Schema zur Erfassung der fachlichen Kompetenzen

Studiengang Informatik (B.Sc.) -- Hochschule Kempten

IST-Analyse

Wissensgebiet	Teilbereich I	Teilbereich II	Teilbereich III	Erinnern / Wissen	Verstehen	Erklären	Verwenden	Anwenden	(Weiter-) Entwickeln	Anmerkungen / Notizen I
Quantum Computing (QC)	Software-Engineering für Quantum Computing	Basics	Vorgehensmodelle	Die Studierenden kennen unterschiedliche "Vorgehensmodelle" (z.B. Wasserfall, Scrum)	Die Studierenden verstehen den Sinn von "Vorgehensmodelle" im SE und warum es unterschiedliche Varianten gibt.	Die Studierenden können die über die unterschiedlichen "Vorgehensmodelle" diskutieren.	Die Studierenden können geeignete "Vorgehensmodelle" unter Anleitung auswählen.	Die Studierenden können geeignete "Vorgehensmodelle" selbstständig auswählen und anwenden.	.	Kompetenzanalysen, die bereits im Rahmen des EVELIN-Projekts am Standort Kempten aufgestellt wurden, wurde als Hilfestellung herangezogen.
			Phasen im SE	Die Studierenden kennen die unterschiedlichen "Phasen in SE". (Requirements Engineering, Analyse, Architektur und Design, Implementierung und Qualitätssicherung)	Die Studierenden verstehen die Hauptaufgaben der einzelnen "Phasen im SE".	Die Studierenden können die "Phasen im SE" und deren Aufgaben mit eigenen Worten erläutern.	.	.	.	
			UML	Die Studierenden kennen den Begriff "UML" und wissen, dass es eine Modellierungssprache ist. Zusätzlich kennen sie auch die zugehörige Notation.	Die Studierenden verstehen die Notation und die Notwendigkeit von "UML".	Die Studierenden können die Grundelemente erklären und Zusammenhänge illustrieren.	Die Studierenden können unter Aufsicht einfache Aufgaben mit Hilfe der "UML" modellieren.	Die Studierenden sind in der Lage die "UML" selbstständig einzusetzen um Sachverhalte zu modellieren.	.	
			SE und Ethik	Die Studierenden kennen die Wichtigkeit des Zusammenhangs von "ethischen Werten und Softwareentwicklung".	Die Studierenden verstehen warum und wozu "ethische Werte für die Softwareentwicklung" bedeutend sind.	Die Studierenden können erklären warum und wozu "ethische Werte für die Softwareentwicklung" bedeutend sind.	.	.	.	
		Requirements-Engineering / (Quantum) Requirements-Engineering	Definitionen von Software-Requirements	Die Studierenden kennen die Definition von "SE Requirements".	Die Studierenden verstehen was der Begriff "SE Requirements" umfasst.	Die Studierenden können den Begriff "SE Requirements" mit eigenen Worten beschreiben.	.	.	.	
			Requirements-Engineering (RE)	Die Studierenden kennen das RE als ein Teilgebiet des SE und auch die Hauptaufgabe.	Die Studierenden verstehen die Notwendigkeit des RE als Teildisziplin.	Die Studierenden können das RE und die Hauptaufgabe erläutern sowie darüber diskutieren.	.	.	.	
			Die Hauptaufgaben im RE	Die Studierenden kennen die "Hauptaufgaben" (ermitteln, dokumentieren, prüfen, verwalten) im RE.	Die Studierende verstehen die "Hauptaufgaben" (ermitteln, dokumentieren, prüfen, verwalten) im RE.	Die Studierenden können die "Hauptaufgaben" im RE mit eigenen Wörtern erklären.	.	.	.	
		Implementierung / (Quantum) Implementierung	Grundlagenwissen über unterschiedliche Programmiersprachen und Einsatzgebiete	Die Studierenden kennen unterschiedliche "Programmiersprachen". (z.B. Python, Java, C++)	Die Studierenden verstehen warum es unterschiedliche Arten von "Programmiersprachen mit unterschiedlichen Einsatzgebieten" gibt. (z.B. Python: Data-Science, C++: Spieleentwicklung)	Die Studierenden sind in der Lage über die "spezifischen Einsatzgebiete von unterschiedlichen Programmiersprachen" sowie über "Programmiersprachen" per se zu diskutieren und argumentieren.	.	.	.	
