Modulbezeichnung	**Software-Reengineering**
Modulverantwortliche(r)	Prof. Dr. R. Koschke
Modulart	Pflicht/Wahl ☐ Wahlpflicht ☒
Spezialisierungsbereich	Systemsoftware / Eingebettete Systeme
Dauer des Moduls	1 Semester
Kreditpunkte	6 CP
Arbeitsaufwand	Berechnung des Workloads Präsenz 56 h Übungsbetrieb/Prüfungsvorbereitung 124 h Summe 180 h
Turnus des Moduls	i. d. R. angeboten alle 2 Semester
Voraussetzung für die Teilnahme	Keine ☐ Folgende Inhaltliche Voraussetzungen: Software-Projekt
Lehr- und Lernformen	Seminar ☐ Vorlesung ☒ Tutorium ☒ Praktikum ☐ Projekt ☐
Lernziele	Die Studierenden verfügen über folgende Fachkompetenzen: • auf welchen Ebenen man Code analysieren kann, • wie man Schwachstellen des Codes auffindet, • wie man duplizierten Code automatisch aufspürt, • wie man Abhängigkeiten zwischen Anweisungen nachverfolgen kann • wie man Code-Muster findet, • wie man den Code automatisch transformieren kann, • wie man die Stellen im Code findet, die eine bestimmte Funktionalität implementieren, • wie man Vererbungshierarchien restrukturieren kann, • wie man Software visualisieren kann, • wie man Software-Architekturen rekonstruiert • wie man Reengineering-Projekte organisiert.

Lerninhalte	Software-Reengineering beschäftigt sich mit Wiedergewinnung verlorener Informationen über existierende Software-Systeme (Reverse Engineering), Restrukturierung der Beschreibung des Systems (Restructuring) und der nachfolgenden Implementierung der Änderungen (Alteration). Reengineering hat dabei nicht nur mit alter Software zu tun; gerade neuere objekt-orientierte Systeme erfordern oft schon bald eine Restrukturierung, weshalb sich ein guter Teil der Vorlesung speziell objekt-orientierter Software widmet (Restrukturierung von Klassenhierarchien, automatisches Refactoring). Auch im Kontext neuerer Ansätze des Software-Engineerings zur Entwicklung ähnlicher Produkte als Produktlinie findet Reengineering Einsatz. • allgemeiner Überblick über das Thema sowie Beziehung des Reengineerings zu verwandten Gebieten der Software-Wartung, Wrapping, etc. • Zwischendarstellungen für Programmanalysen (abstrakte Syntaxbäume, Program Dependency Graph, Static Single Assignment Form), Datenfluss-/Kontrollflussanalysen • Software-Metriken • Software-Architekturrekonstruktion: Reflexionsmethode, Software-Clustering, Symphony • Program Slicing • Klonerkennung • Mustersuche • automatische Code-Transformationen und Refactoring • Begriffsanalyse • Merkmalsuche • Analyse und Restrukturierung von Vererbungshierarchien • Software Visualisierung • Planung und Durchführung von Reengineering-Projekten, Prozessmodelle des Reengineerings Die Übungen dienen, neben der Wiederholung und praktischen Vertiefung des Vorlesungsinhalts, auch der Vorstellung existierender Reengineering-Werkzeuge. Die Vorlesung Software-Reengineering beschäftigt sich mit der Methodik des systematischen Informationengewinns über existierende Programme, die formale Repräsentation von Programmen sowie mit Methoden für semantikerhaltende Transformationen von Programmen. Die in der Vorlesung dargestellte formale Begriffsanalyse bildet eine mathematisch fundierte Methode zur Analyse verschiedener Relationen in Programmen, die auch in anderen Gebieten der Informatik eingesetzt werden kann.
Prüfungsformen	i.d.R. Bearbeitung von Übungsaufgaben und Fachgespräch oder mündliche Prüfung

Literatur	Reengineering • Reengineering - Eine Einführung, Bernd Müller, B.G. Teubner Verlag Stuttgart, 1997 • Object Oriented Reengineering Patterns, Serge Demeyer, Stephane Ducasse, Oscar Nierstrasz, 2007. • Refactoring: Improving the Design of Existing Code, Martin Fowler, Addison-Wesley, 2000. • Modernizing Legacy Systems , Robert C. Seacord, Daniel Plakosh, and Grace A. Lewis. Addison-Wesley, 2003. • Anti Patterns: Entwurfsfehler erkennen und vermeiden, William J. Brown (Autor), Raphael C. Malveau, Mitp-Verlag; zweite überarbeitete Auflage, 2007. Wartung und Evolution • Legacy-Software, Dieter Masak, Springer Verlag, 2006. Prozesse und Management zur Wartung und Migration von Altsystemen. • Nutzung und Wartung von Software - Das Anwendungssystem-Management, Franz Lehner, Hanser Verlag, 1989. • Software-Produktmanagement: Wartung und Weiterentwicklung bestehender Anwendungssysteme Harry M. Sneed, Martin Hasitschka, Maria-Therese Teichmann, Dpunkt Verlag, 2004. • Software Evolution, Tom Mens, Serge Demeyer (Eds.), Springer Verlag, 2008. • Software-Wartung: Grundlagen, Management und Wartungstechniken, Christoph Bommer, Markus Spindler, Volkert Barr, DPunkt Verlag, 2008. • Practical Software Maintenance: Best Practices for Managing Your Software Investment, Thomas M. Pigoski, Wiley & Sons, 1996. Wartbarkeit • Code Quality Management: Technische Qualität industrieller Softwaresysteme transparent und vergleichbar gemacht, Frank Simon, Olaf Seng, Thomas Mohaupt, Dpunkt Verlag, 2006. • Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems von Michele Lanza und Radu Marinescu, Springer Verlag, 2006, ISBN-13 978-3540244295. Programmanalyse • Advanced Compiler Design and Implementation, Steven S. Muchnick, Morgan Kaufmann, 1997. • Principles of Program Analysis, Flemming Nielson, Hanne Riis Nielson, Chris Hankin, Springer Verlag, Auflage: 2., 2004. Software-Visualisierung • Software Visualization, Stephan Diehl, Springer Verlag, 2007. Debugging • Why Programs Fail: A Guide to Systematic Debugging, Andreas Zeller, Dpunkt Verlag, 2005.