die hochschullehre
Interdisziplinäre Zeitschrift für Studium und Lehre

Jahrgang 2, 2016
www.hochschullehre.org

Yvonne Sedelmaier & Dieter Landes

Bewertung studentischer Kompetenzen zur Evaluation der Software Engineering-Ausbildung an Hochschulen mit SECAT (Software Engineering Competence Assessment Tool)

Zusammenfassung

Mit der wachsenden Bedeutung von Software kommt auch der Hochschulausbildung von Software-Ingenieur*innen höhere Aufmerksamkeit zu. Der vorliegende Beitrag beschreibt ein Bewertungsinstrument zur Analyse und Optimierung von Lernprozessen im Software Engineering. Kernelemente des Forschungsansatzes umfassen ein Kompetenzprofil, die darauf aufbauende didaktische Weiterentwicklung der kompetenzorientierten Ausbildung von Software-Ingenieur*innen sowie die systematische Evaluation der Wirksamkeit didaktischer Konzepte, um daraus Rückschlüsse auf die Lehr-Lern-Prozesse und ihre Wirkzusammenhänge zu ziehen.

Schlüsselwörter

Kompetenzbewertung; Fachdidaktik Software Engineering; Evaluation; Kompetenzorientierung; Lehrqualität

Abstract

The growing importance of software also entails increasing attention towards higher education of software engineers. This contribution presents an assessment instrument that is targeted towards analysis and optimization of learning processes in software engineering. Core elements of the research approach encompass a competence profile, the goal-directed didactic refinement of competence-oriented software engineering education, and the systematic evaluation of the efficacy of didactic settings, which may then imply conclusions on teaching and learning processes as well as on their interrelations.

Key words

Competence evaluation; discipline-specific teaching software engineering; evaluation; competence orientation; quality of teaching

1 Rahmenbedingungen und Vorarbeiten

1.1 Rahmenbedingungen

Software beeinflusst mittlerweile viele Bereiche unseres Alltags und zahlreiche Geräte werden mittels Software gesteuert, angefangen vom Herzschrittmacher, über Waschmaschinen und Airbags in Autos bis hin zu Mobiltelefonen (Sommerville, 2011). Damit gewinnt auch die Ausbildung im Software Engineering vermehrt an Bedeutung.

Software Engineering als Teilbereich der Informatik beschäftigt sich mit der Entwicklung komplexer Softwaresysteme in großen, interdisziplinären Teams über einen längeren Zeitraum hinweg und für eine mehr oder weniger bekannte Gruppe späterer Anwender*innen. Meist kann Software nicht nur von einem einzelnen Programmierer entwickelt werden, sondern durch ein oder mehrere Teams von Entwickler*innen. Softwareentwicklung ist ein sehr komplexer Prozess, geht weit über reines Programmieren hinaus und beinhaltet zahlreiche Aktivitäten, bei denen eine Vielzahl an Rollen innerhalb eines Entwicklungsteams zusammenarbeiten muss. Zudem erfordert Softwareentwicklung auch Kommunikation mit vielen Stakeholdern außerhalb des Teams, insbesondere mit Kunden (Vigenschow, Schneider & Meyrose, 2011). Software wird im Normalfall nicht für Informatiker*innen, sondern für Anwender*innen aus nahezu allen anderen Fachbereichen entwickelt.

Hinzu kommen der Umfang des zu entwickelnden Produktes und die Komplexität des Umfelds, in das das Projektergebnis einzubetten ist. Daraus resultieren eine teilweise jahrelange Zeitdauer, diffizile Entwicklungsprozesse und ein hohes Maß an erforderlicher Kreativität, um geeignete Lösungen zu entwickeln. Software-Ingenieur*innen benötigen also neben fundiertem Fachwissen auch zahlreiche miteinander verknüpfte kontextsensitive überfachliche Kompetenzen, die Hochschulausbildung adressieren muss. Erst diese „weichen Kompetenzen“ ermöglichen es den Software-Ingenieur*innen, ihr fachliches Können in die Tat umzusetzen, reines Fachwissen alleine genügt dafür nicht.

Die Komplexität eines echten Softwareentwicklungsprojektes kann wegen eingeschränkter Ressourcen, knapper Zeitfenster und der vielfältigen erforderlichen Kompetenzen nicht eins zu eins in der Hochschulausbildung abgebildet werden. Die Besonderheiten des Faches Software Engineering und die daraus folgenden Herausforderungen für die Hochschullehre begründen die Notwendigkeit einer Fachdidaktik für Software Engineering an Hochschulen.

1.2 Vorarbeiten

Das hier beschriebene Instrument *Software Engineering Competence Assessment Tool* (SECAT) wird im Rahmen einer Fachdidaktik für Software Engineering eingesetzt, um Kompetenzzuwachs Studierender einschätzen und dann Lernprozesse analysieren, besser verstehen und passgenauer adressieren zu können. Das Forschungsdesign fußt auf folgenden drei zentralen Säulen, die jedoch in diesem Beitrag nicht vertieft werden können.

1.2.1 Kompetenzprofil für Software Engineering

Ausgangspunkt ist ein tiefgehendes Verständnis, über welche Kompetenzen erfolgreiche Software-Ingenieur*innen überhaupt verfügen müssen und was sie genau bedeuten. Dies schließt sowohl fachliche, als auch überfachliche Kompetenzen ein, die zusammen genommen ein spezifisches Kompetenzprofil für Software Engineering ergeben. Das Kompetenzprofil *Software Engineering Body of Skills* (SWEBOS) (Sedelmaier & Landes, 2015c, 2015d) beschreibt eben diese kontextsensitiven überfachlichen Kompetenzen und dient in Verbindung mit fachlichen (Bourque & Fairley, 2014) und allgemeinen überfachlichen Kompetenzen als Ziel und Kompass der Hochschulausbildung. SWEBOS wurde datengetrieben mit einem qualitativen Forschungsansatz, der Grounded Theory (Glaser & Strauss, 2010) folgend, erforscht und als *thick description* (Mills, 2010, S. 942) dargestellt. Für eine detaillierte Beschreibung des Forschungsprozesses sowie eine Darstellung des vollständigen Kompetenzprofils wird auf (Sedelmaier & Landes, 2015c, 2015d) verwiesen. Allerdings ist dieses fachspezifische Kompetenzprofil eng mit dem Bewertungsinstrument SECAT verzahnt.

Die in SWEBOS enthaltenen kontextsensitiven überfachlichen Kompetenzen lassen sich wie folgt zusammenfassen, auch wenn das hermeneutische Verständnis und damit der Kontext erst in Verbindung mit den zugeordneten Erläuterungen ersichtlich werden. Dazu wird auf (Sedelmaier & Landes, 2015d) verwiesen:

- Kompetenzen für die professionelle Zusammenarbeit mit anderen Menschen (Z)
- Kommunikative Kompetenzen (K)
- Kompetenzen, um die eigene Arbeit zu strukturieren (S)
- Personale Kompetenzen (P)
- Fähigkeit, komplexe Vorgänge und Systeme sowie Zusammenhänge zu verstehen (Problembewusstsein) (V)
- Fähigkeit, das eigene Wissen und Können, die eigenen Kompetenzen auf konkrete, neue Situationen flexibel und kreativ anzuwenden (Lösungskompetenz) (L)

1.2.2 Kompetenzorientierte Hochschullehre im Software Engineering

Die zweite Säule bei der Entwicklung einer Fachdidaktik für Software Engineering zielt darauf ab, den besonderen Herausforderungen in der Lehre des Software Engineering zu begegnen. Es hat sich insbesondere gezeigt, dass Studierende des Software Engineering selbst noch nie ein großes Softwareprojekt bearbeitet haben und somit die dabei auftretenden Herausforderungen nicht sehen oder unterschätzen, geschweige denn fassen können, warum mögliche Lösungsansätze aus dem Software Engineering greifen, um diese Probleme zu lösen. Zudem neigen Studierende dazu, stark auf den fachlichen Inhalt zu fokussieren und überfachliche Kompetenzen weitgehend auszublenden, weil Softwareentwicklung vermeintlich eine rein technische Angelegenheit darstellt.