			Programmierung / Coding	Die Studierenden können sich die "Programmierung" als ein Teil des SE merken.	Die Studierenden verstehen die Wichtigkeit und Tragweite der "Programmierung"	Die Studierenden können über die "Programmierung" diskutieren und argumentieren.	Die Studierenden können angeleitet Probleme mittels "Programmierung in C/C++" (als Handwerk) lösen.	Die Studierenden sind in der Lage selbstständig Probleme mittels "Programmierung in C/C++" (als Handwerk) zu lösen.	.	
			Compiler / Interpreter	Die Studierenden können sich die beiden Begriffe "Compiler" und "Interpreter" merken.	Die Studierenden verstehen den Unterschied zwischen "Compiler" und "Interpreter".	Die Studierenden können mit eigenen Worten den Unterschied zwischen "Compiler" und "Interpreter" erklären und auch warum beide benötigt werden.	.	.	.	
			Datentypen	Die Studierenden können sich den Begriff "Datentypen" sowie Beispiele (z.B. int, float, bool) merken.	Die Studierenden verstehen warum "Datentypen" benötigt werden und warum es unterschiedliche Arten gibt.	Die Studierenden können "Datentypen" erklären und die Argumentation mit eigenen Beispielen festigen.	Die Studierenden können unter Anleitung den richtigen "Datentyp" auswählen und im eigenen Code (Programm) einsetzen.	Die Studierenden können selbstständig unterschiedliche "Datentypen" auswählen und im eigenem Code (Programm) anwenden.	.	
			Schleife	Die Studierenden können sich den Begriff "Schleife" sowie Beispiele (z.B. for, while) merken.	Die Studierenden verstehen warum "Schleifen" benötigt werden und wie die unterschiedlichen Arten funktionieren.	Die Studierenden können "Schleifen" erklären und die unterschiedlichen Arten illustrieren.	Die Studierenden können unter Anleitung "Schleifen" im eigenen Code (Programm) einsetzen.	Die Studierenden können selbstständig unterschiedliche Arten von "Schleifen" auswählen und im eigenem Code (Programm) anwenden.	.	
			Selektion	Die Studierenden können sich den Begriff "Selektion" sowie Beispiele (z.B. if, switch) merken.	Die Studierenden verstehen warum "Selektion" benötigt wird und wie die unterschiedlichen Arten funktionieren.	Die Studierenden können "Selektion" erklären und die unterschiedlichen Arten illustrieren.	Die Studierenden können unter Anleitung "Selektion" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Aufsicht unterschiedliche Arten der "Selektion" auswählen und im eigenem Code (Programm) anwenden.	.	
			Methode / Funktion	Die Studierenden können sich die Begriffe "Methode" und "Funktion" merken.	Die Studierenden verstehen warum "Methoden" und "Funktionen" benötigt werden und was der Unterschied dazwischen ist.	Die Studierenden können "Methoden" und "Funktionen" erklären und die unterschiedlichen Arten darlegen.	Die Studierenden können unter Anleitung "Methoden" und "Funktionen" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Anleitung "Methoden" und "Funktionen" im eigenem Code (Programm) anwenden.	.	
			Variable	Die Studierenden können sich den Begriff "Variable" merken.	Die Studierenden verstehen warum "Variablen" benötigt werden.	Die Studierenden können "Variablen" erläutern.	Die Studierenden können unter Anleitung "Variablen" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Anleitung "Variablen" im eigenen Code (Programm) einsetzen.	.	
			Spezielle Programmierkonzepte für das QC	.	.	.	.	.	.	
			Debugger	Die Studierenden kennen die Funktion eines "Debuggers".	Die Studierenden verstehen wozu und wie ein "Debugger" verwendet wird und was die Ausgaben bedeuten.	Die Studierenden können erklären, wozu und warum ein "Debugger" verwendet wird und was die Ausgaben eines "Debugger" bedeuten.	Die Studierenden können einen "Debugger" angeleitet verwenden.	Die Studierenden können einen "Debugger" selbstständig verwenden.	.	
