der andere. Das ist eine Kleinigkeit."

[Expertinnen-Interviews\EI03; Position: 3 - 3]

4.6.3. Software(-projekte) strukturiert entwerfen

1.

"Das ganze Projekt muss richtig partitioniert werden, es müssen Files angelegt werden. Es muss bei einem Projekt überlegt werden, was in welcher Klasse vorkommt, um das Programm laufen zu lassen."

[Expertinnen-Interviews\EI00; Position: 9 - 9]

2.

"Aus dem vorgegebenen Problem entwickeln die Studierenden die Programm-Struktur. Das wäre die höchste Form des Programmierens."

[Expertinnen-Interviews\EI00; Position: 18 - 18]

3.

"Dann kann man die Dinge sortieren. Die Fußballspielerin wird als sprachliches Konstrukt eine Klasse. Die Verben werden zu Methoden, und alles was so Substantive sind, werden Attribute. Das heißt, Sie können sich, rein sprachlich an Ihren Lösungseinsatz annähernd und Software von vornherein strukturieren, ohne überhaupt irgendwas von Programmierung verstanden zu haben."

[Expertinnen-Interviews\EI02; Position: 9 - 9]

4.

"Studierende müssen vorm Programmieren Ihre Software schon gliedern, indem Sie sich einfach nur an der Realität orientieren und das sprachlich zerlegen."

[Expertinnen-Interviews\EI02; Position: 10 - 10]

5.

"Was Studierenden fehlt, ist das Gefühl, wie kann man eine Idee repräsentieren. Also zum Beispiel, will man einen Pass von A nach B spielen, muss man zuerst repräsentieren, dass ein Spieler einen Ball hat. Da gibt es viele ineffiziente Lösungsmöglichkeiten. Aber auf solche Ideen kommen die Studierenden. Es ist grundsätzlich in vielen Fällen nicht falsch, aber es ist umständlich."

[Expertinnen-Interviews\EI02; Position: 19 - 19]

6.

"B: Der große Vorteil von Objektorientierung ist die Realität als Vorbild. Die Studierenden können die Realität als Vorbild nehmen und daran ihren System-Entwurf machen. Der

System-Entwurf klappt nicht bei Dingen, die da sind, Sie aber nicht sehen können. Und das abstrakte in einem Software-Entwurf zu integrieren ist ein bisschen schwierig."

[Expertinnen-Interviews\EI02; Position: 58 - 58]

7.

"Das Problem muss aufbereitet werden, sodass es in eine Programmierung umsetzbar ist."

[Expertinnen-Interviews\EI04; Position: 23 - 23]

4.6.4. Selbstständig Methoden schreiben

1.

"Es ist ausreichend, wenn die Studierenden die richtige Methode schreiben, die eine Array-Liste erzeugt und zurückgibt. Dazu muss die Methodendefinition erfolgen mit korrekten Parametern und Rückgabewert."

[Expertinnen-Interviews\EI00; Position: 14 - 14]

2.

"Die Studierenden sollen eine Methodensignatur schreiben und diese mit Code befüllen."

[Expertinnen-Interviews\EI00; Position: 20 - 20]

3.

"Im Sinne der Prüfung ist es schon in Ordnung, wenn die Studierenden die richtige Methodensignatur schreiben können und die Methode richtig mit Code ausfüllen."

[Expertinnen-Interviews\EI00; Position: 23 - 23]

4.

"Was unsere Studierende eher als Problem haben, ist was machen sie in dem Fußballspieler? Das Programmieren der einzelnen Methoden ist schwierig. Das ist dann der Kern, diese Methoden mit Leben zu füllen."

[Expertinnen-Interviews\EI02; Position: 14 - 14]

4.6.5. Einzelne Klassen schreiben

1.

"Für eine gegebene Problembeschreibung sind die Studierenden in der Lage, eine Klasse zu schreiben."

[Expertinnen-Interviews\EI00; Position: 6 - 6]

2.

"Dabei gebe ich ein Problem und die Studierenden schreiben zumindest eine Klasse dazu."

[Expertinnen-Interviews\EI00; Position: 11 - 11]

3.

"Die Studierenden müssen selten Klassen schreiben."

[Expertinnen-Interviews\EI00; Position: 13 - 13]

4.6.6. Modellieren von Problemen und Prozessen

1.

"B: Als Lehr-/Lernziele werden Modellbildung und Problemlösen angestrebt, das heißt einen Ausschnitt der Welt zu modellieren und sich an Schemata zu orientieren."

[Expertinnen-Interviews\EI05; Position: 5 - 5]

2.

"B: Beim Modellieren werden Prozesse ganz abstrakt heruntergebrochen, indem man ein Schema anwendet."

[Expertinnen-Interviews\EI05; Position: 10 - 10]

3.

"Und dann kann man ein Problem abbilden indem man UML-Notationen nutzt oder wie bei den agilen Prozessen User-Stories baut."

[Expertinnen-Interviews\EI05; Position: 11 - 11]

4.6.7. Vorgegebenen Programmcode erweitern

1.

"Auf eine genaue Vorgabe hin ergänzen die Studierenden Code in einer Methode, damit diese eine bestimmte Sache ausführt."

[Expertinnen-Interviews\EI00; Position: 5 - 5]

2.

"Für die Aufgaben haben die Studierenden erstmal den Code verstehen müssen und das Feature an der richtigen Stelle einbauen müssen."

[Expertinnen-Interviews\EI01; Position: 41 - 41]

3.

"B: Beim Programmieren gibt es sehr unterschiedliche Aufgabentypen. Eine ganz klassische Fähigkeit ist es, Programme systematisch anpassen zu können. Denken wir mal an das User-Interface, wo ich nicht nur einen neuen Stil reinbringe. Damit meine ich anpassen an das, was schon da ist, das verstehen, konzeptionalisieren, und ergänzen. Das ist so eine Fähigkeit, das

kann man am Anfang nicht und muss sich im Laufe des Studiums entwickeln, dass bestimmte Sachen systematisch weiterentwickelt werden können."

[Expertinnen-Interviews\EI04; Position: 2 - 2]

4.6.8. Programm mit mehreren Klassen schreiben

1.

"Alternativ können die Studierenden ein kleines Projekt mit mehreren Klassen schreiben."

[Expertinnen-Interviews\EI00; Position: 7 - 7]

2.

"Und um noch weiterzugehen, wie das dann vor allem alles zusammenspielt. Also wie die Algorithmen intern zusammenwirken."

[Expertinnen-Interviews\EI02; Position: 15 - 15]

4.6.9. In vorgegebene Methodensignatur Code schreiben

1.

"Lernende sollen anhand einer vorgegebenen Methode mit vorgegebener Signatur, ein Code-Snippet schreiben."

[Expertinnen-Interviews\EI00; Position: 1 - 1]

2.

"Die Studierenden sollen bei vorgegebener Methode mit Header Code schreiben."

[Expertinnen-Interviews\EI00; Position: 19 - 19]

4.6.10. Selbstständig Code schreiben

1.

"Die Studierenden sollen selbst Code erzeugen mit offenen Vorgaben bzw. Aufgabenstellung."

[Expertinnen-Interviews\EI00; Position: 16 - 16]

2.

"Es ist das Ziel, dass Studierende Code selber produzieren, ohne dass alles im Detail vorgegeben wird."

[Expertinnen-Interviews\EI00; Position: 21 - 21]

4.6.11. (Programmier-)Sprachkonstrukte korrekt verwenden

1.

"Und so ein passives Behalten reicht ja da nicht aus, sondern ich muss das ja aktiv als Sprachelement verfügbar haben, insofern hat es natürlich Ähnlichkeiten mit dem Lernen einer natürlichen Sprache."