Traditionelle Lehre neigt dazu, Studierenden eben solche rein fachlichen Lösungen für Probleme anzubieten, die sie selbst noch nie konkret erlebt haben. Durch induktive und aktivierende Lehrmethoden werden diese abstrakten Problemstellungen im Software Engineering für die Studierenden erfahrbar und insbesondere auch die Bedeutung von Kommunikationsprozessen zwischen vielerlei Projektbeteiligten deutlich. Die didakti-

schen Konzepte fußen auf allgemeindidaktischen Theorien und Modellen wie etwa dem Hamburger oder Berliner Modell (Sedelmaier & Landes, 2015a), werden systematisch (weiter-)entwickeltet und umgesetzt und adressieren sowohl kontextsensitive als auch allgemeine überfachliche sowie fachliche Kompetenzen. Für detaillierte Beschreibungen einzelner didaktischer Konzepte wird auf (Sedelmaier & Landes, 2014a, 2014b, 2014c, 2015b) verwiesen.

1.2.3 Evaluation mit SECAT

Die dritte zentrale Säule im Forschungsdesign stellt die Evaluation von Bildungsmaßnahmen dar, um Erfolgsfaktoren des Lehrens und Lernens von Software Engineering zu identifizieren. Die Bewertung studentischer Kompetenzen liegt dabei als Gradmesser zugrunde und erfolgt mittels SECAT. Hier liegt der Fokus des Beitrages.

2 Theoretische Grundlagen

SECAT als Ansatz zur Kompetenzentwicklung und -einschätzung bei Studierenden basiert auf einem Kompetenzbegriff in der empirischen Bildungsforschung (Hartig, 2008, S. 17) und einem Kompetenzbewertungsansatz in der beruflichen Bildung (Rauner, 2011). Allerdings muss ein eigenes Verständnis von Kompetenz zugrunde gelegt werden, das auf die spezifischen Eigenschaften der Domäne Software Engineering Bezug nimmt. Genauso verhält es sich mit dem Kompetenzbewertungsansatz, der ebenso weiterentwickelt und angepasst werden muss.

2.1 Kompetenzbegriff

Die häufig zitierte Kompetenzdefinition nach Weinert (Weinert, 2001, S. 27–28) ist sehr allgemein. In der empirischen Bildungsforschung existieren Ansätze, Kompetenz als „erlernbare kontextspezifische kognitive Leistungsdispositionen“ (Hartig, 2008, S. 17) zu beschreiben. Ein ähnliches Kompetenzverständnis liegt dem Bewertungsinstrument SECAT zugrunde.

Unsere Forschungsergebnisse legen eine Unterscheidung in drei verschiedene Kompetenzarten, nämlich fachliche, kontextsensitive überfachliche und allgemeine überfachliche Kompetenzen nahe, um den Fachbezug zu Software Engineering herzustellen und ein tiefes Verständnis für die Bedeutung überfachlicher Kompetenzen im Kontext von Software Engineering zu erreichen. Die Unterscheidung fachlicher Kompetenzen, allgemeiner überfachlicher und kontextsensitiver überfachlicher Kompetenzen in SECAT geht auf das zugrundeliegende Kompetenzprofil SWEBOS (Sedelmaier & Landes, 2015d) zurück und folgt somit der Kontextbezogenheit von Kompetenzen in der Definition nach Hartig (Hartig, 2008, S. 17).

Fachliche Kompetenzen beziehen sich auf das Fachwissen der Software-Ingenieur*innen, etwa das Beherrschen einer Programmiersprache oder die Kenntnis von Vorgehensmodellen. Hier lag jahrelang der Schwerpunkt der Hochschulausbildung, auch aus der Vorstellung heraus, dass sich überfachliche Kompetenzen von selbst einstellen.

Allgemeine überfachliche Kompetenzen (z. B. Präsentationstechniken) lassen sich losgelöst vom fachlichen Kontext trainieren und etwa mit BEvaKomp (Braun, 2008) einschätzen.

Einzeln gesehen erscheinen kontextsensitive überfachliche Kompetenzen auf den ersten Blick wenig *besonders*. Sie bekommen jedoch bei näherer Betrachtung im Kontext der jeweiligen Fachlichkeit eine spezifische Charakteristik, werden sehr stark vom Fachlichen einer Disziplin, in diesem Fall des Software Engineering, geprägt und sind unabdingbar, damit technisches Fachwissen überhaupt erst anwendbar wird. Daher müssen kontextsensitive überfachliche Kompetenzen im Zusammenhang mit fachlichen Kompetenzen im Kontext von Software Engineering trainiert werden (Sedelmaier & Landes, 2015d). Somit erhalten allgemein scheinende Kompetenzen wie etwa kommunikative Kompetenzen oder Fragetechniken im Kontext von Software Engineering eine spezielle Ausprägung und unterscheiden sich beispielsweise von Fragetechniken in der Sozialen Arbeit. Es macht wenig Sinn, Informatik-Studierende in Fragetechniken auszubilden, wenn der Anwendungsbezug, wie z.B. der Einsatz in einem Kundengespräch zur Anforderungserhebung, fehlt.

Zudem sind diese Kompetenzen mit weiteren fachlichen und überfachlichen Kompetenzen verknüpft und bilden ein komplexes Geflecht, somit ein spezielles Kompetenzprofil für Software Engineering, weil z.B. eine Fachkompetenz wie etwa eine Modellierungstechnik für Geschäftsprozesse die Art der zu erfragenden Information beeinflusst und der Frageprozess mittels Arbeitstechniken strukturiert werden muss. Kompetenzen im Software Engineering sind also mehrfach komplex im Sinne von Bender et al. (Bender, Lerch, & Scheffel, 2014, S. 3).

2.2 Ziele und Intentionen von SECAT

Kompetenzen im Software Engineering sind sehr komplex. Um Lernprozesse im Software Engineering besser verstehen und Lehre kompetenzorientiert weiterentwickeln zu können, soll der Kompetenzzuwachs infolge eines Lernsettings analysiert werden. Setzt man dann den tatsächlichen Kompetenzzuwachs in Relation zu den angewandten didaktischen Konzepten, ermöglicht dies ein besseres Verständnis der Lehr-Lern-Zusammenhänge und eine gezielte Weiterentwicklung der Lehrkonzepte. Folglich müssen sich die zu erreichenden Kompetenzen (vgl. Abschnitt 1.2.1) im Evaluationsinstrument widerspiegeln.

Funktionales Fachwissen kann relativ einfach über klassische Prüfungsdesigns abgeprüft werden. Allgemeine überfachliche Kompetenzen können wegen der fehlenden Kontextabhängigkeit in unterschiedlicher Ausprägung in nahezu jeder Lehrveranstaltung in jedem Fach trainiert werden. Ein mögliches Evaluationsinstrument für diese Art von Kompetenzen ist etwa BEvaKomp (Braun, 2008).

Ein Evaluationsinstrument für eine Fachdidaktik des Software Engineerings muss aber im Gegensatz dazu die Besonderheiten des Faches berücksichtigen und darauf abgestimmt sein. Zudem muss ein Evaluationsinstrument den Regeln wissenschaftlicher Evaluation folgen (Reischmann, 2003; Rindermann, 2003). Das Evaluationsinstrument SECAT soll somit folgenden Anforderungen genügen:

- SECAT soll sämtliche Kompetenzen des Software Engineering in ihrer gesamten Komplexität abbilden können, also neben den fachlichen Kenntnissen besonders

die kontextsensitiven überfachlichen Kompetenzen einschließlich der fachspezifischen Besonderheiten. Es ist nicht ausreichend, ausschließlich allgemeine überfachliche oder sehr generische fachliche Kompetenzen zu bewerten wie etwa BEvaKomp (Braun, 2008).

