

Modulbeschreibung:

Fortgeschrittene Programmierparadigmen

Allgemeine Information

Anzahl ECTS-Credits

3

Modulkürzel

TSM_AdvPrPa

Version

19.02.2015

Modulverantwortliche/r

Dr. Edgar Lederer, FHNW

Sprache

	Lausanne	Bern	Zürich
Unterricht	<input checked="" type="checkbox"/> E <input checked="" type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input checked="" type="checkbox"/> D <input type="checkbox"/> E
Unterlagen	<input checked="" type="checkbox"/> E <input checked="" type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input checked="" type="checkbox"/> D <input checked="" type="checkbox"/> E
Prüfung	<input type="checkbox"/> E <input checked="" type="checkbox"/> F	<input type="checkbox"/> D <input type="checkbox"/> E <input type="checkbox"/> F	<input checked="" type="checkbox"/> D <input type="checkbox"/> E

Modulkategorie

- Erweiterte theoretische Grundlagen
- Technisch-wissenschaftliche Vertiefung
- Kontextmodule

Lektionen

- 2 Vorlesungslektionen und 1 Übungslektion pro Woche

Kurzbeschreibung / Absicht und Inhalt des Moduls in einigen Sätzen erklären

Neben der allgegenwärtigen objektorientierten Programmierung und den ungenügenden Testverfahren ist eine ganze Fülle an faszinierenden Technologien vorhanden. In diesem Modul werden die Studierenden aus einer allgemeinen Programmiersprachenperspektive in die wichtigsten aufkommenden Technologien eingeführt.

Paradigmen Neben der Objektorientierung als dem am meisten verbreiteten Programmierparadigma wurden in den vergangenen Jahrzehnten andere, ganz unterschiedliche *Paradigmen* entwickelt und zur Reife gebracht. Dazu gehören insbesondere: die *funktionale* Programmierung, aber auch die logische Programmierung und die Constraintprogrammierung. Keines dieser Paradigmen (einschliesslich der Objektorientierung) ist für die Lösung aller Arten von Problemen gut geeignet, besitzt jedoch seine bestimmten Anwendungsgebiete, in denen es seine Stärken ausspielen kann. Da moderne Software viele solche Gebiete umfasst, erscheint die simultane Anwendung mehrerer Paradigmen und ihre nahtlose Integration in *Multiparadigmen*-Programmiersprachen angebracht.

Types Programmiersprachen mit einem reichhaltigen und konsistenten Typsystem erlauben es, gewisse Fehler bereits zur Übersetzungszeit zu erkennen. Mit dem Typsystem können in eigenen Datentypen Invarianten deklariert werden, die dann vom Compiler geprüft werden. Programmierung in und mit einem starken Typsystem kann als erster Schritt in Richtung Programmverifikation betrachtet werden.

Korrektheit Die Auswahl des richtigen Programmierparadigmas für ein gegebenes Problem erleichtert zwar seine Lösung, garantiert jedoch nicht seine *Korrektheit*, die wichtigste aller Softwarequalitäten. Eine solche Garantie verlangt zusätzlich zur eigentlichen Implementation (das "Wie?") eine Spezifikation (das "Was?") und den Beweis der Korrektheit (das "Warum?"). Kontinuierliche Forschung seit den Anfängen der Computertechnik hat zu einer Verifikationstechnik geführt, die nun in industriellen Anwendungen Einzug hält. Da die *objektorientierte* Programmierung allgegenwärtig ist, kommt ihrer Spezifikation und Verifikation besondere Bedeutung zu.

Dieses Modul bietet:

- einen Überblick über Programmierkonzepte, -paradigmen und -sprachen;
- eine umfassende Einführung in die funktionale Programmierung (am Beispiel von Haskell oder Scala);

- eine Einführung in die Multiparadigmen-Programmierung mit besonderer Betonung auf Typen (unter Verwendung von Scala, einer Kombination aus funktionaler und objektorientierter Programmierung);
- eine Einführung in die Theorie und Praxis der Spezifikation und Verifikation von imperativen Programmen als Basis für die Verifikation von objektorientierten Programmen (mit Beispiel Dafny).

Ziele, Inhalt und Methoden

Lernziele, zu erwerbende Kompetenzen

Die Studierenden eignen sich ein Verständnis für die aufkommenden Paradigmen, praktische Fähigkeiten in der modernen, funktionalen, *type-full* und multiparadigmatischen Programmierung und ein grundlegendes Verständnis für den immer wichtiger werdenden Bereich der Softwarespezifikation und -verifikation an.

Modulinhalt mit Gewichtung der Lehrinhalte

Funktionale Programmierung (6 Wochen)

- Programmierkonzepte, -paradigmen und -sprachen.
- Fehlen von Zustand, referenzielle Transparenz, Argumentation über Programme.
- *Eager* im Vergleich zu *lazy evaluation*.
- Typen und Typinferenz.
- Funktionen höherer Ordnung.
- Konkrete Datentypen und *pattern matching*.
- Eine Anwendung: Interpreter für eine kleine imperative Programmiersprache.

Multiparadigmen- und stark getypte Programmierung (4 Wochen)

- Trait Typen (und Mixin Komposition als Variante zur klassischen Vererbung).
- Generische Typen (Ko- und Kontravarianz für Typparameter).
- Typklassen und implizite Parameter.
- Typsichere DSLs (*Domain Specific Languages*).

Programmverifikation (4 Wochen)

- Zuverlässigkeit über Testen und Verifizieren.
- Hoare-Logik und schwächste Vorbedingungen.
- Architektur von Verifikationswerkzeugen.
- Eine Anwendung: Verification Condition Generator.
- Ein aktuelles Verifikationswerkzeug: Dafny.

Lehr- und Lernmethode

- Frontalunterricht.
- Programmier- und Verifikationsübungen.

Voraussetzungen, Vorkenntnisse, Eingangskompetenzen

- Gute Arbeitskenntnisse der objektorientierten Programmierung.

Bibliografie

- Graham Hutton, Programming in Haskell, Cambridge, 2008.
- Miran Lipovaca, Learn You a Haskell for Great Good!, No Starch Press, 2011.
- Martin Odersky, Lex Spoon and Bill Venners, Programming in Scala, Artima, 2008.
- David Gries, The Science of Programming, Springer, 1981 (ein klassischer Text).
- José Bacelar Almeida et al., Rigorous Software Development, Springer, 2011.
- Federico Biancuzzi und Shane Warden, Masterminds of Programming: Conversations with the Creators of Major Programming Languages, O'Reilly, 2009 (zur Entspannung).

Leistungsbewertung

Zulassungsbedingungen für die Modulschlussprüfung (Testatbedingungen)

Schriftliche Modulschlussprüfung

Prüfungsdauer:	120 Minuten.
Erlaubte Hilfsmittel	Alle schriftlichen Unterlagen, aber keinerlei elektronische Geräte.