			Software-Prototypen	Die Studierenden wissen, was "Software-Prototypen" sind und wofür sie verwendet werden.	Die Studierenden verstehen, was "Software-Prototypen" sind und wofür sie verwendet werden.	Die Studierenden können mit eigenen Worten erklären was "Software-Prototypen" sind und warum sie eingesetzt werden.	Die Studierenden können unter Anleitung einfache "Software-Prototypen" erstellen.	Die Studierenden sind in der Lage selbstständig für Probleme "Software-Prototypen" umzusetzen.	.	
		Design / (Quantum) Design	SE Design	Die Studierenden wissen, dass "SE-Design" eine Rolle im SE Projekt spielt.	Die Studierenden verstehen, welche Rolle das "SE-Design" in einem SE Projekt spielt	Der Studierende kann erklären, welche Rolle "SE-Design" im SE spielt.	.	.	.	
			Design / Architektur	Die Studierenden kennen den "Unterschied zwischen Architektur und Design".	Die Studierenden verstehen den "Unterschied zwischen Architektur und Design".	.	.	.	.	
			Design-Pattern / Muster	Die Studierenden kennen ausgewählte "Muster" (z.B. Observer, Kompositum und MVC, Adaptermuster, Zustand, Singelton, Strategy, Money, Fassade, Command, Command with undo, Command Processor, Factory, Decorator).	Die Studierenden verstehen den Unterschied und die Einsatzmöglichkeiten ausgewählter "Muster".	Die Studierenden können den Unterschied und die Einsatzmöglichkeiten ausgewählter "Muster" erklären.	Die Studierenden können die ausgewählten "Muster" in vorgegebenen Szenarien verwenden.	Die Studierenden können die ausgewählten "Muster" in nicht vorgegebenen Szenarien verwenden.	.	
			Struckturdiagramme	Die Studierenden kennen das Objekt-, Klassen-, Deployment-, Paket-, Komponenten- sowie das Subkomponentendiagramm.	Die Studierenden verstehen wie das Objekt-, Klassen-, Deployment-, Paket-, Komponenten- sowie das Subkomponentendiagramm verwendet wird.	Die Studierenden können erklären, in welchem Kontext das Objekt-, Klassen-, Deployment-, Paket-, Komponenten- sowie das Subkomponentendiagramm verwendet wird.	Die Studierenden können das Objekt-, Klassen-, Deployment-, Paket-, Komponenten- sowie das Subkomponentendiagramm angeleitet verwenden.	Die Studierenden können das Objekt- und das Klassendiagramm selbstständig anwenden.	.	
			Verhaltensdiagramme	Die Studierenden kennen das Kommunikations-, Aktivitäts-, Zustands- und Sequenzdiagramm.	Die Studierenden verstehen, wie das Kommunikations-, Aktivitäts-, Zustands- und Sequenzdiagramm verwendet wird.	Die Studierenden können erklären, wie das Kommunikations-, Aktivitäts-, Zustands- und Sequenzdiagramm eingesetzt wird.	Die Studierenden können das Kommunikations-, Aktivitäts-, Zustands- und Sequenzdiagramm verwenden.	Die Studierenden können das Kommunikations-, Aktivitäts-, Zustands- und Sequenzdiagramm anwenden.	.	
			Objektorientiertes Design	Die Studierenden wissen, dass es Methoden und Strategien gibt, um ein "objektorientieres SE-Design" anfertigen zu können.	Die Studierenden verstehen, wie Methoden und Strategien für ein "objektorientieres SE-Design" eingesetzt werden können.	Die Studierenden können erklären, wie verschiedene Methoden und Strategien eingesetzt werden können, um "objektorientieres SE-Design" anzufertigen.	Die Studierenden können in einem praxisbezogenen Beispiel Methoden und Strategien verwenden, um ein "objektorientieres SE-Design" anzufertigen.	Die Studierenden können selbstständig Methoden und Strategien anwenden, um ein "objektorientieres SE-Design" anzufertigen.	.	
			Hardware & Softwarekomponenten	.	.	.	.	.	.	
		Testing / (Quantum) Testing	Grundbegriffe	Die Studierenden wissen, was mit dem Begriff "Test" gemeint ist und kennen verschiedene Fehlerarten.	Die Studierenden verstehen den Unterschied zwischen verschiedenen "Fehlerarten".	Die Studierenden können die Bedeutung von Tests allgemein und den Unterschied zwischen verschiedenen "Fehlerarten" erklären.	.	.	.	