- Eine einzige Lehrveranstaltung kann nicht alle erforderlichen Kompetenzen ausbilden. Dafür sind mehrere aufeinander abgestimmte Lehrveranstaltungen notwendig, die sich sinnvoll ergänzen und Kompetenzen Schritt für Schritt trainieren. Zum einen scheint es sinnvoll, die Kompetenzentwicklung durch einzelne Lehrveranstaltungen oder didaktische Methoden sowie deren Wirksamkeit zu evaluieren. Zum anderen soll die entwickelte Evaluationsmethodik auch quasi als summative Evaluation auf die Hochschulausbildung im Software Engineering als Gesamtes blicken können. Es ist also erforderlich, Kompetenzen auf verschiedenen Niveaustufen zu evaluieren.
- Gradmesser in SECAT sind besonders die fachlichen und kontextsensitiven überfachlichen Kompetenzen, die Studierende aufgrund der Lehrveranstaltung weiterentwickelt haben. Ziel ist also eine Bewertung des Kompetenzzuwachses aufgrund des Lehr-/Lernsettings.
- Im Software Engineering zählen die Einsicht und die persönliche Reife als Voraussetzung für Problembewusstsein, was sich nur schwer nur von außen beobachten lässt. Problembewusstsein und Lösungskompetenz im Sinne von Abwägen von Lösungsalternativen sind jedoch zentrale Kompetenzen im Software Engineering. Daher setzt das Instrument SECAT – anders als der Ansatz von Rauner – vorwiegend auf Selbstbewertungen und Selbsteinschätzung. Dass diese Form der Evaluation ebenso valide und reliabel sein kann, zeigt BEvaKomp (Braun, 2008).
- Gutes Software Engineering zeigt sich sowohl in den Abläufen und Prozessen während der Entwicklung, als auch im Ergebnis, der fertigen Software. Daher fließen auch beide Aspekte in die Bewertung der Lehre mit ein: Befähigt sie die Studierenden dazu, mit Hilfe eines systematischen Arbeitsprozesses zu guten Ergebnissen zu kommen? Eine Bewertung über Prozesse allein greift zu kurz, denn auch zielgerichtete Arbeitsprozesse führen im Software Engineering nicht zwangsläufig zu perfekten Ergebnissen. Umgekehrt sind im Software Engineering gute Ergebnisse auch gelegentlich durch überragende Leistungen Einzelner zu erzielen, ohne dass jedoch eine hinreichende Systematik verfolgt wurde.
- Da Software Engineering in einen komplexen organisatorischen bzw. betrieblichen Kontext eingebunden ist, sollen verschiedene Perspektiven auf einen Softwareentwicklungsprozess auch in die Analyse der Lernprozesse einbezogen werden können. Daher ermöglicht SECAT sowohl Selbsteinschätzungen der Kompetenzentwicklung von Studierenden, als auch externe Blickwinkel wie etwa die von Lehrenden oder Kunden.
- Da Software Engineering häufig im Team stattfindet, soll das Bewertungsinstrument sowohl individuelle, als auch Teamleistungen bewerten.
- SECAT erlaubt Deltaanalysen von Kompetenzen, die sich zwischen einem Zeitpunkt x und einen Zeitpunkt y entwickelt haben. So werden Kompetenzentwick-

lungsverläufe deutlich, die wiederum das Verständnis für Lehr-Lern-Prozesse und ihre Zusammenhänge vertiefen.

- Hochschullehre muss das Vorwissen der Studierenden berücksichtigen, das zu verschiedenen Zeitpunkten im Studium unterschiedlich ist. Ähnlich dem Modell nach Rauner (Rauner, 2011) soll SECAT die Einschätzung von Kompetenzen auf verschiedenen Niveaustufen ermöglichen. Dann kann SECAT während des gesamten Studiums eingesetzt werden und sowohl die individuellen Kompetenzen einzelner Studierender erheben, als auch die Lehre adressatenorientiert evaluieren und weiterentwickeln.

2.3 Kompetenzmodell

Ein empirisch und theoretisch fundierter Ansatz zur Kompetenzmessung, dessen Kompetenzmodell zudem einen Kontextbezug beinhaltet, ist in der beruflichen Bildung zu finden. Das dort zugrunde liegende Kompetenzmodell besteht aus den drei Kompetenzniveaustufen funktionale, prozessuale/konzeptuelle Kompetenz sowie ganzheitliche Gestaltungskompetenz, die durch acht Kriterien charakterisiert sind (Rauner, 2011). Um mit SECAT studentische Kompetenzen im Software Engineering einzuschätzen, ist es notwendig, dieses Kompetenzmodell auf den Kontext von Software Engineering anzupassen. Die acht Kriterien wurden also von der Berufsbildung im gewerblich-technischen Bereich auf Software Engineering übertragen. Mittels dieser acht Kriterien lässt sich das Kompetenzprofil SWEBOS (Sedelmaier & Landes, 2015d) für Software Engineering komplett abbilden.

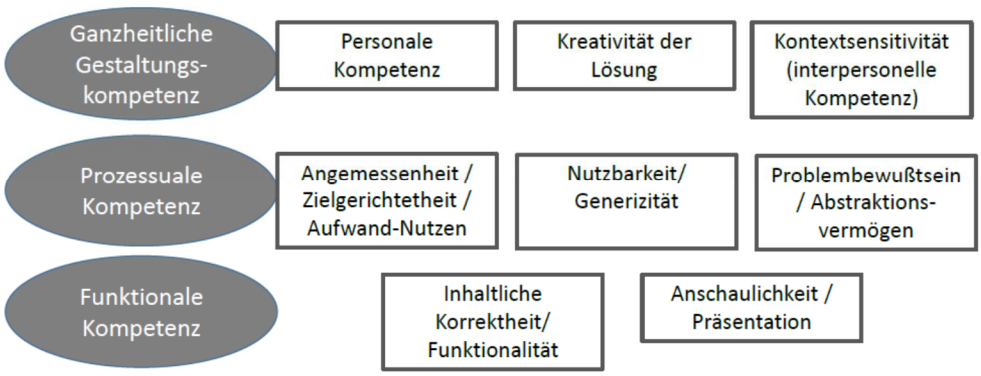

Abbildung 1 Kompetenzmodell in SECAT

Die Aufgliederung in die drei Niveaustufen funktionale, prozessuale und ganzheitliche Gestaltungskompetenz wurde beibehalten (siehe Abb.1). Auch die beiden Kriterien für funktionale Kompetenz wurden nahezu unverändert übernommen, während die übrigen an die Besonderheiten des Software Engineering angepasst und erweitert wurden.

In Übereinstimmung mit Rauner lassen sich die Kriterien *Funktionalität* und *Anschaulichkeit / Präsentation* wie folgt beschreiben:

> "(2) Funktionalität
> Die Funktionalität einer vorgeschlagenen Lösung beruflicher Aufgaben ist ein auf der

> Hand liegendes Kernkriterium bei deren Bewertung. Die Funktionalität verweist auf die instrumentelle Fachkompetenz bzw. das kontextfreie, fachsystematische Wissen und die fachkundlichen Fertigkeiten. Der Nachweis der Funktionalität einer Lösungsvariante ist grundlegend und maßgebend für alle weiteren Anforderungen, die an Aufgabenlösungen gestellt werden." (Rauner, 2011, p. 57).

Im Software Engineering handelt es sich hierbei etwa um das Beherrschen einer Programmiersprache.