[Expertinnen-Interviews\EI04; Position: 28 - 28]

2.

"Die Studierenden müssen die Sprachkonstrukte selber anwenden können. Und sie müssen die Programmiersprache aktiv benutzen können."

[Expertinnen-Interviews\EI04; Position: 30 - 30]

4.6.12. Schleifen schreiben

1.

"B: Schleifen schreiben am Anfang kriegen die Studierenden hin."

[Expertinnen-Interviews\EI06; Position: 30 - 30]

4.6.13. Algorithmen mit Hilfe von Entwurfsmustern problemadäquat entwerfen

1.

"B: Das kann man noch herunterbrechen. Beim Problemlösen gibt es die bottom-up, top-down und die greedy und die Varianten. Die können konkretisiert werden, aber den roten Faden zu behalten, ist wichtiger. So wird auf wenige Elemente reduziert, die immer wieder genutzt werden."

[Expertinnen-Interviews\EI05; Position: 9 - 9]

4.6.14. Selbstständig Klasse mit Methode schreiben

1.

"Bei vorgegebenem Problem schreiben die Studierenden selbstständig eine Klasse mit Methode"

[Expertinnen-Interviews\EI00; Position: 2 - 2]

4.6.15. Systematik beim Problemlösen entwickeln

1.

"B: In der Germanistik gibt es auch diese zwei Teile. Es gibt zum einen die Literaturwissenschaft, wo man interpretieren kann und zum anderen die Sprachwissenschaft. Und da gibt es diejenigen, die eher mit der Literatur gut klarkommen. Und dann gibt es die Studierenden, denen Sprachwissenschaft eher lag. Und Letzteres ist die Denkrichtung, die Richtung Programmieren geht. Das sind einfach vielleicht einfach zwei Arten zu Denken. Kann man vielleicht lernen, weiß ich nicht, kenne ich mich nicht genügend aus. Das ist in anderen Bereichen vermutlich ähnlich."

[Expertinnen-Interviews\EI01; Position: 69 - 69]

2.

"B: Programmieren setzt sich aus zwei wesentlichen Dingen zusammen. Das eine ist Intuition. Intuition ist etwas, was sich nur schwer lernen lässt. Das ist der knifflige Teil. Das ist oft so: hat man früh genug damit angefangen, oder nicht. Wenn Sie zum Beispiel als Kind etwas gelernt haben, eine Sprache, ein Musikinstrument, dann ist das so ähnlich. Wenn Sie etwas als Kind so gelernt haben, ist es etwas völlig Selbstverständliches."

[Expertinnen-Interviews\EI02; Position: 1 - 1]

3.

"Das Kernproblem, was die Studierenden bei uns haben, ist von der Aufgabenstellung ausgehend eine Lösung zu finden. Das hat etwas mit Intuition zu tun. Das ist etwas, was sich nur schwer lernen lässt, oder sich im späteren Alter mit viel Aufwand kompensieren lässt."

[Expertinnen-Interviews\EI02; Position: 4 - 4]

4.

"Ich lasse die Studierenden zum Beispiel ein Fußballspiel programmieren, damit sie diese Herangehensweise lernen."

[Expertinnen-Interviews\EI02; Position: 7 - 7]

5.

"B: Das kriegen die Studierenden soweit hin. Da kommen die von selber drauf. Es ist oft intuitiv klar, weil ich setze im dritten, vierten Semester an."

[Expertinnen-Interviews\EI02; Position: 11 - 11]

6.

"B: Das ist immer das Schöne, wenn man sieht, wie Studierende dazulernen. Das wird sichtbar, indem Studierende systematischer an Problemlösungen rangehen. Das ist das Entscheidende, was ich für mich im Studium mitgenommen habe."

[Expertinnen-Interviews\EI02; Position: 22 - 22]

7.

"Und man muss die systematische Vorgehensweise haben, wie man da rangeht."

[Expertinnen-Interviews\EI02; Position: 24 - 24]

8.

"Bei einer Programmieraufgabe ist es ähnlich wie in einer Mathematik-Aufgabe, da gibt es wenige Grauzonen. Entweder jemand macht was, und das kann er auch auf verschiedene

Wege machen. Es gibt da nicht unbedingt den einen Ziel-Weg. Aber es gibt keine Ungewissheit."

[Expertinnen-Interviews\EI02; Position: 37 - 37]

9.

"Man erkennt es, aber das gilt nicht nur für Programmieren. Man erkennt es eher an der Arbeitssystematik. In der ersten Praktikumsstunde im ersten Semester programmiere ich noch nicht, sondern habe Denksportaufgaben dabei, Knobelaufgaben, oder eine Aufgabe aus einem Kinderbuch. Der Punkt ist aber nicht, die Aufgabe sofort zu lösen."

[Expertinnen-Interviews\EI03; Position: 19 - 19]

10.

"Das Hauptproblem, wenn es mit der Programmierung nicht klappt, liegt bei den Studierenden, nicht an der Programmierung, sie würden auch bei Literaturwissenschaft scheitern, oder bei Maschinenbau, weil es an der Beschäftigung mit Problemen, oder der Herangehensweise mit Problemen liegt."

[Expertinnen-Interviews\EI03; Position: 22 - 22]

11.

"B: Interessant ist immer, wenn man zeigen kann, eine individuelle Problemlösung lässt sich verallgemeinern. Weil das ist ja genau das, was wieder am Anfang ansetzt. Das macht die Erfahrung aus, wenn man sagt, Moment, ich habe doch mal ein ähnliches Programm gemacht, wann ist ein Wochentag vor einem anderen. Das geht natürlich mit Uhrzeiten genauso. Das geht aber auch mit Objekten im zweidimensionalen Raum und der Frage, welcher weiter links vom anderen liegt."

[Expertinnen-Interviews\EI03; Position: 67 - 67]

12.

"Das ist auch etwas, was man in der ganzen Softwaretechnik sieht, dass man individuelle Lösungskonzepte, Einzellösungskonzepte verallgemeinert zu allgemeinen Konzepten, mit der man eine ganze Gruppe von Problemen lösen kann."

[Expertinnen-Interviews\EI03; Position: 68 - 68]

13.

"Und das macht dann den Novizen und zum Schluss dann den Experten aus, ob man so seinen Baukasten an Strategien zur Problemlösung hat. Da sollen die Leute ja hinkommen."

[Expertinnen-Interviews\EI03; Position: 69 - 69]

14.

"Meistens hilft es, wenn man sich an einem roten Faden festklammern kann. Bei Überforderung kann man dann Standardschema X ausprobieren. Wenn die eine Möglichkeit schiefgeht, kann man ich das nächste probieren."

[Expertinnen-Interviews\EI05; Position: 18 - 18]

15.

"B: Die Herausforderung in dem Lernprozess ist, das obere Ende zu erreichen, wo Studierende durch sehr einfache Schemata gute Erfolge bei Programmieraufgaben erzielen."

[Expertinnen-Interviews\EI05; Position: 27 - 27]

16.

"B: Neben Offenheit spielt strukturierte Denkfähigkeit eine Rolle."

[Expertinnen-Interviews\EI05; Position: 46 - 46]

17.

"B: Man muss algorithmisch denken, und sich überlegen, was in welcher Reihenfolge gemacht werden muss."

[Expertinnen-Interviews\EI06; Position: 4 - 4]

18.

"Man muss sich den gesamten Ablauf vorher überlegen."

[Expertinnen-Interviews\EI06; Position: 6 - 6]

19.

"Man muss gewohnt sein, an alle Randfälle zu denken, und diese berücksichtigen."

[Expertinnen-Interviews\EI06; Position: 7 - 7]

20.

"Die Studierenden haben mehr oder weniger viel Talent zum Programmieren. Manche haben schon eine Ablauf-orientierte Denkweise."

