

Flexibles E-Assessment auf Basis einer Service-orientierten Architektur

Konzepte, Implementierung und Praxiserfahrungen

Mario Amelung Katrin Krieger Dietmar Rösner

Otto-von-Guericke-Universität Magdeburg



DeLFI 2009

Motivation

- ▶ Schaffung zusätzlicher Übungsmöglichkeiten
 - ▶ Schaffung zeit- und ortsunabhängiger Lernmöglichkeiten
 - ▶ Förderung von Motivation und Interesse
 - ▶ Verbesserung der Lernerfolgskontrolle
 - ▶ Reduzierung des Aufwands seitens der Lehrenden
- ▶ Intensivierung und effizientere Gestaltung des Übungsbetriebs

Motivation

- ▶ Schaffung zusätzlicher Übungsmöglichkeiten
- ▶ Schaffung zeit- und ortsunabhängiger Lernmöglichkeiten
- ▶ Förderung von Motivation und Interesse
- ▶ Verbesserung der Lernerfolgskontrolle
- ▶ Reduzierung des Aufwands seitens der Lehrenden

- ▶ **Intensivierung und effizientere Gestaltung des Übungsbetriebs**

Stand der Technik

Vielzahl von Systemen, die

- ▶ sich auf bestimmte Programmiersprachen und/oder Aufgabentypen beschränken
- ▶ z. B. Scheme-robo, AutoGrader, Trakla2

Einige Systeme, die

- ▶ prinzipiell für beliebige Programmiersprachen nutzbar sind, da die Überprüfung durch Module implementiert wird
- ▶ z. B. CourseMarker, BOSS, AT(x)-Framework

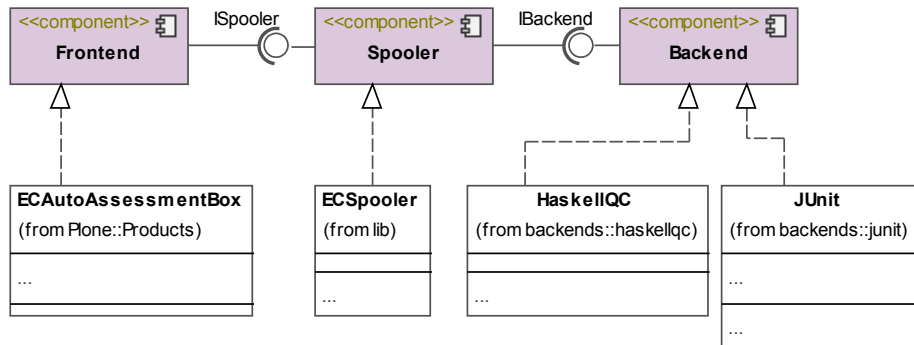
Anforderungen

- ▶ flexible Integration der Test- und Bewertungsfunktionalität in bestehende Lernumgebungen
- ▶ einfache Erweiterbarkeit
 - ▶ zusätzliche Aufgabenformen
 - ▶ Programmiersprachen
 - ▶ Testmethoden
- ▶ automatische Auswertung von Programmierlösungen
 - ▶ verschiedenen Programmiersprachen
 - ▶ unterschiedlichen Testmethoden
- ▶ automatische Auswertung von Aufgaben in anderen formalen Systemen

Überblick



Realisierung



Frontend ECAutoAssessmentBox

Default Categorization Dates Ownership Settings **Backend**

Automatically accept assignments

If selected, an assignment which passes all tests will be automatically accepted.

Tests

Select one or more tests.

JUnitTests

Unit tests

Enter one or more JUnit tests. If you have to reference the class of the submission use variable "\${CLASS}". [e.g. \${CLASS} myClass = new \${CLASS}(); or \${CLASS}.method()]

```
@Test public void encodeChar(){
    ${CLASS} stud = new ${CLASS}();
    assertEquals('b', stud.encode('a', 1));
}

@Test public void encodeEncode(){
    ${CLASS} stud = new ${CLASS}();
    assertEquals('a', stud.encode(stud.encode('a', 5), -5));
}
```

Frontend ECAutoAssessmentBox

Answer:

```
public class LittleMath
{
    /**
     * @param x The input value.
     * @return The absolute value of x.
     */
    public int abs(int x)
    {
        if (x < 0) return x;
        return x;
    }
}
```

 [kate.20080118.132047.txt \(Plain Text OKb\)](#)

Auto feedback:

```
testAbs(JUnitTester): 'abs(-3)' failed.
```

The absolute value of a real number x is defined as the unsigned portion of x .

```
expected:<3> but was:<-3>
```

by [Kate Milliken](#) — last modified 2008-01-18 13:20

Backend-Erstellung

- ▶ »Was soll getestet werden?«
- ▶ »Wie soll getestet werden?«
- ▶ Eingabe und Ausgabe
- ▶ Compiler und Interpreter
- ▶ Sicherheitsaspekte

Einsatz an der OvGU Magdeburg

- ▶ Backends u. a. für Haskell, Scheme, CommonLisp, Erlang, Prolog, Python und Java
- ▶ Wintersemester 2008/09
 - ▶ »Algorithmen und Datenstrukturen« (300 Studierende)
 - ▶ »Funktionale Programmierung« (30 Studierende)
- ▶ Sommersemester 2009
 - ▶ »Algorithmen und Datenstrukturen« (300 Studierende)
 - ▶ »Programmierparadigmen« (60 Studierende)
- ▶ Allein in AuD 170 Aufgaben mit mehr als 24.000 automatisch überprüften studentischen Einreichungen.

Weitere Anwender

- ▶ Universität Rostock
 - ▶ Einsatz der Backends für Haskell und Java
 - ▶ Winter 2007: »Abstrakte Datentypen« (200 Studierende)
 - ▶ Sommer 2007: »Algorithmen und Datenstrukturen« (140 Studierende)
- ▶ LMU München
 - ▶ Entwicklung eines eigenen Backends für *SML*
 - ▶ Sommer 2008: »Programmieren und Modellieren« (200 Studierende)

Zusammenfassung und Ausblick

- ▶ Flexibilität durch Service-orientierten Ansatz
 - ▶ Anbindung an beliebige Frontends
 - ▶ Entwicklung eigener Backends
- ▶ Praktische Anwendung in der Informatiklehre
- ▶ Backends für nichttextuelle Einreichungen, z. B. *UML*
- ▶ Evaluierung anderer Frontends

Zusammenfassung und Ausblick

- ▶ Flexibilität durch Service-orientierten Ansatz
 - ▶ Anbindung an beliebige Frontends
 - ▶ Entwicklung eigener Backends
- ▶ Praktische Anwendung in der Informatiklehre
- ▶ Backends für nichttextuelle Einreichungen, z. B. *UML*
- ▶ Evaluierung anderer Frontends

Weitere Information

- ▶ Web-Seite:

<http://wdok.cs.uni-magdeburg.de/eduComponents/>

- ▶ Software:

<http://wdok.cs.uni-magdeburg.de/software/>

- ▶ Demo-Server:

<http://wdok.cs.uni-magdeburg.de/demo/>