> "(1) Anschaulichkeit / Präsentation
> Das Ergebnis beruflicher Aufgaben wird im Planungs- und Vorbereitungsprozess vorweg genommen und so dokumentiert und präsentiert, dass der/die Auftraggeber (Vorgesetzte, Kunden) die Lösungsvorschläge kommunizieren und bewerten können. Insofern handelt es sich bei der Veranschaulichung und Präsentation bzw. bei der Form einer Aufgabenlösung um eine Grundform beruflicher Arbeit und beruflichen Lernens. Eine zentrale Facette für die Kommunikation im Beruf ist die Fähigkeit, sich durch Beschreibungen, Zeichnungen und Skizzen und klar und strukturiert mitteilen zu können." (Rauner, 2011, p. 56)

Auf dem Niveau der prozessualen Kompetenz erfolgt die Beschreibung der Kriterien in Anlehnung an Rauner, jedoch mit Anpassung an Software Engineering:

Das Kriterium *Wirtschaftlichkeit* wird mit Bezug auf Software Engineering konkretisiert. Während Rauner allgemein die "...kontextbezogene Berücksichtigung wirtschaftlicher Aspekte bei der Lösung beruflicher Aufgaben" (Rauner, 2011, S. 57) nennt, kommt im Software Engineering die Ausrichtung des zu entwickelnden Softwaresystems auf Kundenanforderungen stärker zum Tragen. Auch hier ist wie bei Rauner das Verhältnis von Aufwand und betrieblichem Nutzen zu berücksichtigen. Daher heißt dieses Kriterium in SECAT *Angemessenheit / Zielgerichtetheit / Aufwand-Nutzen-Verhältnis*.

Rauners Kriterium *Nachhaltigkeit / Gebrauchswertorientierung* ist auf ein nicht real greifbares Produkt wie Software nicht direkt übertragbar und wird zum Kriterium *Nutzbarkeit / Generizität*[1] weiterentwickelt. Das Kriterium *Nutzbarkeit / Generizität* im Software Engineering bezieht sich auf Aspekte, die die Qualität eines Softwaresystems beschreiben. Gerade auch im Software Engineering ist Qualität ein bedeutender Faktor und häufig nur schwer zu fassen (vgl. z.B. Reißing, 2015). Einen Beitrag dazu leistet im Software Engineering u.a. der international anerkannte und weit verbreitete Standard ISO/IEC 25010 (ISO/IEC JTC 1/SC 25010:2011). Daher besitzt das Kriterium *Nutzbarkeit / Generizität* im Software Engineering folgende Indikatoren:

- Vollständigkeit: Deckt die Lösung die wesentlichen Anforderungen des Auftraggebers / Kunden ab?
- Usability: Wie hoch ist die Bedienerfreundlichkeit der Lösung für Anwender*innen?

[1] Generizität bezieht sich auf die Allgemeingültigkeit und Wiederverwendbarkeit einzelner Elemente im Software Engineering, z.B. Klassenbibliotheken oder durch Parametrierung.

- Robustheit: Wie unempfindlich ist die Lösung gegenüber Fehlbedienung oder anderen äußeren Störeinflüssen?
- Effizienz: Wie leistungsfähig ist die Lösung im Hinblick auf Laufzeit und Ressourcennutzung?
- Sicherheit: a) Ist das System vor unbefugtem Zugriff geschützt? (security) b) Vermeidet das System in ausreichendem Maß Schädigungen von Nutzer*innen oder seiner Einsatzumgebung? (safety)
- Erweiterbarkeit: Wie einfach lässt sich die Lösung auf geänderte Anforderungen und Rahmenbedingungen anpassen und erweitern?
- Dokumentation, Strukturiertheit: Wie gut wird die Beseitigung / Korrektur von Fehlern und / oder die Wiederverwendung einzelner Softwarekomponenten in der Lösung unterstützt?

Auch Rauners Kriterium *Geschäfts- und Arbeitsprozessorientierung* wird in SECAT modifiziert: Im Software Engineering genügt es nicht, lediglich vor- und nachgelagerte Prozesse und Schnittstellen bei der Lösung zu berücksichtigen. Vielmehr muss das Softwaresystem in ein komplexes Geflecht interagierender Faktoren eingepasst und mit einer Vielzahl an Rollen und Aufgaben innerhalb des Entwicklungsteams erstellt werden. Dabei ist es erforderlich, komplexe Vorgänge und Systeme sowie deren Zusammenhänge zu erkennen und zu verstehen. Da jedes Projekt anders ist, müssen auch fachliche Lösungsmöglichkeiten abgewogen und an die jeweilige Situation angepasst werden. Der Kontext, in dem die Aufgaben zu lösen sind, bedingt eine Vielzahl an technischen Herausforderungen für Software-Ingenieure, die in SWEBOS mit Problembewusstsein und Lösungskompetenz beschrieben werden. Dieses Kriterium wird zu *Problembewusstsein / Abstraktionsvermögen*.

Auch die Ebene der ganzheitlichen Gestaltungskompetenz erfordert Adaptionen. Rauners Kriterium *Sozialverträglichkeit* ändert im Zusammenhang mit Software Engineering seine Bedeutung. Der Zusammenarbeit in und außerhalb eines Teams kommt eine stärkere Bedeutung zu als in der gewerblich-technischen Ausbildung. Software kann meist nicht alleine von einem einzelnen entwickelt werden. Vielmehr findet Softwareentwicklung in einem sehr komplexen Umfeld statt. Dies beinhaltet zum einen die Zusammenarbeit innerhalb eines Teams, in dem verschiedene Rollen ausgefüllt werden müssen und zielgerichtet und effektiv zusammengearbeitet werden muss. Zum anderen ist es auch erforderlich, außerhalb des Teams, etwa mit Kunden oder späteren Nutzern, zu kommunizieren. Es müssen immer auch die Rahmenbedingungen, der Kontext, berücksichtigt und damit aktiv interagiert werden. Diese Kompetenz ist in SWEBOS als *Zusammenarbeit mit anderen Menschen* (Z) näher erläutert und beinhaltet einen Blick über den Tellerrand der eigenen *Entwicklungsinsel* hinaus. Hier unterscheidet sich das Kriterium vom Kriterium *Problembewusstsein / Abstraktionsvermögen*, das noch stark an der zu entwickelnden Software orientiert ist und auf komplexe technische Zusammenhänge abzielt, während das Kriterium *Kontextsensitivität / interpersonelle Kompetenz* auch etwa interdisziplinäre Zusammenhänge berücksichtigt. Hier geht es nicht mehr um das Softwaresystem *im Kleinen*, sondern um den Kontext, in den dieses eingebettet ist. Im Kriterium *Problembewusstsein / Abstraktionsvermögen* liegt der Fokus auf der zu entwickeln-

den Software und dem direkten Umfeld, während *interpersonelle Kompetenz* größere Zusammenhänge berücksichtigt und auch beispielsweise Aspekte der Teamführung integriert. Daher wurde dieses Kriterium in SECAT zu *Kontextsensitivität / interpersonelle Kompetenz*.

Rauners Kriterium *Umweltverträglichkeit* kommt im Zusammenhang mit physisch nicht greifbarer Software nur am Rande zum Tragen. Daher wurde es ersetzt durch ein Kriterium, das auf die personalen überfachlichen Kompetenzen von Software-Ingenieur*innen fokussiert, nämlich das Kriterium *personale Kompetenz*. Die Strukturierung in personale und interpersonelle Kompetenzen findet sich ebenfalls im Kompetenzprofil wieder und hat sich dort als sinnvoll erwiesen. Interpersonelle Kompetenzen kommen im Umgang und bei der Zusammenarbeit mit anderen Menschen besonders zum Tragen, wie etwa kommunikative Kompetenzen. Dagegen liegen *personale* Kompetenzen in der Person und ihren Einstellungen und Werten selbst.

Das Kriterium *personale Kompetenz* in SECAT beschreibt die in einer Person liegenden Fähigkeiten und Kompetenzen, die im Software Engineering erforderlich sind. Dazu gehören beispielsweise die Fähigkeit zur Strukturierung der eigenen Arbeit, Selbstreflexion oder auch Zeitmanagement (vgl. SWEBOS S und SWEBOS P) (Sedelmaier & Landes, 2015d). Anders als interpersonelle Kompetenzen / Kontextsensitivität kommen personale Kompetenzen unabhängig von der Zusammenarbeit mit anderen zum Tragen.