[Expertinnen-Interviews\EI06; Position: 12 - 12]

21.

"Viele Anfänger haben schon als Kind programmiert, wo man spielerisch anfängt, und das als ganz natürliche Herangehensweise betrachtet. Das ist sicherlich ein Vorteil. Wer eine Ablauf-orientierte Denkweise gewohnt ist, hat es beim Programmieren einfacher."

[Expertinnen-Interviews\EI06; Position: 19 - 19]

22.

"B: Wahrscheinlich ist das eine Denkweise, die in den Leuten drinsteckt oder nicht."

[Expertinnen-Interviews\EI06; Position: 20 - 20]

23.

"B: Weiß ich nicht, ob man diese Ablauf-orientierte Denkweise viel fördern kann. Natürlich indem man als Kind solche Sachen schon spielerisch macht, zum Beispiel mit einem programmierbaren Taschenrechner. Damit haben wir gespielt und in der Schule verbotenerweise einen Wettbewerb daraus gemacht. Damit übt man das in einem Alter, wo man das noch mehr verinnerlicht."

[Expertinnen-Interviews\EI06; Position: 21 - 21]

24.

"Ich nehme an, dass das eher so ein bisschen was angeborenes ist. Oder man übernimmt diese Denkweise von den Eltern. Nicht überall im Leben ist das als sinnige Vorgehensweise zu gebrauchen, wenn man z. Bsp. nicht entscheidungsfreudig ist. Also ich würde da jetzt nicht unbedingt als ist gut oder ist schlecht betrachten."

[Expertinnen-Interviews\EI06; Position: 22 - 22]

25.

"Manche Studierende haben eine Ablauf-orientierte Vorgehensweise/Denkweise. Logistik ist sicherlich ähnlich von der Vorgehensweise."

[Expertinnen-Interviews\EI06; Position: 23 - 23]

26.

"B: Ein Ablauf-orientiertes Vorgehen hilft den Studierenden, programmieren zu lernen."

[Expertinnen-Interviews\EI06; Position: 24 - 24]

4.6.16. selbstständige Organisation des Lernprozesses

1.

"Bei dem Projekt wird selbstständig gearbeitet."

[Expertinnen-Interviews\EI00; Position: 8 - 8]

2.

"B: Wer programmieren kann, versucht selbstständig, ein Ziel zu erreichen, die Aufgabe umzusetzen, ohne ständig zu schauen, die Lösung durch andere Kommilitonen zu erhalten, während man selbst nichts gemacht hat."

[Expertinnen-Interviews\EI01; Position: 2 - 2]

3.

"Wer das macht, versucht es auch selbstständig, evtl. auch durch googeln, auf Stack-Overflow schauen, etc."

[Expertinnen-Interviews\EI01; Position: 5 - 5]

4.

"B: Wenn die Studierenden selber etwas gemacht haben, ist es in Ordnung. Dann haben sie eine Lösung und selber was gemacht."

[Expertinnen-Interviews\EI01; Position: 40 - 40]

5.

"B: Ich habe ziemlich viel nachgedacht über meine Rolle als Vorlesender, weil ich kann nicht jedes Detail der Programmiersprache in der Vorlesung vorstellen. Die Leute müssen einfach lernen, dass sie das sich aus eigener Kraft im Selbststudium teilweise erarbeiten müssen."

[Expertinnen-Interviews\EI04; Position: 88 - 88]

6.

"So und das Selbststudium ist schwer, weil das sind sie zum Beispiel von der Schule in der Regel nicht gewöhnt. Sondern da wird das geprüft, was in der Stunde besprochen wurde und alles andere wird so getan, als wenn es das nicht gibt. Und das ist hier natürlich gänzlich anders. Wichtig ist nicht das, was ich gesagt habe, sondern richtig ist das, was richtig belegbar, beweisbar richtig ist."

[Expertinnen-Interviews\EI04; Position: 89 - 89]

7.

"B: Die Lernerfolge sind völlig individuell. Bei einigen klappt der Transfer und bei anderen nicht. An der Uni hat man nicht den Anspruch, dass man auch die letzten mitnimmt. Manche Studierenden haben einen anderen Zugang und müssen nicht zwangsläufig eine Lernstrategie nutzen."

[Expertinnen-Interviews\EI05; Position: 63 - 63]

8.

"Es ist außerdem eine sehr individuelle Sache, ob man diese eine Art Bloßstellung durch die Präsentation der eigenen Lösung persönlich aushält. Und wer das nicht aushält, hat den Nachteil, dass er das Feedback in dem Moment nicht kriegt. Das muss er sich eben anders holen."

[Expertinnen-Interviews\EI05; Position: 65 - 65]

9.

"B: Diejenigen Studierenden, die immer nachfragen, haben das das Vorgehen nicht verinnerlicht. Zu viel Unterstützung zu geben, ist kontraproduktiv. Studierende müssen sich alleine da durchkämpfen, dann ist der Lerneffekt höher, als mit viel Unterstützung."

[Expertinnen-Interviews\EI06; Position: 46 - 46]

10.

"Andere führen in Dreier-, Vierer-Gruppen ihre eigene Diskussion und fühlen sich genervt, wenn man fragt, ob man helfen kann."

[Expertinnen-Interviews\EI06; Position: 48 - 48]

11.

"Und es gibt einzelne, die alle fünf Minuten fragen. Bei denen ist Hopfen und Malz verloren. Die haben einen falschen Ansatz, wenn sie permanent fragen müssen."

[Expertinnen-Interviews\EI06; Position: 49 - 49]

12.

"Selbstständig denken und rumprobieren ist nützlich beim Programmieren, obwohl ich selbst nicht gerne probiere."

[Expertinnen-Interviews\EI06; Position: 50 - 50]

13.

"Selbstständiges Denken braucht man zum Erfolg."

[Expertinnen-Interviews\EI06; Position: 51 - 51]

14.

"Was behindert, ist Unselbstständigkeit. Man muss selbstständig denken und man darf nicht immerzu andere fragen."

[Expertinnen-Interviews\EI06; Position: 57 - 57]

15.

"B: Ich beobachte manchmal eine Gruppe von Studierenden, die sich während des gesamten Semesters permanent meldet und dann in der Klausur durchfällt. Die sind dann nicht selbstständig. Die selbstständigen sehe ich nicht."

[Expertinnen-Interviews\EI06; Position: 63 - 63]

16.

"Wer am Angang viel Hilfe sucht, tut das in der Regel kontinuierlich. Es kann sein, dass manche Studierende aufgehört haben, oder selbstständig geworden sind. Das kann ich nicht beurteilen."

[Expertinnen-Interviews\EI06; Position: 64 - 64]

17.

"Ich habe eine Studentin im Kopf, die permanent gefragt hat und durch Ausdauer den Bachelor-Abschluss geschafft hat. Aber selbstständig war sie nicht."

[Expertinnen-Interviews\EI06; Position: 65 - 65]

18.

"B: Ein wenig selbstständiger wird man mit zunehmendem Alter. Aber ich glaube es ist relativ fest gegeben, wie selbstständig man ist."

[Expertinnen-Interviews\EI06; Position: 66 - 66]

4.6.17. Abstraktion von Handlungsvorschriften (Rekursion)

1.

"B: Also ein Knackpunkt, der schwer ist, ist vielleicht Rekursion. Wer Rekursion sofort schluckt und versteht, kann programmieren."

[Expertinnen-Interviews\EI03; Position: 29 - 29]

2.

