ePS2017 - Workshop 2

Ergebnisprotokoll der Diskussion

Moderation: Michael Striewe, Benjamin Otto
Protokoll: Michael Stiewe und mehrere Teilnehmende

Variabilität in Programmieraufgaben

Michael Striewe, Benjamin Otto

- Ziele:
  - Vermeiden von Abschreiben
  - Anpassung an Individualität im Lernprozess
- Vorerfahrungen:
  - Parametrisierte Aufgaben in der Mathematik
  - Programmieraufagben automatisiert testen: (JACK (https://jack-community.org))
    - Statisch Unit Tests
    - Dynamisch:
  - Bachelorarbeit HS Hannover
- Fragen:
  - Welche Software / System werden eingesetzt / sind einsetzbar?
    - Antwort (Werbung): https://www.waxmann.com/buch3606 (Open Access)
      - Danke!!! ;-)
    - VPL, the Virtual Programming lab for Moodle (http://vpl.dis.ulpgc.es/)
    - Grader- und LMS-übegreifendes Aufgabenformat für Programmieraufgaben (ProFormA): https://eleed.campussource.de/archive/11/4138 und https://github.com/ProFormA/taskxml
  - Lassen sich Methoden aus Continious Integration verwenden?
  - Welche Artefakte sind von Variabilität betroffen: Aufgabentext? Vorgegebenes Programmfragment? Bewertungs-(Unittest-)Code? Feedbacktexte? Struktur des Bewertungsschemas? Interne Erläuterungen der Aufgabe (didaktische Hintergrundinfos)? ...?
    - Individuelles Feedback ist wichtig und großer Mehrwert!
    - Abhängigkeiten zwischen Variationspunkten entstehen sehr schnell
  - Lassen sich diese Artefakte automatisiert generieren?
    - Vorab oder on-the-fly in der Prüfung?
      - On-the-fly-Erzeugung wird im Übungsbetrieb von Studierenden sehr geschätzt
  - Verschiedene Ebenen der Variabilität

        - "Low-Level": Bezeichner & etc. austauschen -> Klausurtauglich, weil leicht gleichwertig zu gestalten. Aber Gefahr: Wechsel der Domäne (Kreisfläche / Dreicksfläche, ...) unterschiedlich schwer?
            - -> Lösungsvorschlag: Aufgabentext variabel gestalten. Formel für Kreisfläche, Dreiecksfläche, etc. vorgeben -> Dann wieder vergleichbar
        - "High-Level": Algorithmen etc.
        - Variabilität in den Qualitätsanforderungen an die Lösung
    - Notation von Variabilität
        - Können Authoring-Tools helfen? Programmierer können vielleicht auch JSON o.ä.
    - Was können LMS allgemein in diesem Bereich schon?
        - Zufällige Parametrisierung ist möglich, komplexere Adaptivität bisher kaum
- Probleme
    - In summativen Tests (Zeitdruck?) kompiliert der Code tlw. nicht
    - Wie kann man die Schwierigkeit von Programmieraufgaben überhaupt messen?
        - Schwierigkeit ist relativ zur Lehre
- Lösungsideen:
    - Variabilität über Interfaces
    - Variation der Schwierigkeit über Menge an vorgegebenem Code
    - Schwierigkeit pro Aufgabe und Offset/Faktor (?) pro Variante
    - Typisierung von Variationspunkten
        - In manchen zieht man zufällig aus einer (kleinen) Liste
- Schritte
    - Format definieren, in dem man Variabilität maschinenlesbar aufschreiben kann
        - Key-Value?
    - Anforderungen priorisieren, welche Art von Variabiltät man in der Praxis am häufigsten braucht

Weitere Werbung:

- GI Fachgruppe E-Learning
    - Workshop "Automatische Bewertung von Programmieraufgaben" (ABP) 5./6. Oktober in Potsdam
    - Jährliche Konferenz DeLFI (2018 in Frankfurt)
- Konferenz "Hochschuldidaktik Informatik" (HDI), 2jährig

Anderes Beispiel:
Write a class Student in the package de.hsh that stores a name and a matriculation number. Provide a constructor and a toString method. Your class should reject illegal, i. e. negative matriculation numbers. Test your class using the following client code:

- Student o1= new Student("Smith", 68930);
- System.out.println(o1); // prints Smith (68930)
- Student o2= new Student("Smith", -1); // throws IllegalArgumentException
-

Variante:
Write a class State in the package net.geo that stores a state name and a number of residents. Provide a constructor and a toString method. Your class should reject illegal, i. e. negative numbers of residents. Test your class using the following client code:

- State o1= new State("Alabama", 4779736);
- System.out.println(o1); // prints Alabama [4779736]
- State o2= new State("Alabama", -1); // throws RuntimeException