Rauners Kriterium der Kreativität, in dem er auf die höchst unterschiedlichen Gestaltungsspielräume bei der Lösung beruflicher Aufgaben hinweist, wurde unverändert übernommen. Er weist darauf hin, dass "das Kriterium 'Kreative Lösung' in besonderer Weise berufsspezifisch interpretiert und operationalisiert werden" muss (Rauner, 2011, S. 58).

2.4 Abgrenzung zu anderen Evaluations- und Messinstrumenten

Die Abgrenzung zu anderen Evaluations- und Messinstrumenten lässt sich in Tabelle 1 zusammenfassen.

Tabelle 1: Abgrenzung

	BEvaKomp	Rauner	SECAT
Ziel der Einschätzung	Lehrkonzept	Individuelle Kompetenz	Lehrkonzept
Kompetenzbegriff	Allgemein überfachlich	Fachlich, allgemein überfachlich und kontextbezogen überfachlich	Fachlich, allgemein überfachlich und kontextsensitiv überfachlich
Art der Einschätzung	Kompetenzzuwachs (relativ)	Kompetenzniveau (absolut)	Kompetenzzuwachs (relativ)
Fokus	Kompetenz	Aufgabe	Kompetenz, z.T. Aufgabe
Spezifität	Generisch, fach- und domain-unabhängig	Fach- bzw. domainspezifisch	Fach- bzw. domainspezifisch
Umfang der Einschätzung	Einzelperson	Einzelperson	Einzelperson; Personengruppe / Team
Perspektive	Selbsteinschätzung	Fremdeinschätzung	Selbst- und Fremdeinschätzung
Umfang / Scope	statisch (fester Fragebogen)	statisch (feste Bewertungskriterien)	variabel
Zeitpunkt	variabel	fest	variabel
Gegenstand	Outcome / Ergebnis	Outcome / Ergebnis	Outcome / Ergebnis; Prozess / Vorgehen

3 Operationalisierung von SECAT auf den konkreten Anwendungsfall

Das Kompetenzmodell im Bewertungsinstrument SECAT bildet als Zielmarke der Hochschulausbildung das gesamte Kompetenzprofil im Software Engineering ab. Üblicherweise werden allerdings in den einzelnen Lehrveranstaltungen nur Teilbereiche des Kompetenzprofils adressiert, so dass SECAT durch einen Konkretisierungsschritt auf den spezifischen Anwendungsfall etwa eines speziellen Lernsettings anzupassen ist, bevor eine Operationalisierung mit Items sinnvoll möglich ist.

Dieser Konkretisierungsschritt stimmt die Kriterien des Kompetenzmodells auf die jeweiligen Lehrziele einzelner Lehrveranstaltungen oder -einheiten ab. Dabei werden die acht Kriterien spezifiziert. So zeigt sich interpersonelle Kompetenz etwa beim Führen eines Kundengesprächs oder durch die Zusammenarbeit in ein Team.

Tabelle 2: Kompetenzmodell und zugehörige Konkretisierungen im Software Engineering Competence Assessment Tool (SECAT)

Kompetenzniveau Kriterium		Konkretisierungen (ggf. mit Verweis zum Kompetenzprofil SWEBOS – vgl. Abschnitt 1.2.1)
3.	Ganzheitliche Gestaltungskompetenz	
3.1	Kontextsensitivität (interpersonelle Kompetenz)	o Moderation o Führen eines Kundengespräches o Platz im Team/in der Gruppe finden o Sensitivität / Offenheit für Empfindungen und Bedürfnisse anderer (Empathie) o Führen eines Entwicklungsteams o professionelle Zusammenarbeit mit anderen Menschen (SWEBOS Z) o Sachliche Arbeitsweise / Verbindlichkeit o Konfliktfähigkeit
3.2	Personale Kompetenzen	o Durchhaltevermögen, am-Ball-bleiben, nachfragen, nachhaken (SWEBOS P) o Arbeitstechniken (SWEBOS S: Kompetenzen, um die eigene Arbeit zu strukturieren) o Strukturiertes Vorgehen o Rollenverteilung in der Gruppe o Zeitmanagement (SWEBOS S) / Termintreue o Kenntnis eigener Talente (SWEBOS Z) o Offenheit für Neues o Eigeninitiative / Motivation o Zielorientierung (SWEBOS S) o Selbstreflexion (SWEBOS P) o Frustrationstoleranz o Kompromissfähigkeit o Individuelle Teamfähigkeit o Qualitätsanspruch / Sorgfalt
3.3	Kreativität der Lösung	o Methodenvielfalt, um an Ergebnisse zu kommen o Berücksichtigung vorhandener Komponenten und Integration zu Lösung o Effizienz des Lösungswegs
2.	Prozessuale Kompetenz	o Zielgerichtetheit
2.1	Angemessenheit / Zielgerichtetheit / Aufwand-Nutzen-Verhältnis	o Anwendung methodischer Vorgehensweisen und Abstimmung auf den direkten Kontext o Aufwand-Nutzen-Abwägung, Priorisierung von Anforderungen o Anwendung methodischer Vorgehensweisen und Abstimmung auf den Kontext

		o Passgenauigkeit des Systems
2.2	Nutzbarkeit / Generizität	o Vollständigkeit o Usability o Robustheit / Fehlerfreiheit des Systems o Effizienz o Sicherheit (security, safety) o Erweiterbarkeit, Weiterentwicklungsmöglichkeiten des Systems (Nachhaltigkeit) o Dokumentation
2.3	Problembewusstsein / Abstraktionsvermögen	o Abwägen von Lösungsalternativen o Verständnis komplexer Vorgänge, Systeme und Zusammenhänge (Problembewusstsein) (SWEBOS V) o Berücksichtigung nicht-funktionaler Anforderungen
1.	Funktionale Kompetenz	
1.1	Inhaltliche Korrektheit / Funktionalität	
1.2	Anschaulichkeit / Präsentation	

So adressiert eine Lehrveranstaltung im vierten Semester des Bachelorstudiengangs Informatik der Hochschule Coburg im Bereich der *Kontextsensitivität/interpersonelle Kompetenzen* beispielsweise Kompetenzen, die erforderlich sind, um ein Kundengespräch zu führen. Dazu zählen z.B. Moderationstechniken, Gesprächsführung, Empathie, was sich auch im Kompetenzprofil wiederfindet. Im Gegensatz dazu konkretisiert sich Kontextsensitivität für einen Masterstudierenden, der ein Softwareentwicklungs-Projekt leitet, auf andere Weise, nämlich eher durch Teamkompetenzen wie Motivationsfähigkeit oder Führen von Mitarbeitergesprächen. Auch diese Kompetenzen sind im Kompetenzprofil zu finden, und auch sie konkretisieren das Kriterium *Kontextsensitivität*, erhalten im speziellen Kontext jedoch eine eigene Ausprägung (vgl. Tab.2). Die unterschiedlichen Schwerpunkte und Gewichtungen in den Konkretisierungen ergeben sich aus den unterschiedlichen Lehrzielen.

Selbst wenn die Konkretisierung dieselbe ist, so kann sie immer noch auf verschiedenen Niveaustufen angesiedelt sein, abhängig von Vorwissen und Kontext. Moderation beinhaltet etwa für Studienanfänger*innen eine andere und geringere Komplexität als für Masterstudierende und erfährt so eine andere Ausprägung.

Auch diese Konkretisierungen sind *nur* ein Zwischenschritt und müssen auf den jeweiligen Anwendungsfall heruntergebrochen werden. Also: Wie äußert sich diese interpersonelle Kompetenz beim Führen eines Kundengesprächs? Hier findet eine Spezifizierung auf die Lehrziele statt. Stehen die Konkretisierungen der jeweiligen Kriterien über Lehrziele dann fest, werden Items entwickelt, die diese Konkretisierungen weiter operationalisieren. So wird eine Bewertung aus unterschiedlichen Blickwinkeln mit verschiedenen Fragebögen für Studierende, Dozenten oder Projektkunden möglich (vgl. Abb. 2).