"B: Rekursion enthält die Essenz dessen, was unsere gängige Abstraktion der Programmierung ist. Wer Rekursion verstanden hat, kann Funktionsaufrufe, kennt Funktionen. Wer Rekursion kann, hat zumindest intuitiv, technisch gesehen, das Konzept vom Stack verstanden. Wer Rekursion kann, kann Induktionsbeweise. Vollständige Induktion ist Rekursion in der Mathematik. Wer Rekursion kann, kann Mathematik dazu."

[Expertinnen-Interviews\EI03; Position: 30 - 30]

3.

"B: Man könnte sagen, ein Programm ist diskrete Mathematik in angewandter Form. Typen sind Mengen, Funktionen sind Programme, das Datenmodell ist Mathematik."

[Expertinnen-Interviews\EI03; Position: 31 - 31]

4.

"Mit wenig Zeit komplexe Programme zu schreiben, erfordert abstraktes Denken."

[Expertinnen-Interviews\EI06; Position: 27 - 27]

5.

"Rekursion ist schwierig."

[Expertinnen-Interviews\EI06; Position: 31 - 31]

6.

"Dann kommt Rekursion, wo das, was man hinschreibt nicht mehr direkt am Programm beobachtet werden kann. Damit hat man einen riesen Sprung und braucht dieses mathematische, ähnlich einem Induktionsbeweis. Das ist für viele überhaupt nicht machbar."

[Expertinnen-Interviews\EI06; Position: 33 - 33]

7.

"Rekursion ist das haarige Thema, was in der Vorlesung immer länger behandelt wird, obwohl es eigentlich nicht viel zu sagen gibt. Eigentlich könnte man das in zwanzig Minuten erklären, aber man braucht viele Beispiele. Manche kriegen das nicht hin, weil das ein Sprung ist, etwas anderes hinzuschreiben, als das was in Wirklichkeit abläuft. Einige Studierende empfinden das als zu fremd, unabhängig davon, wie oft man diese Besonderheit erklärt."

[Expertinnen-Interviews\EI06; Position: 36 - 36]

8.

"B: Es fällt den Studierenden schwer, wenn man das, was passiert, nicht sieht."

[Expertinnen-Interviews\EI06; Position: 37 - 37]

9.

"B: Die Studierenden selbst teilen nicht mit, was Ihnen schwer fällt. Sie teilen nur mit, nicht zu wissen, wie etwas funktioniert. Trotz mehrfacher Hinweise können die Rekursionsvorschriften nicht aufgeschrieben werden. Die Studierenden finden das merkwürdig."

[Expertinnen-Interviews\EI06; Position: 38 - 38]

10.

"Alles außer Rekursion kann mit mehr oder weniger viel Übung gemeistert werden, bei dem ein Aha-Erlebnis auftritt, wenn etwas entsprechend funktioniert."

[Expertinnen-Interviews\EI06; Position: 40 - 40]

4.6.18. Transfer

1.

"B: Die Studierenden bekommen genau gesagt, was in den Klausuren drankommt, sie kennen die Aufgabenstellungen, die kommen können. Sie kennen alten Klausuren, auch wenn ich die regelmäßig ändere. Von den Ideen her, ist es doch ähnlich. Wenn man sich vernünftig vorbereitet, kann man die Klausuren ordentlich bestehen."

[Expertinnen-Interviews\EI02; Position: 35 - 35]

2.

"Also es sind reine Programmieraufgabe, die in irgendeiner Form in den Vorlesungen oder in den Übungen gemacht wurden. Wenn Studierende die Übungen, die Vorlesung mitgemacht hat, wenn er sich auf die Klausur vorbereitet hat, dann kann zumindest diese Aufgabenstellung bewältigt werden."

[Expertinnen-Interviews\EI02; Position: 38 - 38]

3.

"Man muss den Transfer schaffen von Übungsaufgabe zu Prüfungsaufgaben. Wobei ich zum Teil konkret Prüfungsaufgaben vorher übe."

[Expertinnen-Interviews\EI02; Position: 39 - 39]

5. Herausforderungen beim Lernen und Lehren von Programmieren

5.1. Herausforderungen Lehrende

5.1.1. (Prüfungs-)Aufgaben konzipieren und Bewerten

5.1.1.1. Menschliche Hilfe in Lernsystem nachbilden

1.

"Es sind eben auch so einfache Sachen, zusätzliche Aufgaben. Aber dabei habe ich echte menschliche Hilfe. Das ist eine Herausforderung, das elektronisch abzubilden."

[Expertinnen-Interviews\EI04; Position: 50 - 50]

5.1.1.2. Angemessene Prüfungsformate auswählen

1.

"Ich glaube es ist schwierig, aktive Programmierkompetenz abzuprüfen und deswegen wird es bisher noch nicht so intensiv gemacht. Wenn das abgeprüft würde, dann würden die Leute selbstverständlich nach dem ersten Mal durchfallen und würden sich selbst überlegen, wie man eine Main-Methode schreibe. Das muss man dann einfach."

[Expertinnen-Interviews\EI00; Position: 56 - 56]

2.

"Es muss eine Möglichkeit geben, die Coding-Kompetenz standardisiert abzuprüfen. Wenn ich das nicht mache, dann werden die Studierenden nie coden. Dann werden sie nie selber Programme schreiben."

[Expertinnen-Interviews\EI00; Position: 63 - 63]

3.

"B: Dieses Benutzen abzuprüfen, ist möglich, aber es ist mehr Aufwand. Technisch ist es mehr Aufwand, denn man muss Klausuren am Rechner schreiben, was rechtlich nicht ganz einfach ist."

[Expertinnen-Interviews\EI00; Position: 65 - 65]

4.

"In der Klausur muss berücksichtigt werden, wie lange die Bearbeitungszeit solcher Aufgaben aussieht. Das kann man nur schätzen."

[Expertinnen-Interviews\EI00; Position: 109 - 109]

5.

"Ich kann nicht sagen, was die Übungsaufgaben für Auswirkungen haben. Ich würde mir wünschen, dass, die Leute aktiv Code schreiben können in den Klausuren. Das habe ich bisher noch nicht abgeprüft. Inzwischen haben die Studierenden verstanden, wie sie die Methodensignatur selber schreiben. In den Übungen sieht man, es bewegt sich etwas. Aber das müsste man weiter auswerten anhand der Klausurergebnisse über einige Semester."

[Expertinnen-Interviews\EI00; Position: 118 - 118]

6.

"B: In der Klausur mache ich Programmieraufgaben, aber auf Papier. Also fünf Aufgaben, vier Programmieraufgaben auf Papier, eine Aufgabe im Freitext bei der man Dinge erklären muss in ganzen deutschen Sätzen mit Punkt und Prädikat, Subjekt, Objekt. Ich habe noch nicht diese ganzen EvaSys oder E-Klausuren geschrieben, keine Multiple Choice-Aufgaben. Bei uns wird von Hand geschrieben und von Hand korrigiert."

[Expertinnen-Interviews\EI03; Position: 90 - 90]

5.1.1.3. Entwicklung angemessener, bewertbarer Aufgaben

1.

"B: Das selbstständige Schreiben von Programmen abzuprüfen, ist fast unmöglich. Ist ein Problem ganz grob vorgeben, existieren fünf oder mehr verschiedene Möglichkeiten, die richtig sind."

[Expertinnen-Interviews\EI00; Position: 22 - 22]

2.

"Das ist auch eine Herausforderung, weil zum Beispiel Vererbung nicht mit Hilfe von CodeRunner Aufgaben abgebildet werden kann. Dazu müsste zuerst eine Technologie vorhanden sein, die tatsächlich per Introspektion oder per Reflection-API die tatsächlichen Anforderungen abprüft. Also etwas, das nicht geprüft wird oder geprüft werden kann, wird nicht gemacht."

[Expertinnen-Interviews\EI00; Position: 51 - 51]

3.