			Testarten I	Die Studierenden kennen verschiede "Testarten", die bei unterschiedlichen Anwendungsarten verwendet werden können.	Die Studierenden verstehen, wie verschiedene "Testarten" funktionieren und wie diese bei unterschiedlichen Anwendungsarten verwendet werden können.	Die Studierenden können erklären, wie verschiedene "Testarten" funktionieren und wie diese bei unterschiedlichen Anwendungsarten verwendet werden können.	.	.	.	
			Testarten II	Die Studierenden wissen, dass es für verschiedene Realisierungsphasen unterschiedliche "Testarten", (z.B. Unittests) gibt	Die Studierenden verstehen, für welche Phase welcher "Test" (z.B. Unittests) geeignet ist.	Die Studierenden können erklären, warum welcher "Test" für welche Phase geeignet ist (z.B. Unittests).	Die Studierenden können in einem vorgegebenen Beispiel eine geeignete "Testarten" auswählen und diese anwenden (z.B. Unittests).	Die Studierenden können selbstständig geeignete "Testarten" auswählen und diese anwenden (z.B. Unittests).	.	
			Testtechniken	Die Studierenden kennen die folgenden "Testtechniken": functional, sructural, deterministic.	Die Studierenden verstehen warum, wann und wie man die folgenden "Testtechniken" einsetzen und kombinieren kann.	Die Studierenden können erklären warum, wann und wie man die folgenden "Testtechniken" einsetzen und kombinieren kann.	.	.	.	
		Wartung	Grundbegriffe Wartung	Die Studierenden wissen, was mit "Wartung und Wartungsbetrieb" in SE-Projekten gemeint ist.	Die Studierenden verstehen warum "Wartung und Wartungsbetrieb" benötigt wird. (Warum Software Wartung benötigt.)	Die Studierenden können erklären, was mit "Wartung und Wartungsbetrieb" gemeint ist.	.	.	.	
		Auslieferung	Grundbegriffe Auslieferung	Die Studierenden wissen was mit der "Auslieferung in SE-Projekten" gemeint ist.	Die Studierenden verstehen was mit "Auslieferung in SE-Projekten" gemeint ist.	Die Studierenden können erklären, was mit "Auslieferung in SE-Projekten" gemeint ist.	.	.	.	

Studiengang Informatik (M.Sc.) -- Hochschule Kempten

SOLL-Analyse

Teilbereich III	Erinnern / Wissen	Verstehen	Erklären	Verwenden	Anwenden	(Weiter-) Entwickeln	Anmerkungen / Notizen
Vorgehensmodelle	Die Studierenden kennen unterschiedliche "Vorgehensmodelle" (z.B. Wasserfall, Scrum)	Die Studierenden verstehen den Sinn von "Vorgehensmodelle" im SE und warum es unterschiedliche Varianten gibt.	Die Studierenden können die über die unterschiedlichen "Vorgehensmodelle" diskutieren.	Die Studierenden können geeignete "Vorgehensmodelle" unter Anleitung auswählen.	Die Studierenden können geeignete "Vorgehensmodelle" selbstständig auswählen und anwenden.	.	.
Phasen im SE	Die Studierenden kennen die unterschiedlichen "Phase in QSE". (Requirements Engineering, Analyse, Architektur und Design, Implementierung und Qualitätssicherung)	Die Studierenden verstehen die Hauptaufgaben der einzelnen "Phasen im QSE".	Die Studierenden können die "Phasen" und Aufgaben mit eigenen Worten erläutern. Zusätzlich auch den Unterschied zwischen SE und QSE erklären können.	.	.	.	.
UML	Die Studierenden kennen den Begriff "UML" und wissen, dass es eine Modellierungssprache ist. Zusätzlich kennen sie auch die zugehörige Notation.	Die Studierenden verstehen die Notation und die Notwendigkeit von "UML".	Die Studierenden können die Grundelemente erklären und Zusammenhänge illustrieren.	Die Studierenden können unter Aufsicht einfache Aufgaben mit Hilfe der "UML" modellieren.	Die Studierenden sind in der Lage die "UML" selbstständig einzusetzen um Sachverhalte zu modellieren.	.	Aktuell gibt es keine QC-spezifische Modellierungssprache und deshalb sollte die "UML" eingesetzt werden, falls es möglich ist.