Aus Perspektive der Studierenden können sowohl Selbsteinschätzungen stattfinden, als auch Fremdbewertungen von Peers erhoben werden. Zudem ist es auf dieser Ebene dann auch möglich, neben Items, die die Teamleistung beurteilen, auch Items zu entwickeln, die individuelle Kompetenzentwicklung bewerten.

Abbildung 2 Vorgehensweise bei der Item-Entwicklung

Die Anzahl der Kriterien bzw. Items je Konkretisierung bzw. Kriterium erlaubt außerdem eine unterschiedliche Gewichtung der Kompetenzen z. B. für verschiedene Zielgruppen. So wurden etwa die einzelnen Kriterien für die Bewertung der Projektleiter*innen eines Software-Projektes wie in Tabelle 3 gezeigt gewichtet. Dabei wurde der Kompetenzzuwachs durch die Lehrenden, die Kunden des Projekts sowie seine Teammitglieder eingeschätzt. Da die Teammitglieder aufgrund mangelnder Berufserfahrung keine Aussagen zur Kreativität der Lösung bzw. zur Nutzbarkeit des Systems treffen konnten, wurden keine Fragen aus diesem Bereich gestellt (vgl. Abb. 3).

Tabelle 3: Gewichtung der Kriterien mittels Anzahl der Items am Beispiel

Kriterium	**Anzahl der Fragen je Perspektive**		
	Lehrende	**Kunde**	**Teammitglieder**
Funktionale Kompetenz			
o Inhaltliche Korrektheit / Funktionalität	4	2	4
o Anschaulichkeit / Präsentation	2	3	3
Prozessuale Kompetenz			
o Angemessenheit / Zielgerichtetheit / Aufwand-Nutzen-Verhältnis	4	6	11
o Nutzbarkeit / Generizität	4	5	0
o Problembewusstsein / Abstraktionsvermögen	4	2	5
Ganzheitliche Gestaltungskompetenz			
o Kontextsensitivität (interpersonelle Kompetenz)	6	6	3
o Personale Kompetenz	6	1	7
o Kreativität der Lösung	4	2	0
Summe der Items	**34**	**27**	**33**

Die Items können in perspektivenspezifischen Fragebögen zusammengefasst und abgefragt werden, wofür eine Likert-Skala geeignet erscheint.

Die Darstellung der Ergebnisse erfolgt üblicherweise auf Ebene der Kriterien oder innerhalb derselben Fragewelle auf Ebene der Konkretisierungen. Auf Ebene der Kriterien laufen alle Perspektiven der Bewertung zusammen. Die Kriterien bilden den festen, strukturgebenden Rahmen. Anpassung und Variabilität findet auf der darunter liegenden Stufe der Konkretisierungsebene statt. Der Vorteil dieser Methodik liegt in der Vergleichbarkeit über verschiedene Blickwinkel und Zeitpunkte hinweg bei ausreichender Anpassungsmöglichkeit an den konkreten Anwendungsfall.

4 Ergebnisse der Kompetenzbewertung mit SECAT

4.1 Validität

Beispielhaft sollen hier die Ergebnisse der Evaluation aus einem Software-Engineering-Projekt der Hochschule Coburg angeführt werden (Sedelmaier & Landes, 2014b). In diesem Lehrformat arbeiten Bachelor-Studierende in ihrem letzten Studiensemester in Teams zusammen, die von jeweils einem Masterstudierenden geleitet werden. Die Lehrziele sind für die beiden Studierendentypen unterschiedlich. Somit muss auch die Kompetenzeinschätzung angepasst werden, etwa auf die Lehrziele der Masterstudierenden.

Neben der Selbstbewertung durch die Studierenden selbst sind weitere Perspektiven möglich. Mit SECAT können Teammitglieder, Teamleiter*innen oder auch Kunden in die Kompetenzevaluation einbezogen werden. Als Beispiel sei hier das Kompetenzprofil eines Masterstudierenden eines Softwareentwicklungsprojektes aufgeführt (Abb. 3). Es zeigt die Perspektive des Lehrenden, eines Kunden sowie die Perspektive des Teams, das

der Masterstudierende geleitet hat. Der Masterstudierende erhielt ein 360°-Feedback von seinen Teammitgliedern, also fünf Bachelorstudierenden. So ist es mit SECAT möglich, dem Masterstudierenden ein umfassendes Feedback zu geben, das weit über die Sicht eines einzelnen Lehrenden hinausgeht. Dieses Vorgehen belegt zudem die Validität des Bewertungsinstruments, da die erhobenen Daten auf eine hohe Interrater-Reliabilität hinweisen.

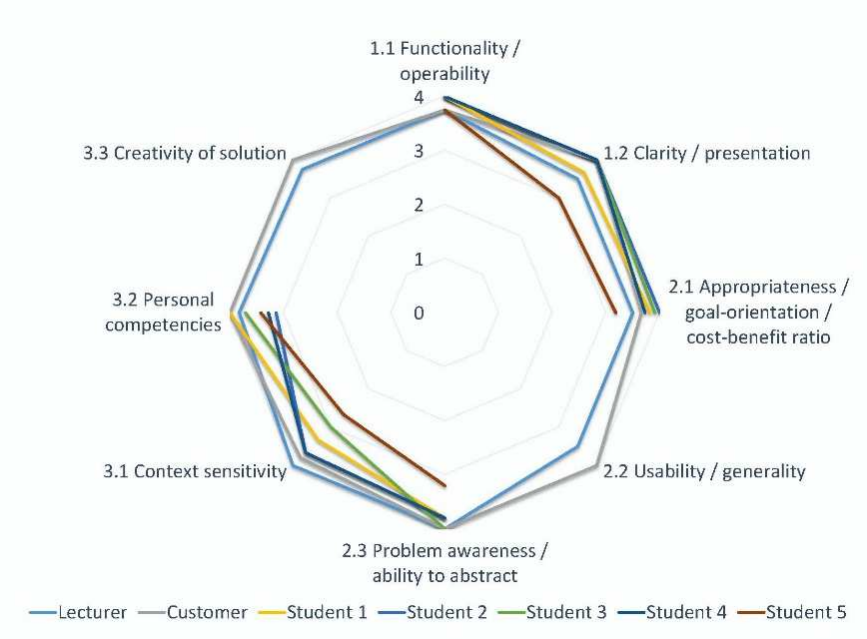

Abbildung 3: Kompetenzprofil eines Masterstudierenden bewertet aus verschiedenen Blickwinkeln

4.2 Ergebnisse im Hinblick auf die Weiterentwicklung kompetenzorientierter Lehrkonzepte

Werden die Ergebnisse der Kompetenzbewertung mehrerer Studierender, die dieselbe Lehrveranstaltung besucht haben, zusammengenommen, können etwa über Mittelwerte oder Mediane Rückschlüsse auf die Wirksamkeit der Lehre und der Lehrmethoden gezogen werden. Wenn mehrere Fragebögen zusammengenommen werden, mildern sich individuelle Ausreißer und Schwächen ab. Auch hier zeigen Ergebnisse bereits deutlich, dass diejenigen Studierenden, die an den evaluierten Lehrmethoden teilgenommen haben, einen Kompetenzzuwachs in den adressierten Kompetenzen sehen, der sich in Werten größer als 2 zeigt (vgl. Abb. 4).

Abb. 4 zeigt Daten aus einer Lehrveranstaltung im vierten Fachsemester des Studiengangs Informatik der Hochschule Coburg. Hier sollten die Studierenden in einem Kundengespräch Anforderungen an eine zu entwickelnde Software erheben (Sedelmaier & Landes, 2014a). Dazu wurden sie mit einem konkreten Arbeitsauftrag in die Vorbereitung eines solchen Kundengesprächs und dann in das Gespräch selbst geschickt. Die adressierten Lehrziele stammten in diesem Fall vornehmlich aus den Kriterien Problembewusstsein, Kontextsensitivität, Personale Kompetenz und Kreativität der Lösung. Diese

Kriterien wurden in den folgenden Bereichen (Tab. 4) konkretisiert, die sich ebenso in der Auswertung (Abb. 4) wiederfinden.