"Die Studierenden müssen überprüft werden hinsichtlich ihrer Lösungen. Studierende erzeugen in manchen Fällen irgendeine Lösung. Statt einer Iteration werden zum Beispiel nacheinander einzelne Werte ausgedruckt. Das muss man sehr aufwendig prüfen."

[Expertinnen-Interviews\EI00; Position: 52 - 52]

4.

"B: In der Schule werden immer isolierte Probleme vorgegeben, die man leicht abprüfen kann. Das hat auch seine Gründe, da geht es um Chancen-Gerechtigkeit in der Klausur - die muss für alle gleich sein. Komplexere Dinge abzuprüfen ist schwieriger und unter Umständen

potenziell ungerecht. Deswegen versuchen Lehrende in der Klausur Dinge zu verlangen, die exakt, leicht und identisch korrigierbar sind."

[Expertinnen-Interviews\EI00; Position: 54 - 54]

5.

"Programme zu korrigieren ist wie einen Aufsatz zu korrigieren. Wenn es fünfzehn verschiedene mögliche Lösungen gibt, die alle zum gleichen Ziel führen, stellt sich die Frage, was ist die richtige und was ist besser, was ist schlechter? Deswegen wird das Konstruieren kreativer, individueller Lösungen an den Schulen, speziell, nicht gelehrt und an den Hochschulen oft auch nicht. Da lernen die Studierenden eher APIs und Code-Snippets auswendig, die sie dann auf das Blatt Papier schreiben müssen."

[Expertinnen-Interviews\EI00; Position: 55 - 55]

6.

"Das ist eine schwierige Aufgabe, Code zu bewerten. Ist der Code syntaktisch korrekt? In CodeRunner muss der Code syntaktisch korrekt sein. Und dann gibt es einen Satz Testfälle, die erfüllt sein müssen. Die muss der Code richtig lösen. Dann gibt es volle Wertung. Das ist ok, wenn die Testfälle gut gestellt sind. Aber partielle Lösungen gibt es zum Beispiel nicht. Also es kann keine Lösung nur teilweise richtig sein. Es gibt entweder richtig oder nicht richtig. Das heißt, man müsste viele einfache Aufgaben stellen, um gerecht Teilpunkte vergeben zu können. Das ist eine Herausforderung, was die Prüfungserstellung angeht."

[Expertinnen-Interviews\EI00; Position: 61 - 61]

7.

"B: Das ist wie bei Multiple Choice-Aufgaben. Es gibt entweder volle Punktzahl oder null Punkte für eine Aufgabe. Es wäre sehr unfair in einer Klausur zwei dieser Aufgaben zu stellen, weil dann ggf. ein Viertel der Studierenden null Punkte in der Klausur hat. Das heißt, es müssen viele Aufgaben erstellt werden, die nur relativ wenige Punkte bringt. Wenn ein Student eine verfehlt, ist die Auswirkung auf Folgeaufgaben geringer. Wenn sich Studierende richtig vorbereitet haben, werden statistisch die meisten dieser Aufgaben richtig gelöst. Wenn die Aufgaben zu einfach sind, können komplexe Sachen nicht mehr abgefragt werden. Da habe ich noch keine gute Lösung."

[Expertinnen-Interviews\EI00; Position: 62 - 62]

8.

"Die Information zur Angemessenheit der Aufgaben ist wichtig, weil ich versuche meine Klausuraufgaben exakt so zu stellen, wie die Übungsaufgaben. Ich verspreche meinen Studierenden, dass es keine Überraschungen in der Klausur gibt. Wer die Übungsaufgaben kann, hat mindestens eine 2, oder wird in jedem Fall bestehen."

[Expertinnen-Interviews\EI00; Position: 98 - 98]

9.

"Deswegen muss ich wissen, sind die Übungsaufgaben adäquat. Können das überhaupt ein paar lösen oder haben alle überhaupt keine Ahnung. Dann muss man vielleicht noch einmal schauen."

[Expertinnen-Interviews\EI00; Position: 100 - 100]

10.

"B: Hauptsächlich meckern die Studierenden, dass die CodeRunner Aufgaben schwierig sind."

[Expertinnen-Interviews\EI00; Position: 117 - 117]

11.

"Bei den Programmen sehe ich meistens direkt, das kann ich durchgehen und abhaken. Oder bei komplexeren Prozeduren, die man schreiben soll, sehe ich direkt, ob es etwas mit Aufgabe zu tun hat. Oder es sind ein paar Fehler drin, aber richtiger Gedankengang. Gibt dann nur einen Punkt Abzug. Die lassen sich einfach korrigieren, weil das sieht man mit einem Blick. Da gibt es nur ein paar Leute, die mal so eine absurde Lösung implementieren, die muss man genauer anschauen, aber bei 90 Prozent sieht man sofort, ob die angestrebte Lösung erreicht wurde oder nicht. Aber das ist meistens relativ einfach zu unterscheiden. Ich korrigiere die Textaufgabe als erste. Und sie ist der beste Indikator für das Gesamtergebnis."

[Expertinnen-Interviews\EI03; Position: 92 - 92]

12.

"Und dann gibt es diese vielen Möglichkeiten, das Problem angehen zu können. Und ich glaube viele Leute sind dann auch erschlagen davon."

[Expertinnen-Interviews\EI04; Position: 35 - 35]

13.

"Also sie haben das Missverständnis, dass Informatik gleich Programmieren ist. Und wenn sie programmieren könnten, könnten sie auch die Informatik. Das stimmt natürlich nicht. Aber, man hat das Gefühl, alle Algorithmen gleich mit zu lernen. Aber darum geht es in der Programmierausbildung gar nicht."

[Expertinnen-Interviews\EI04; Position: 36 - 36]

14.

"Und das heißt, es wird eine Kunst sein, für die verschiedenen Fähigkeitsstände, die richtigen Aufgaben zu stellen."

[Expertinnen-Interviews\EI04; Position: 82 - 82]

15.

"B: Die Studierenden wollten CodeRunner nicht mehr. Aber vielleicht habe ich es auch falsch eingesetzt, oder ich habe es sicher falsch eingesetzt. Also es geht dann gut, wenn ich eine Einheit habe, zum Beispiel ein Modul, das ich schlicht und einfach an seiner Schnittstelle testen kann. Es ist eher schwierig, zum Beispiel, wenn ich interaktive Eingaben zulasse, weil ich dann alle möglichen Fehler vernünftig abdecken muss und auch die, die an der Schnittstelle gemacht werden können."

[Expertinnen-Interviews\EI04; Position: 102 - 102]

16.

"Ich bin zu der Überzeugung gekommen, man darf CodeRunner nicht als Erstes einsetzen. Und dann muss man gut testbare Einheiten haben, und gut testbare Probleme. Es gibt solche, die sind nicht gut testbar, und dann sollte man es auch nicht einsetzen."

[Expertinnen-Interviews\EI04; Position: 103 - 103]

17.

"Und man sollte es nicht unbedingt einsetzen, wenn viel interaktive Eingabe ist, weil man so viele Fälle abtesten muss, die man gar nicht abtesten will, die aber auftreten können."

[Expertinnen-Interviews\EI04; Position: 104 - 104]

18.

"Außerdem ist es methodisch nicht gut, weil ich möchte, dass die Leute kreativ sind und sie sollen ihr eigenes Interface entwickeln. Ich kann es aber nur dann testen, wenn ich es genau vorgegeben habe. Das kann ich natürlich machen, wenn ich ein Modul teste, weil das vollkommen klar ist, dass es genau und präzise den und den Parameter in den und den Schranken vorgeht. Da geht das wunderbar. Aber an einem User-Interface möchte ich, dass die Leute da ihre Kreativität ausleben können."

[Expertinnen-Interviews\EI04; Position: 105 - 105]

19.