SE und Ethik	Die Studierenden kennen die Wichtigkeit des Zusammenhangs von "ethischen Werten und Softwareentwicklung im Kontext QC".	Die Studierenden verstehen warum und wozu "ethische Werte für die Softwareentwicklung im Kontext QC" bedeutend sind.	Die Studierenden können erklären warum und wozu "ethische Werte für die Softwareentwicklung im Kontext QC" bedeutend sind.	.	.	.	.
Definitionen von Software-Requirements	Die Studierenden kennen die Definition von "SE Requirements".	Die Studierenden verstehen was der Begriff "SE Requirements" umfasst.	Die Studierenden können den Begriff "SE Requirements" mit eigenen Worten beschreiben.	.	.	.	Das Gebiet (Quantum)-RE ist ein neues Forschungsgebiet und deshalb muss stets auf den Stand der Forschung geachtet werden.
Requirements-Engineering (RE)	Die Studierenden kennen das RE als Teilgebiet des SE und auch die Hauptaufgabe.	Die Studierenden verstehen die Notwendigkeit des RE als Teildisziplin.	Die Studierenden können das RE und die Hauptaufgabe erläutern sowie darüber diskutieren.	.	.	.	
Die Hauptaufgaben im RE	Die Studierenden kennen die "Hauptaufgaben" (ermitteln, dokumentieren, prüfen, verwalten) im RE.	Die Studierende verstehen die "Hauptaufgaben" (ermitteln, dokumentieren, prüfen, verwalten) im RE.	Die Studierenden können die "Hauptaufgaben" im RE mit eigenen Wörtern erklären.	.	.	.	
Grundlagenwissen über unterschiedliche Programmiersprachen und Einsatzgebiete	Die Studierenden kennen unterschiedliche klassische (z.B. Python, Java, C++) und nicht "klassische Programmiersprachen" (z.B. Qiskit, Q#).	Die Studierenden verstehen warum es unterschiedliche Arten von "Programmiersprachen" gibt mit unterschiedlichen Einsatzgebieten gibt. Zusätzlich verstehen die Studierenden warum es für QC neue bzw. Erweiterte Sprachen bedarf. (Nutzung von physikalischen Effekten bei der Programmierung)	Die Studierenden sind in der Lage über die spezifischen Einsatzgebiete von unterschiedlichen "Programmiersprachen" zu diskutieren und argumentieren.	.	.	.	.
Programmierung / Coding	Die Studierenden können sich die "Programmierung" als ein Teil der Informatik (mit QC) merken.	Die Studierenden verstehen die Wichtigkeit und Tragweite der "Programmierung"	Die Studierenden können über die "Programmierung" diskutieren und argumentieren.	Die Studierenden können angeleitet Probleme mittels "Programmierung" (als Handwerk mit Python (Qiskit)) lösen.	Die Studierenden sind in der Lage selbstständig Probleme mittels "Programmierung" (als Handwerk mit Python (Qiskit)) zu lösen.	.	.
Compiler / Interpreter	Die Studierenden können sich die beiden Begriffe "Compiler" und "Interpreter" merken.	Die Studierenden verstehen den Unterschied zwischen "Compiler" und "Interpreter".	Die Studierenden können mit eigenen Worten den Unterschied zwischen "Compiler" und "Interpreter" erklären und auch warum beide benötigt werden.	.	.	.	.
Datentypen	Die Studierenden können sich den Begriff "Datentypen" sowie Beispiele (z.B. int, float, Qubit) merken.	Die Studierenden verstehen warum "Datentypen" benötigt werden und warum es unterschiedliche Arten gibt.	Die Studierenden können "Datentypen" erklären und die Argumentation mit eigenen Beispielen festigen.	Die Studierenden können unter Anleitung den richtigen "Datentyp" auswählen und im eigenen Code (Programm) einsetzen.	Die Studierenden können selbstständig unterschiedliche "Datentypen" auswählen und im eigenem Code (Programm) anwenden.	.	.
Schleife	Die Studierenden können sich den Begriff "Schleife" sowie Beispiele (z.B. for, while) merken.	Die Studierenden verstehen warum "Schleifen" benötigt werden und wie die unterschiedlichen Arten funktionieren.	Die Studierenden können "Schleifen" erklären und die unterschiedlichen Arten illustrieren.	Die Studierenden können unter Anleitung "Schleifen" im eigenen Code (Programm) einsetzen.	Die Studierenden können selbstständig unterschiedliche Arten von "Schleifen" auswählen und im eigenem Code (Programm) anwenden.	.	.