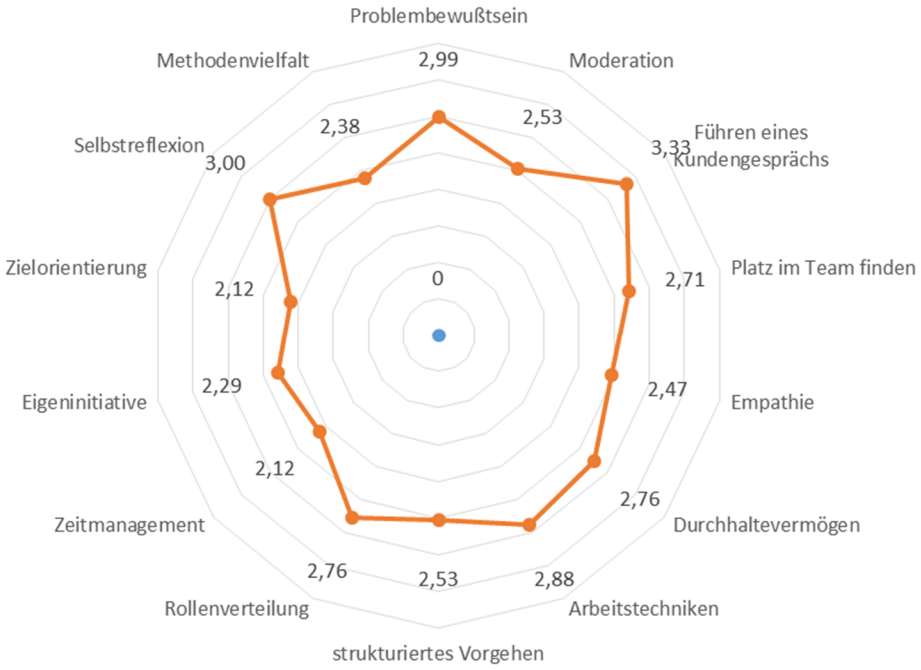

Abbildung 4: Mittlerer Kompetenzzuwachs aufgrund der Lehreinheit je Konkretisierung

Die stärkste Gewichtung lag mit 10 Fragen auf der Kontextsensitivität, gefolgt von personalen Kompetenzen (8 Fragen), Problembewusstsein mit 4 Fragen und Kreativität der Lösung mit zwei Fragen. In diesem konkreten Fall wurde nur die individuelle Selbsteinschätzung der Studierenden als Perspektive verwendet.

Bei genauerer Analyse der Daten lässt sich feststellen, dass besonders diejenigen Studierenden (Nr. 3, 5 und 10), die nicht an allen entsprechenden Lehreinheiten teilgenommen hatten, einen geringeren Kompetenzzuwachs in den adressierten Bereichen angeben als ihre Kommilitonen (Abb. 5). Insgesamt jedoch weisen die Studierenden in den meisten Fällen einen deutlichen Kompetenzzuwachs auf (Abb. 4).

Aus diesen Resultaten können Rückschlüsse auf das angewandte didaktische Konzept und seine Wirksamkeit gezogen werden.

Tabelle 4: Anpassung an konkrete Lehrziele am Beispiel

Kriterium	**Konkretisierungen**
Prozessuale Kompetenz	
Problembewusstsein / Abstraktions-vermögen	o Fähigkeit, komplexe Vorgänge und Systeme zu verstehen, Zusammenhänge zu verstehen (Problembewusstsein) (SWEBOS V)
Ganzheitliche Gestaltungskompetenz	
Kontextsensitivität (interpersonelle Kompetenz)	o Moderation o Führen eines Kundengesprächs o Platz im Team finden o Sensitivität / Offenheit für Empfindungen und Bedürfnisse anderer (Empathie)
Personale Kompetenz	o Arbeitstechniken (SWEBOS S) o Strukturiertes Vorgehen o Rollenverteilung in der Gruppe o Zeitmanagement (SWEBOS S) / Termintreue o Eigeninitiative / Motivation o Zielorientierung (SWEBOS S) o Selbstreflexion (SWEBOS P)
Kreativität der Lösung	o Methodenvielfalt, um an Ergebnisse zu kommen

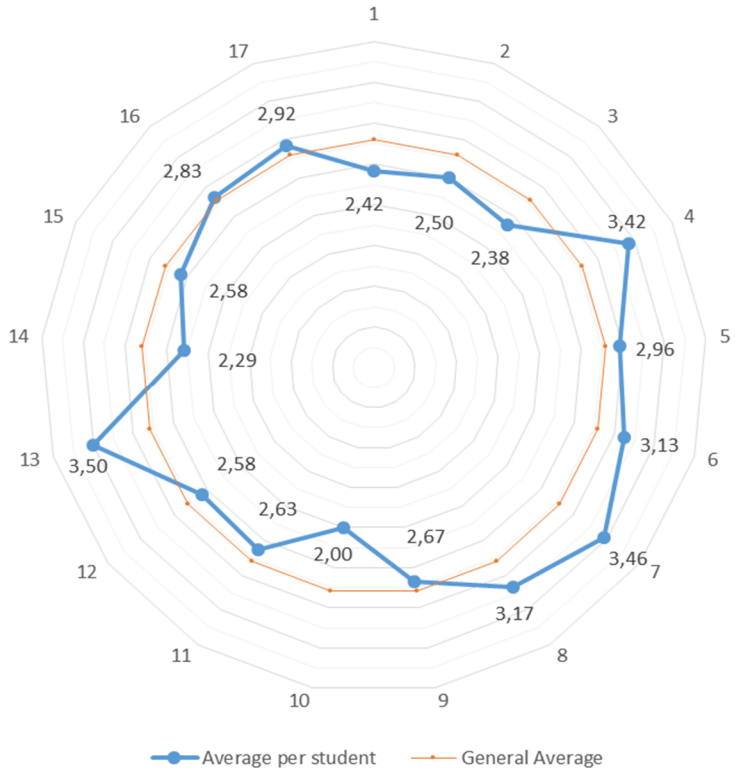

Abbildung 5 Kompetenzzuwachs in den adressierten Kriterien je Studierender

5 Zusammenfassung und Ausblick

Der vorliegende Beitrag beschreibt ein Bewertungsinstrument zur kompetenzorientierten Weiterentwicklung der Lehre von Software Engineering. Kernelemente des Forschungsansatzes umfassen ein Kompetenzprofil, die darauf aufbauende kompetenzorientierte Weiterentwicklung von Lehrkonzepten sowie die systematische Evaluation der Wirksamkeit dieser Lehrkonzepte, um daraus Rückschlüsse auf die Lehr-Lern-Prozesse und ihre Wirkzusammenhänge zu ziehen.

Die vorliegenden Ergebnisse zeigen deutlich, dass sich aus kumulierten Werten mehrerer Studierender derselben Lehreinheit Rückschlüsse auf die Wirksamkeit von Lehrkonzepten ziehen lassen. Ein besseres Verständnis, welche didaktischen Konzepte gut funktioniert haben und welche noch Verbesserungspotential aufweisen, erlaubt es, daraus auch Rückschlüsse auf abstrakterer Ebene zu ziehen. So kann schrittweise eine Fachdidaktik für Software Engineering an Hochschulen entwickelt werden, indem gezeigt werden kann, wie Lernerfolg, der hier evaluiert wird, und didaktischer Ansatz zusammenhängen könnten und miteinander in Wechselwirkung stehen. Daraus resultiert ein besseres Verständnis für Lehr-Lernprozesse im Software Engineering.