"B: Beispiele von Lösungen zu den Aufgaben erhalten die Studierenden eine Woche später. Es ist auch in Ordnung, wenn die Studierenden andere Ansätze haben. Das macht es schwer für die Hilfskräfte. Alle Lösungen die zum gewünschten Ergebnis führen, erhalten die gleiche Punktzahl. Bei Programmieraufgaben schaffen die Hilfskräfte das gut."

[Expertinnen-Interviews\EI06; Position: 83 - 83]

5.1.2. Lernatmosphäre in Präsenzveranstaltungen schaffen

1.

"B: Die Gruppe spielt auch eine Rolle beim Lernen. In den aktiven Gruppen haben die starken die schwachen ein bisschen mitgezogen und jeder wollte programmieren können. Umgekehrt können in anderen Gruppen alle abgeneigt sein. Die gehen dann nicht gerne in die Übung.

Gleiche Lehrveranstaltung, einfach nur eine andere Gruppe. Also ich denke, die Gruppe ist ein riesen Punkt und da sollte man schauen, dass in jeder Gruppe immer ein paar starke Leute sind, die auch andere motivieren können, sodass sie nachziehen wollen."

[Expertinnen-Interviews\EI01; Position: 27 - 27]

2.

"Problem ist dann natürlich, wenn dann eine ganze Gruppe von solchen Leuten zusammensitzt. Dieses Gefühl ist manchmal geprägt durch die Fachschaft. Da sind auch Studenten, die zwar zum Beispiel noch nie in der Mathematik waren, aber den Erstsemestern erzählen, wie schrecklich die Mathematik ist, damit die dann Angst haben und auch nicht in die Mathematik gehen. Das multipliziert sich dann weiter."

[Expertinnen-Interviews\EI01; Position: 31 - 31]

3.

"Hinderlich kann zu viel Gequatsche und Reinreden sein."

[Expertinnen-Interviews\EI03; Position: 51 - 51]

4.

"B: Ganz viele kommen nicht in die Übung, ich würde es auch nicht tun. Die Atmosphäre ist nicht optimal. Der Rechnerpool ist im dunklen Keller, und im Sommer wird es sehr warm. Die Luft ist schlecht. Durch die Rechner hat man einen permanenten Geräuschpegel. Ich würde da auch nicht programmieren wollen. Viele setzen sich lieber daheim in Ruhe an ihren Rechner, das ist effektiver."

[Expertinnen-Interviews\EI06; Position: 53 - 53]

5.

"In den Vorlesungen ist es dann laut und es fallen alle durch. Da kommt nichts Positives bei raus."

[Expertinnen-Interviews\EI06; Position: 92 - 92]

5.1.3. Begrenzte Kapazitäten der Lehrenden für Feedback

1.

"Es gibt auch Wochen, in denen man es kaum schafft, zusätzliches Feedback zu geben."

[Expertinnen-Interviews\EI03; Position: 62 - 62]

2.

"Es gibt natürlich auch Wochen, da habe ich es kaum geschafft, darüber geguckt. Das sage ich in dem Fall auch ehrlich zu den Studenten."

[Expertinnen-Interviews\EI03; Position: 63 - 63]

3.

"Das war auch eine Erleichterung bezüglich des individuellen Durchgehens. Ich musste vorher mit jedem im Gespräch nochmal sicherstellen, dass er das, was er mir zeigt, in seinem Programm auch erklären kann. Und das war auch ein Stressfaktor, wenn ein Studierender das nicht erklären kann, dann muss man hart genug sein und konfrontieren, wenn die Lösung offensichtlich keine Eigenleistung war. Und das strengt einen selber psychologisch an, das macht keinen Spaß. Das kostet Kraft, die man doch lieber in was Inhaltliches stecken kann."

[Expertinnen-Interviews\EI03; Position: 96 - 96]

4.

"B: Für Feedback via E-Mail reicht die Zeit nicht."

[Expertinnen-Interviews\EI05; Position: 54 - 54]

5.1.4. Verhindern von Trittbrettfahrern & Plagiaten

1.

"Also wenn jemand mit einer Aufgabe kommt, die er sowieso nicht selber gemacht hat, und stellt dann irgendwie eine Frage, aus der man rückschließen kann, dass die Aufgabe nicht selbst gemacht wurde, das ist dann teilweise dreist. Dann habe ich auch Leute schon mal sitzen lassen. Aber das ist pro Semester vielleicht einer."

[Expertinnen-Interviews\EI01; Position: 54 - 54]

2.

"Am Anfang besteht die Gefahr, dass sich einzelne schwache Studierende an leistungsstarke ranhängen und abschreiben. Es fallen zu viele Studierende durch die Klausur. Durch Gruppenarbeit sollte man diese Gefahr nicht erhöhen."

[Expertinnen-Interviews\EI06; Position: 69 - 69]

3.

"Jemand ohne Vorwissen soll das nicht ausnutzen in einer Gruppenarbeit, ohne selbst zu lernen."

[Expertinnen-Interviews\EI06; Position: 71 - 71]

5.1.5. begrenzte Kompetenzen von Tutoren

1.

"Und da haben wir einfach die Schwierigkeit, dass das nicht alle gute Lehrer sind. Also es sind alles gute Informatiker, aber nicht alles gute Lehrer. Und, ja, das ist eine Schwierigkeit."

[Expertinnen-Interviews\EI04; Position: 93 - 93]

2.

"Aber da treten dann auch menschliche Probleme auf. Also wer hat Recht und alles was da so auftreten kann im zwischenmenschlichen Bereich, tritt da in den Übungsgruppen auf."

[Expertinnen-Interviews\EI04; Position: 95 - 95]

3.

"Bei Randthemen (wie z. Bsp. Unterschied zwischen Ausdruck und Anweisung, wozu gibt es Datentypen) wird es schwieriger für die Hilfskräfte, weiterzuhelfen."

[Expertinnen-Interviews\EI06; Position: 84 - 84]

5.1.6. Programmiersprachen-unabhängige Konstrukte identifizieren

1.

"B: Programmieren an sich ist eine endliche Menge von Konzepten, die zusammen eine Vielfältigkeit entwickelt, die unendlich ist."

[Expertinnen-Interviews\EI03; Position: 26 - 26]

2.

"Man muss nur eine sehr kleine, endliche Menge von Konzepten unterrichten und die sind eigentlich übertragbar."

[Expertinnen-Interviews\EI03; Position: 27 - 27]

3.

"Das wirkt sich auch auf vieles aus, aber was ich glaube, das, was du versuchst, herauszufinden, was ist eigentlich Kompetenz, die das Programmieren ausmacht und die ist tatsächlich unabhängig von der Programmiersprache und deshalb ist es so schwierig zu sagen was ist schwer, was ist nicht schwer. Es gibt Dinge, die sind in der einen Sprache schwer und in der anderen leicht. Oder kann man in der einen gar nicht richtig erklären, weil es die da so gar nicht gibt. Aber im ersten Semester sind die Programmiersprachen-unabhängigen Konstrukte sind viel wichtiger bei der Programmierkompetenz."

[Expertinnen-Interviews\EI03; Position: 98 - 98]

5.1.7. Theorierelevanz in der Programmierung aufzeigen

1.

"Es ist die gesamte Curriculums-Aufgabe, das auch theoretisch zu analysieren und zu verstehen. Und wir haben tatsächlich die beiden Sachen stark getrennt. Also wir haben im ersten Semester meine Vorlesung zur Programmierung. Diskrete Mathematik, wo man über Mengen, Relationen, Induktionsbeweise, das geht Hand in Hand. Aber wir haben es getrennt im zweiten Semester dann die Algorithmen und Datenstrukturen. Eigentlich gehört es aber zusammen."