Selektion	Die Studierenden können sich den Begriff "Selektion" sowie Beispiele (z.B. if, Switch) merken.	Die Studierenden verstehen warum "Selektion" benötigt wird und wie die unterschiedlichen Arten funktionieren.	Die Studierenden können "Selektion" erklären und die unterschiedlichen Arten illustrieren.	Die Studierenden können unter Anleitung "Selektion" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Aufsicht unterschiedliche Arten der "Selektion" auswählen und im eigenem Code (Programm) anwenden.	.	.
Methode / Funktion	Die Studierenden können sich die Begriffe "Methode" und "Funktion" merken.	Die Studierenden verstehen warum "Methoden" und "Funktionen" benötigt werden und was der Unterschied dazwischen ist.	Die Studierenden können "Methoden" und "Funktionen" erklären und die unterschiedlichen Arten darlegen.	Die Studierenden können unter Anleitung "Methoden" und "Funktionen" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Aufsicht "Methoden" und "Funktionen" im eigenem Code (Programm) anwenden.	.	.
Variable	Die Studierenden können sich den Begriff "Variable" merken.	Die Studierenden verstehen warum "Variablen" benötigt werden.	Die Studierenden können "Variablen" erläutern.	Die Studierenden können unter Anleitung "Variablen" im eigenen Code (Programm) einsetzen.	Die Studierenden können ohne Anleitung "Variablen" im eigenen Code (Programm) einsetzen.	.	.
Spezielle Programmierkonzepte für das QC	Die Studierenden können sich merken, dass es für die Programmierung von Quantencomputer "zusätzliche, nicht klassische Konzepte (z.B. Messen von Qubits, Superposition, Verbundenheit)" gibt.	Die Studierenden verstehen, was diese "Konzept für die Programmierung von QC" bedeuten.	Die Studierenden können diese Konzepte in Kontext "Programmierung" erklären.	Die Studierenden können unter Anleitung die "spezielle Konzepte für das QC" im eigenen Code (Programm) einsetzen.	Die Studierenden können selbstständig auswählen und die "speziellen Konzepte für das QC" einsetzten.	.	.
Debugger	Die Studierenden kennen die Funktion eines Debuggers.	Die Studierenden verstehen wozu und wie ein Debugger verwendet wird und was die Ausgaben bedeuten.	Die Studierenden können erklären, wozu und warum ein Debugger verwendet wird und was die Ausgaben eines Debuggers bedeuten.	Die Studierenden können einen Debugger angeleitet verwenden.	Die Studierenden können einen Debugger selbstständig verwenden.	.	Es gibt derzeitig noch keine "echten" QC-Debugger.
Software-Prototypen	Die Studierenden wissen, was Prototypen sind und wofür sie verwendet werden.	Die Studierenden verstehen, was Prototypen sind und wofür sie verwendet werden.	Die Studierenden können mit eigenen Worten erklären was Prototypen sind und warum sie eingesetzt werden.	Die Studierenden können unter Anleitung einfache Prototypen erstellen.	Die Studierenden sind in der Lage selbstständig für Probleme Prototypen umzusetzen.	.	.
SE Design	Die Studierenden wissen, dass "SE-Design" eine Rolle im SE Projekt spielt.	Die Studierenden verstehen, welche Rolle das "SE-Design" in einem SE Projekt spielt	Der Studierende kann erklären, welche Rolle "SE-Design" im SE spielt.	.	.	.	. I. Das Gebiet (Quantum)-Design ist ein neues Forschungsgebiet und deshalb muss stets auf den Stand der Forschung geachtet werden. II. Es gibt derzeitig nur experimentelle Methoden für Design & Architektur.
Design / Architektur	Die Studierenden kennen den "Unterschied zwischen Architektur und Design".	Die Studierenden verstehen den "Unterschied zwischen Architektur und Design".	Die Studierenden können den Unterschied zwischen "Architektur" und Design in eigenen Worten erklären.	.	.	.	.
