A107: Programmierparadigmen

ECTS Credits: 7

Workload: 210

Kontaktstunden: 80

Semester: 5, 6

Prüfungssemester: 6

ModulnameEnglisch: Programming Paradigms

Modulverantwortlicher

Prof. Dr. Joachim Sauer

Lehrende

Prof. Dr.-Ing. Johannes Brauer

Positionierung im Studiengang

Das Modul ergänzt die in anderen Programmiermodulen ausgiebig thematisierte Objektorientierung um einen Einstieg in das prädikative Programmierparadigma und Parallelprogrammierung und erweitert die Kenntnisse in funktionaler Programmierung.

Qualifikationsziele und Lernergebnisse

Absolventen besitzen ein Verständnis von mathematischen und formalen Grundlagen der Informatik

- Absolventen des Moduls kennen die mathematischen Grundlagen der funktionalen und prädikativen Programmierung.

Absolventen sind in der Lage, Problemlösungen algorithmisch zu formulieren und in verständliche und effiziente Computerprogramme umzusetzen

Nach Abschluss des Moduls haben die Studierenden die Fähigkeit erlangt,

- korrekte Programme systematisch zu entwickeln,
- fortgeschrittene Programmiertechniken anzuwenden und
- den Nutzen verschiedener Programmierstile zu erkennen.

Absolventen können professionelle Methoden und Werkzeuge zur Softwareentwicklung einsetzen

- Absolventen des Moduls kennen innovative Methoden der professionellen Softwareentwicklung.

Lerninhalte

Gegenüberstellung der klassischen Programmierparadigmen:

- imperative Programmierung
- funktionale Programmierung
- prädikative Programmierung

Wiederholung und Ausbau der Grundlagen der funktionalen Programmierung:

- Funktionsbegriff / funktionale Abstraktion
- Programmieren mit Funktionen
- Methodische Abstraktion: Entwurfsvorschriften
- Funktionen höherer Ordnung

Vertiefung der Kenntnisse in funktionaler Programmierung:

- Auswertungsreihenfolge
- Umgang mit unendlichen Datenstrukturen
- Effizienz von funktionalen Programmen
- Definition und Einsatz eigener Datentypen
- Weitere funktionale Programmiertechniken

Prädikative Programmierung:

- Grundlagen
- Programmiertechniken
- Anwendungen

- Prolog

Erweiterung der prädikativen Programmierung:

- Constraint-Programmierung

Parallelprogrammierung:

- Unterschiede zwischen konkurrierenden und parallelen Prozessen
- Synchrone vs. asynchrone Prozesse
- Typische Probleme der Parallelprogrammierung

Lehr- und Lernmethoden

- Vorlesung im seminaristischen Stil
- Gruppen- und Einzelübungen
- Abgabe der Lösungen von entsprechend gekennzeichneten Übungsaufgaben (max. vier) ist Voraussetzung für die Teilnahme an der Klausur.

Vorlesungs-/Prüfungssprache: Deutsch/Deutsch

Leistungsbewertung: Klausur

Literatur/Lehrmaterial

- Folien der Lehrveranstaltung
- Online-Manuals der verwendeten Werkzeuge
- Literaturliste, z. B.:
  - Abelson, Sussman, Sussman: Structure and Interpretation of Computer Programs
  - Bengel, Baun, Kunze, Stucky: Masterkurs Parallele und Verteilte Systeme
  - Felleisen, Findler, Flatt, Krishnamurthi: How to Design Programs
  - Klaeren, Sperber: Die Macht der Abstraktion
  - Bramer: Logic Programming with Prolog
  - Rauber, Rünger: Parallele Programmierung
  - Thom Frühwirth, Slim Abdennadher. Essentials of Constraint Programming

Erforderliche Vorkenntnisse

- Objektorientierte Programmierung, z. B. durch I166: Einführung in die objektorientierte Programmierung
- Grundlagen der funktionalen Programmierung, z. B. durch I167: Einführung in die Programmierung

Vormodule

I145 Diskrete Mathematik 1, A100 Formale Grundlagen der Informatik, I143 Praxis der Softwareentwicklung

Verflechtung mit der betrieblichen Praxis

Abteilungen / Unternehmensbereiche:

- Softwareentwicklung
- Qualitätsmanagement
- Softwareprojekte

Kernfragen / Themengebiete:

- Wie findet Qualitätssicherung der Software im Unternehmen statt?
- Lässt sich das Abstraktionsniveau bei der Software-Entwicklung erhöhen und was könnte der Nutzen sein?
- Wie relevant sind Entwurfsvorschriften in der betrieblichen Praxis?

Beispiele für aktives Mitwirken / Unterstützung:

- Schreiben funktionaler Programme/Programmteile
- Schreiben prädikativer Programme/Programmteile