Aus den vorliegenden Ergebnissen kann beispielsweise geschlossen werden, dass im Software Engineering induktive Lehr-Lern-Arrangements erfolgreicher sind, die es Studierenden ermöglichen, zuerst ein Problem zu erleben und zu erfassen, bevor sie eine mögliche Lösung geliefert bekommen (Sedelmaier, 2016). Aufgrund der Komplexität des Fachs fehlt Studierenden andernfalls häufig das Verständnis für den Lerninhalt, was somit auch zu einem schlechteren Lernergebnis führt. Es scheint für Studierende vermeintlich sinnlos, sich mit Lösungen für Probleme zu beschäftigen, die sie selbst noch nie erfahren haben. Kompetenzbewertung mit SECAT verdeutlicht, welche Art von *Problemerfahrung* es Studierenden ermöglicht, sich für das zu Lernende zu öffnen und es anzunehmen, welche Voraussetzungen die Studierenden mitbringen und wie ihnen der Zugang zum Thema erleichtert werden kann.

In weiteren Studien gilt es, das Instrument SECAT weiter zu validieren. Dazu ist eine Ausweitung der Datenbasis erforderlich. In einem weiteren Schritt kann dann die Übertragbarkeit von SECAT auf Hochschulfächer mit ähnlichen Herausforderungen geprüft werden.

In diesem Zusammenhang ist es sinnvoll, die bereits als Prototyp entstandene Software weiterzuentwickeln. Diese Software vereinfacht zum einen das Erstellen von Befragungen und bietet zum anderen verschiedene Auswertungsmöglichkeiten. Das dahinter liegende Datenmodell des Softwarewerkzeugs ist auf das Kompetenzmodell von SECAT einschließlich der Konkretisierungsebene sowie die Vorgehensweise bei der Bewertung abgestimmt. Zudem erleichtert diese dann die Auswertung und Vergleichbarkeit der Daten. Dies ist insbesondere bei einer Ausweitung der Datenbasis hilfreich.

Danksagung

Das Projekt EVELIN wird vom Bundesministerium für Bildung und Forschung unter dem Förderkennzeichen 01PL12022A im Rahmen des Qualitätspakts Lehre gefördert.

Literatur

Bender, W., Lerch, S. & Scheffel, M. (2014). *Interdisziplinäre Kompetenzen Studierender evaluieren.: 2. Zwischenbericht der Wissenschaftlichen Begleitstudie zum Projekt „Der Coburger Weg" zum 31.05.2014.*

Bourque, P. & Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge Version 3.0 - SWEBOK. IEEE Computer Society Press. Abgerufen von: http://www.computer.org/ieeecs-swebokdelivery-portlet/swebok/SWEBOKv3.pdf

Braun, E. (2008). *Das Berliner Evaluationsinstrument für selbsteingeschätzte studentische Kompetenzen (BEvaKomp)*. Göttingen: V & R Unipress.

Glaser, B. G., & Strauss, A. L. (2010). *Grounded Theory: Strategien qualitativer Forschung* (3. Auflage). Bern: Huber.

Hartig, J. (2008). Kompetenzerfassung von Bildungsprozessen. In N. Jude, J. Hartig, & E. Klieme (Hrsg.), *Bildungsforschung: Vol. 26. Kompetenzerfassung in pädagogischen Handlungsfeldern - Theorien, Konzepte und Methoden* (S. 15–25). Bonn, Berlin: Bundesministerium für Bildung und Forschung (BMBF).

Mills, A. J. (2010). *Encyclopedia of Case Study Research* (Bd. 1). Thousand Oaks: Sage Publications.

Rauner, F. (2011). *Messen beruflicher Kompetenzen. Bildung und Arbeitswelt: Vol. 24*. Münster: Lit.

Reischmann, J. (2003). *Weiterbildungs-Evaluation: Lernerfolge messbar machen. Grundlagen der Weiterbildung Arbeitshilfen*. Neuwied: Luchterhand.

Reißing, R. (2015). Softwarequalität. In J. Krahl & J. Löffl (Eds.), *Zwischen den Welten: Vol. 3. Qualitäten* (pp. 87–108). Göttingen: Cuvillier Verlag.

Rindermann, H. (2003). Lehrevaluation an Hochschulen: Schlussfolgerungen aus Forschung und Anwendung für Hochschulunterricht und seine Evaluation. *Zeitschrift für Evaluation*. (2), 233–256. Abgerufen von: http://www.zfev.de/fruehereAusgabe/ausgabe2003-2/artikel/ZfEv2-2003_5-Rindermann.pdf

Sedelmaier, Y. (2016). *Interdisziplinäre Fachdidaktik für Software Engineering – Forschungsbasierte Entwicklung und Evaluation eines anwendungsbezogenen didaktischen Ansatzes*. Bamberg: opus. Abgerufen von: https://opus4.kobv.de/opus4-bamberg/files/46321/sedelmaierDissopusk_A3a.pdf

Sedelmaier, Y. & Landes, D. (2014a). A Multi-Level Didactical Approach to Build up Competencies in Requirements Engineering. In B. Penzenstadler, S. Gregory, & D. Landes (Eds.), *8th International Workshop on Requirements Engineering Education & Training (REET 2014), 22nd International Conference on Requirements Engineering (RE 2014)* (pp. 26–34). CEUR Workshop Proceedings vol. 1217. Abgerufen von: http://ceur-ws.org/Vol-1217/paper4.pdf

Sedelmaier, Y. & Landes, D. (2014b). Practicing Soft Skills in Software Engineering. In L. Yu (Ed.), *Overcoming Challenges in Software Engineering Education* (S. 161–179). IGI Global.

Sedelmaier, Y. & Landes, D. (2014c). Using Business Process Models to Foster Competencies in Requirements Engineering. In *27th International Conference on Software Engineering Education and Training (CSEE&T)* (S. 13–22).

Sedelmaier, Y. & Landes, D. (2015a). A Competence-Oriented Approach to Subject-Matter Didactics for Software Engineering. *International Journal of Engineering Pedagogy (iJEP)*, 5(3), 34–44. doi:10.3991/ijep.v5i3.4664

Sedelmaier, Y. & Landes, D. (2015b). Active and Inductive Learning in Software Engineering Education. In *37th International Conference on Software Engineering (ICSE)* (S. 418–427).

Sedelmaier, Y. & Landes, D. (2015c). SWEBOS - The Software Engineering Body of Skills. *International Journal of Engineering Pedagogy (iJEP)*, 5(1), 12–19. Abgerufen von: http://online-journals.org/index.php/i-jep/article/view/4047

Sedelmaier, Y. & Landes, D. (2015d). Überfachliche Kompetenz im Software Engineering - Modellierung, Förderung und Messung in der Hochschulausbildung. In U. Riegel, S. Schubert, G. Siebert-Ott, & K. Macha (Eds.), *Kompetenzmodellierung und -messung in den Fachdidaktiken* (S. 111–130). Münster: Waxmann.

Sommerville, I. (2011). *Software Engineering* (9. Auflage). Boston: Pearson.

ISO/IEC JTC 1/SC 25010:2011 (2011, March 1). Berlin: Beuth.

Vigenschow, U., Schneider, B., & Meyrose, I. (2011). *Soft Skills für Softwareentwickler: Fragetechniken, Konfliktmanagement, Kommunikationstypen und -modelle* (2. Auflage). Heidelberg: dpunkt.verlag.

Weinert, F. E. (2001). Vergleichende Leistungsmessung in Schulen - eine umstrittene Selbstverständlichkeit. In F. E. Weinert (Hrsg.), *Beltz-Pädagogik. Leistungsmessungen in Schulen* (S. 17–31). Weinheim: Beltz.

Autor/-innen

Dr. Yvonne Sedelmaier. Hochschule Coburg, Fakultät für Elektrotechnik und Informatik, Coburg, Deutschland; Email: yvonne.sedelmaier@hs-coburg.de

Prof. Dr. Dieter Landes. Hochschule Coburg, Fakultät für Elektrotechnik und Informatik, Coburg, Deutschland; Email: dieter.landes@hs-coburg.de

Zitiervorschlag: Sedelmaier, Y. & Landes, D. (2016). Bewertung studentischer Kompetenzen zur Evaluation der Software Engineering-Austbildung an Hochschulen mit SECAT (Software Engineering Competence Assessment Tool). *die hochschullehre*, Jahrgang 2/2016, online unter: www.hochschullehre.org