Design-Pattern / Muster	Die Studierenden kennen ausgewählte "(Quantum)-Muster".	Die Studierenden verstehen den Unterschied und die Einsatzmöglichkeiten ausgewählter "(Quantum)-Muster".	Die Studierenden können den Unterschied und die Einsatzmöglichkeiten ausgewählter "(Quantum)-Muster" erklären.	Die Studierenden können die ausgewählten "(Quantum)-Muster" in vorgegebenen Szenarien verwenden.	Die Studierenden können die ausgewählten "(Quantum)-Muster" in nicht vorgegebenen Szenarien verwenden.	.	Keine genauere Spezifizierung von Design-Pattern / Muster in QC aktuell möglich.
Struckturdiagramme	Die Studierenden kennen unterschiedliche "Struckturdiagramme" für das QC.	Die Studierenden verstehen wie die unterschiedlichen "Struckturdiagramme" für das QC verwendet werden.	Die Studierenden können erklären, in welchem Kontext die "Struckturdiagramme" für das QC verwendet werden.	Die Studierenden können "Struckturdiagramme" für das QC angeleitet anwenden.	Die Studierenden können "Struckturdiagramme" für das QC selbstständig anwenden.	.	Keine genauere Spezifizierung von Struckturdiagramme in QC aktuell möglich.
Verhaltensdiagramme	Die Studierenden kennen unterschiedliche "Verhaltensdiagramme" für das QC.	Die Studierenden verstehen wie die unterschiedlichen "Verhaltensdiagramme" für das QC verwendet werden.	Die Studierenden können erklären, in welchem Kontext die "Verhaltensdiagramme" für das QC verwendet werden.	Die Studierenden können "Verhaltensdiagramme" für das QC angeleitet anwenden.	Die Studierenden können "Verhaltensdiagramme" für das QC selbstständig anwenden.	.	Keine genauere Spezifizierung von Verhaltensdiagrammen in QC aktuell möglich.
Objektorientiertes Design	Die Studierenden wissen, dass es Methoden und Strategien gibt, um ein "objektorientieres SE-Design" anfertigen zu können.	Die Studierenden verstehen, wie Methoden und Strategien für ein "objektorientieres SE-Design" eingesetzt werden können.	Die Studierenden können erklären, wie verschiedene Methoden und Strategien eingesetzt werden können, um "objektorientieres SE-Design" anzufertigen.	Die Studierenden können in einem praxisbezogenen Beispiel Methoden und Strategien verwenden, um ein "objektorientieres SE-Design" anzufertigen.	Die Studierenden können selbstständig Methoden und Strategien anwenden, um ein "objektorientieres SE-Design" anzufertigen.	.	Derzeitig noch unklar ob OOB sich auch im QC etablieren wird.
Hardware & Softwarekomponenten	Die Studierenden wissen, dass es eine wechselseitige Beziehung zwischen Hardware und Softwarekomponenten gibt und wie diese miteinander kommunizieren.	Die Studierenden versteht die wechselseitige Beziehung zwischen Hardware und Softwarekomponenten und wie diese miteinander kommunizieren.	Die Studierenden können die wechselseitige Beziehung zwischen Hardware und Softwarekomponenten erklären. Zudem können sie erklären, wie diese miteinander kommunizieren.	Die Studierenden können in einem vorgegebenen Szenario beurteilen, welche Softwarekomponenten zu welcher Hardware passen und kann entscheiden, wie diese miteinander kommunizieren.	Die Studierenden können selbstständig beurteilen, welche Softwarekomponenten zu welcher Hardware passen, kann entscheiden wie diese miteinander kommunizieren.	.	.
Grundbegriffe	Die Studierenden wissen, was mit dem Begriff "Test" gemeint ist und kennen verschiedene Fehlerarten.	Die Studierenden verstehen den Unterschied zwischen verschiedenen "Fehlerarten".	Die Studierenden können die Bedeutung von Tests allgemein und den Unterschied zwischen verschiedenen "Fehlerarten" erklären.	.	.	.	I. Das Gebiet (Quantum)-Design ist ein neues Forschungsgebiet und deshalb muss stets auf den Stand der Forschung geachtet werden. II. Es gibt derzeitig nur experimentelle Testing-Verfahren.
Testarten I	Die Studierenden kennen verschiede "Testarten", die bei unterschiedlichen Anwendungsarten verwendet werden können.	Die Studierenden verstehen, wie verschiedene "Testarten" funktionieren und wie diese bei unterschiedlichen Anwendungsarten verwendet werden können.	Die Studierenden können erklären, wie verschiedene "Testarten" funktionieren und wie diese bei unterschiedlichen Anwendungsarten verwendet werden können.	.	.	.	