[Expertinnen-Interviews\EI03; Position: 33 - 33]

2.

"Wenn ich einen Fachinformatiker ausbilden würde, könnte ich auf viele dieser theoretischen Bereiche verzichten. Weil der vielleicht in seinem Arbeitsumfeld relativ klare Programmier-Aufgaben hat, was er programmieren soll und dann vielleicht diese theoretische Untermauerung nicht braucht, um diese Aufgaben zu lösen. Aber an einer Hochschule erwartet man, dass man Leute rausschickt, die auch ohne klaren Auftrag von oben ein Projekt stemmen können. Die brauchen theoretischen Unterbau, dass sie die nächsten vierzig Jahre da auch bestehen können."

[Expertinnen-Interviews\EI03; Position: 34 - 34]

3.

"An der Fachhochschule haben wir Leute, die hatten eine dreijährige Ausbildung, haben schon Projekte gemacht und sonst was. Denen muss ich ja auch zeigen, dass sie hier wirklich noch was anderes lernen."

[Expertinnen-Interviews\EI03; Position: 35 - 35]

5.2. Herausforderungen Lernende

5.2.1. Informatik-spezifische Strategien erlernen

1.

"Es gibt Leute, die fangen mit Aufgabenlösen gar nicht erst an, weil sie das Gefühl haben, es noch nicht vollständig verstanden zu haben. Also die haben das absolute Bestreben, es vollständig zu verstehen und erst dann würden sie anfangen. Und so lernt man nicht, oder sehr schwer programmieren."

[Expertinnen-Interviews\EI04; Position: 58 - 58]

2.

"Es gibt Leute, die lesen einen Absatz und fangen an zu programmieren und denken, sie kriege das jetzt hin und programmieren oder probieren vergleichsweise unsystematisch herum. Die haben auch extreme Schwierigkeiten. Und leider gibt es in der Informatik beide Gruppen."

[Expertinnen-Interviews\EI04; Position: 59 - 59]

3.

"Und nur der mittlere Teil ist der, der so eine Mischung aus programmieren, also probieren und verstehen versucht, der optimal durchkommt. Aber ist man eine andere Persönlichkeit, habe man Schwierigkeiten, man eckt an. Man kann am Anfang nicht alles verstehen können, das sind viel zu viele Details. Und man kann nicht nur anfangen und probieren, auch das funktioniert nicht."

[Expertinnen-Interviews\EI04; Position: 60 - 60]

4.

"B: Es gibt eine Art von Fähigkeiten, die speziell für die Informatik entwickelt werden, oder abgewöhnt werden müssen. Es ist gar nichts Schlechtes daran, es richtig verstehen zu wollen, bevor man es anwendet. Es ist alles ok, aber so kommt man in der Informatik nicht zum Ziel, weil man damit leben können, dass es Sachen gibt, die haben anderen Leute irgendwie gelöst."

[Expertinnen-Interviews\EI04; Position: 61 - 61]

5.

"Und es gibt auf der anderen Seite diese Leute, die alles ausprobieren wollen. Denen kommt Sinn erfassendes Lesen nicht in den Sinn. Und die probieren das aus. Aber man kann nicht alles ausprobieren."

[Expertinnen-Interviews\EI04; Position: 62 - 62]

6.

"Beide Extreme fühlen sich von der Informatik als Persönlichkeit angezogen. Richtig verstehen, alles logisch. Das ist toll, wenn ich so strukturiert bin. Ich kann ausprobieren, ich kann sofort programmieren, denkt die andere Gruppe. Und beide laufen auf Grundeis und kommen mit diesem Verhalten nicht weiter."

[Expertinnen-Interviews\EI04; Position: 63 - 63]

7.

"Und das sind Sachen, die man nicht der Schule ankreiden kann, sondern was entweder der Informatik anzukreiden ist, dass man beides braucht. Jeder muss selbst entscheiden dann, probiere ich das gerade aus oder lese ich das nach? Und das ist eine Entscheidung, die treffe ich selber zehn Mal am Tag. Also probiere ich das jetzt aus oder lese ich da die ganze Beschreibung dazu."

[Expertinnen-Interviews\EI04; Position: 64 - 64]

8.

"Hinderlich sind jeweils diese Fälle, von denen ich eben gesprochen habe. Entweder alles durch Ausprobieren lösen zu wollen, ist extrem hinderlich, oder alles hundertprozentig verstehen zu wollen, bevor ich anfange, ist auch hinderlich."

[Expertinnen-Interviews\EI04; Position: 74 - 74]

9.

"B: Ich weiß nicht, warum diese Gruppe erfolgreicher ist. Aber das unterscheidet uns vielleicht zum Beispiel von der Mathematik, muss man einfach sagen. In der Mathematik, da wäre es genauso. Um die Beweise zu führen, muss man es komplett verstanden haben. Also da funktioniert dieses Streben nach Komplettverstehen und dann kommt das bisschen handwerkliche Umsetzen. Und mit probieren geht fast gar nichts."

[Expertinnen-Interviews\EI04; Position: 76 - 76]

10.

"B: In der Informatik geht das mit probieren, und es gehört es auch mit dazu, um wirklich gut zu sein. Und das, das habe ich eigentlich so auch noch nie formuliert, aber ich glaube, das ist ein Kennzeichen von Informatik-Lernen. Ich muss das mal noch mal weiter durchdenken."

[Expertinnen-Interviews\EI04; Position: 77 - 77]

5.2.2. Aufeinander aufbauende Inhalte und Fähigkeiten

1.

"Eine große Schwierigkeit ist, dass Programmierung aufeinander aufbaut, damit auch auf der Veranstaltung des vorherigen Semesters. Nur durch immenses Nacharbeiten können Defizite aufgeholt werden."

[Expertinnen-Interviews\EI00; Position: 25 - 25]

2.

"B: Der Zuwachs passiert schon deshalb, weil Aufgaben aufeinander aufbauen und die Schwierigkeiten."

[Expertinnen-Interviews\EI03; Position: 16 - 16]

3.

"Wenn die Programme größer werden, wird es auch schwieriger für die Studierenden."

[Expertinnen-Interviews\EI06; Position: 35 - 35]

4.

"In dem Moment, wo die daheim nichts üben, haben sie im Dezember keine Chancen, etwas in der Vorlesung zu verstehen. Dann sind die Studenten frustriert und ich auch, weil sie nicht mehr mitmachen."

[Expertinnen-Interviews\EI06; Position: 89 - 89]

5.

"Programmieren baut massiv aufeinander auf. Die ersten Themen müssen verstanden werden, damit die nächsten verstanden werden können. Das wird durch die regelmäßigen Übungen erreicht. Egal wie gut das gemeistert wird, die Übung stellt eine Vertiefung der vorher behandelten Themen dar."

[Expertinnen-Interviews\EI06; Position: 90 - 90]

6.

"B: Algorithmen und Datenstrukturen wurde in der Vergangenheit ohne Pflicht durchgeführt. Das hat keinen Spaß gemacht. In den fortgeschrittenen Vorlesungen saßen alle da und wussten nicht, wovon ich rede."

[Expertinnen-Interviews\EI06; Position: 91 - 91]

7.

"Die Aufgaben sind nicht schwer. Fünfzig Prozent der Punkte können leicht erreicht werden. Damit zeigen sie ein Mindestmaß an Auseinandersetzung mit dem Thema, sodass sie in den kommenden Veranstaltungen noch folgen können."

[Expertinnen-Interviews\EI06; Position: 98 - 98]

8.

"Und das muss sein, weil Programmierung aufeinander aufbaut. Die Studierenden müssen dranbleiben."

[Expertinnen-Interviews\EI06; Position: 99 - 99]

5.2.3. Programmieren lernen in endlicher Zeit

1.

"Die Art von abstraktem Denken in der Programmierung kann nicht von allen in endlicher Zeit erlernt. Innerhalb eine Jahres würde es vielleicht funktionieren, aber nicht mit 250 Menschen pro Semester."

[Expertinnen-Interviews\EI00; Position: 41 - 41]

2.

"Das Feedback, das die Studierenden mitnehmen müssten, ist, dass sie in einer gewissen Zeit dazu in der Lage sind, diese Aufgabe zu lösen. Ist dem nicht so, haben sie ein Problem. Das sollte für die Studierenden ein Signal sein, um festzustellen, ob sie noch mehr machen müssen, oder das was sie machen ausreicht."

[Expertinnen-Interviews\EI00; Position: 103 - 103]

3.

"Für diese drei oder vier Aufgaben jede Woche darf man höchstens eine dreiviertel Stunde brauchen. Wenn man das nicht schafft, hat man im Allgemeinen ein Problem."

[Expertinnen-Interviews\EI00; Position: 108 - 108]

4.

"Aber das Feedback für die Studierenden ist, ob sie die Aufgabe in der Zeit schaffen oder nicht. Und das ist ein wichtiges Feedback."

[Expertinnen-Interviews\EI00; Position: 110 - 110]

5.

"Es gibt natürlich auch Studierende, die haben es in den Aufgaben hinbekommen, haben aber einfach zu lange Zeit gebraucht in der Klausur."

[Expertinnen-Interviews\EI01; Position: 63 - 63]

6.

"Programmieren ist nicht so kompliziert, das kann jeder schaffen. Es ist bloß eine Frage, wie lange braucht man dafür."

[Expertinnen-Interviews\EI06; Position: 25 - 25]

5.2.4. Abschreckende Nerd-Klischees

1.

"Es gibt leider das Nerd-Klischee. Das gibt es genauso für die Mathematik und die Physik, was eben in den Schulen doch die Runde macht, insbesondere unter den Mädchen. An manchen Schulen ist es extremer als an anderen. Da macht es die Runde, dass die Informatik oder die Ingenieursdisziplinen sich mehr um die Bedienung von Technik, als um die Gestaltung kümmern. Also das irgendwas ist, wo man von der Technik abhängig ist und dementsprechend so eine gewisse innere Einstellung besteht, dass es etwas für die anderen ist und nicht für einen selber. Da gibt es ja auch manchmal diese, ich sag jetzt mal Nerd-Kultur, die einen, die das können und die anderen, die das eben nicht können."

[Expertinnen-Interviews\EI05; Position: 42 - 42]

2.

"Und das ist dann wie bei den Indischen Kasten, dass es keine Chance gibt, von der einen in die andere Gruppe zu kommen. Weder in die eine noch in die andere Richtung sage ich mal."

[Expertinnen-Interviews\EI05; Position: 43 - 43]

3.

"Und ich glaube, das ist eine Sache, deswegen kämpfe ich auch so für so ein Pflichtfach Informatik, damit man eben insbesondere dieses Klischee aufräumt."

[Expertinnen-Interviews\EI05; Position: 44 - 44]

5.2.5. Zeitliche Einschränkungen durch Nebenjob

1.

"B: Die meisten Studierende arbeiten nebenbei, was den Lernfortschritt negativ beeinflusst."

[Expertinnen-Interviews\EI00; Position: 67 - 67]

2.

"Die Studierenden arbeiten alle nebenbei. Dadurch versuchen sie das Modul mit dem wenigsten Zeitaufwand zu meistern. Über eine nicht bestandene Klausur wundert man sich, aber diejenigen haben zu wenig Aufwand investiert. Wenn ich den Studierenden dieses Feedback zum zu geringen selbst erbrachten Zeitaufwand im Semester geben könnte, dass sie mehr tun müssen, wäre es besser. Es sollte ihnen wichtig sein. Aber die Studierenden haben nicht den Gesamtblick. Sie wissen auch noch nicht, wie die Klausur aussehen wird."

[Expertinnen-Interviews\EI00; Position: 105 - 105]

5.2.6. Angst vor Mathematik und formalen Ausdrücken

1.

"Viele Studierende haben große Angst vor Mathematik, oder vor allem was nach Komplexität aussieht."

[Expertinnen-Interviews\EI00; Position: 35 - 35]

5.2.7. Nichterscheinen in Präsenzveranstaltungen

1.

"B: Wer nicht in die Übung kommt, dem kann ich leider nicht helfen."

[Expertinnen-Interviews\EI01; Position: 61 - 61]

5.2.8. Fach-Englisch für Informatiker

1.

"B: Das Problem ist, man hat so vieles nicht gelernt. Was wir in der Schule nicht lernen, ist zum Beispiel die englischen Sprache. Schulenglisch ist eben nicht gleich dem informatischen Englisch. Man hatte das Gefühl, jedes zehnte Wort nachschlagen zu müssen. Das ist natürlich eine Katastrophe. Aber es tauchen viele Fremdworte für Unbelastete auf. Das heißt, das ist eine zweite große Hürde. Englisch ist die Lingua Franca der Informatik und die besten Erklärungen und Hilfen gibt es oft auch nur in Englisch. Das heißt, Studierende müssen sich damit vertraut machen und das gehört einfach mit zum Informatik-Studium dazu. Und typischerweise habe ich nicht so eine einführende Veranstaltung, sondern man muss sich wirklich dadurch quälen, bis man diese Fachtermini kann. Es ist wichtig, dass ich die ganzen Termini, und da sind viele am Anfang, parat habe. Das ist so eine Schwierigkeit."

[Expertinnen-Interviews\EI04; Position: 54 - 54]

5.2.9. kognitive Kompetenzen[4]

5.2.9.1. Systematik beim Problemlösen entwickeln

5.2.9.2. Abstraktion von Handlungsvorschriften

5.2.9.3. Transfer

5.2.9.4. Logische Ausdrücke korrekt benutzen

5.2.9.5. Werkzeuge zur Software-Entwicklung nutzen

5.2.9.6. Programme schreiben zur Problemlösung

5.2.9.7. Externe Ressourcen zum Lernen nutzen

5.2.9.8. Selbständige Organisation des Lernprozesses

5.2.9.9. Konzepte der Programmierung in eigenen Worten erklären

5.2.10. nicht-kognitive Kompetenzen

5.2.10.1. Sozial-kommunikative Fähigkeiten

5.2.10.1.1. mangelnde Kooperation & Kollaboration

5.2.10.1.2. mangelnde Kommunikation

5.2.10.2. Haltung & Einstellung

5.2.10.2.1. fehlendes Durchhaltevermögen bei Frust

5.2.10.2.2. fehlende Begeisterung & Spaß am Problemlösen/Informatik

5.2.10.2.3. fehlende Lernbereitschaft/Offenheit

5.2.10.2.4. fehlende aktive, regelmäßige Mitarbeit

5.2.10.2.5. fehlende freiwillige Mitarbeit

5.2.10.3. Programmiererfahrung fehlt durch zu wenig zeitintensives Üben

5.2.10.4. Selbstkompetenz

5.2.10.4.1. fehlendes Selbstvertrauen in eigene Fähigkeiten

5.2.10.4.2. fehlende Kreativität

5.2.10.4.3. fehlende Vorkenntnisse

[4] Kompetenzen, die im Zusammenhang mit Schwierigkeiten von Studierenden genannt werden, werden nicht als solche (und damit ein drittes Mal) kodiert. Stattdessen werden die bereits definierten und klassifizierten kognitiven und nicht-kognitiven Kompetenzen als zusätzliche Herausforderungen gelistet. Daher erfolgt an dieser Stelle lediglich die erneute Auflistung der Kategorien.