Testarten II	Die Studierenden wissen, dass es für verschiedene Realisierungsphasen unterschiedliche "Testarten" gibt.	Die Studierenden verstehen, für welche Phase welcher "Test" geeignet ist.	Die Studierenden können erklären, warum welcher "Test" für welche Phase geeignet ist.	Die Studierenden können in einem vorgegebenen Beispiel eine geeignete "Testarten" auswählen und diese anwenden.	Die Studierenden können selbstständig geeignete "Testarten" auswählen und diese anwenden.	.	
Testtechniken	Die Studierenden kennen die folgenden "Testtechniken": functional, sructural, deterministic.	Die Studierenden verstehen warum, wann und wie man die folgenden "Testtechniken" einsetzen und kombinieren kann.	Die Studierenden können erklären warum, wann und wie man die folgenden "Testtechniken" einsetzen und kombinieren kann.	.	.	.	
Grundbegriffe Wartung	Die Studierenden wissen, was mit "Wartung und Wartungsbetrieb" in SE-Projekten gemeint ist.	Die Studierenden verstehen warum "Wartung und Wartungsbetrieb" benötigt wird. (Warum Software Wartung benötigt.)	Die Studierenden können erklären, was mit "Wartung und Wartungsbetrieb" gemeint ist.	.	.	.	Das Gebiet (Quantum)-Wartung ist ein neues Forschungsgebiet und deshalb muss stets auf den Stand der Forschung geachtet werden.
Grundbegriffe Auslieferung	Die Studierenden wissen was mit der "Auslieferung in SE-Projekten Kontext QC" gemeint ist.	Die Studierenden verstehen was mit "Auslieferung in SE-Projekten im Kontext QC" gemeint ist.	Die Studierenden können erklären, was mit "Auslieferung in SE-Projekten Kontext QC" gemeint ist. Zusätzlich sind die Studiereden in der Lage die derzeitigen Probe bei der "Auslieferung im Kontext QC" zu diskutieren. (Aktuell wird noch keine Software ausgeliefert.)	.	.	.	Das Gebiet (Quantum)-Auslieferung ist ein neues Forschungsgebiet und deshalb muss stets auf den Stand der Forschung geachtet werden.

IST-SOLL-Abgleich

Teilbereich III	Erinnern	Verstehen	Erklären	Verwenden	Anwenden	(Weiter-) Entwickeln	Anmerkungen / Notizen
Vorgehensmodelle	=	=	=	=	=	.	.
Phasen im SE	=	=	=	.	.	.	.
UML	=	=	=	=	=	.	.
SE und Ethik	+	+	+	.	.	.	.
Definitionen von Software-Requirements	=	=	=	.	.	.	.
Requirements-Engineering (RE)	=	=	=	.	.	.	.
Die Hauptaufgaben im RE	=	=	=	.	.	.	.
Grundlagenwissen über unterschiedliche Programmiersprachen und Einsatzgebiete	+	+	+	.	.	.	.
Programmierung / Coding	+	+	+	+	+	.	.
Compiler / Interpreter	=	=	=	.	.	.	.
Datentypen	+	+	+	+	+	.	.
Schleife	+	+	+	+	+	.	.
Selektion	+	+	+	+	+	.	.
Methode / Funktion	+	+	+	+	+	.	.
Variable	+	+	+	+	+	.	.
Spezielle Programmierkonzepte für das QC	O	O	O	O	O	.	.
Debugger	+	+	+	+	+	.	.
Software-Prototypen	+	+	+	+	+	.	.
SE Design	=	=	=	.	.	.	.
Design / Architektur	=	=	+	.	.	.	.
Design-Pattern / Muster	+	+	+	+	+	.	.
Struckturdiagramme	+	+	+	+	+	.	.
Verhaltensdiagramme	+	+	+	+	+	.	.
Objektorientiertes Design	+	+	+	+	+	.	.
Hardware & Softwarekomponenten	O	O	O	O	O	.	.
Grundbegriffe	+	+	+	.	.	.	.
Testarten I	+	+	+	.	.	.	.
Testarten II	+	+	+	+	+	.	.
Testtechniken	+	+	+	.	.	.	.
Grundbegriffe Wartung	+	+	+	.	.	.	.
Grundbegriffe Auslieferung	+	+	+	.	.	.	.