Lehransatz: Selbständiges Erlernen von Programmiersprachen mit automatischen Tests und graphischer Oberfläche

Prof. Dr. Barbara Dörsam

1. Lehrkonzeption

a. Pfeiler des Lehrkonzeptes

Mein Lehrkonzept zur Veranstaltung „Softwareentwicklung für Ingenieure II" im Studiengang Druck- und Medientechnologie (DT) sieht für die praktischen Übungen der Studenten eine Plattform zum selbständigen Erlernen einer Programmiersprache vor. Die Plattform wurde von mir mit drei Zielsetzungen entwickelt:

- **Motivation**: Studenten finden durch die Plattform Spaß am Programmieren und lösen die Übungsaufgaben, weil sie das Ergebnis sehen wollen und nicht, weil sie z.B. Punkte dafür bekommen.
- **Qualität**: Die Programmierqualität wird durch Einführung automatisierter Tests gesteigert, da die Studenten „nebenbei" lernen, ihre eigenen Programme ausführlich zu testen und somit eigene Ergebnisse kritisch zu hinterfragen.
- **Schnelligkeit**: Die Studenten haben die Möglichkeit, eine Programmiersprache schnell und zeitgemäß im Selbststudium zu erlernen, da sie auch außerhalb der regulären Übungstermine ihre Programmieraufgaben bearbeiten und ihre Lösungen eigenständig auf Korrektheit überprüfen können.

b. Skizze der Lehr-/Lernveranstaltung sowie ihr Innovationsgehalt

Die Veranstaltung „Softwareentwicklung für Ingenieure II" besteht aus zwei Teilen: der Vorlesung mit theoretischen Inhalten und dem zugehörigen praktischen Übungsteil. Innovativ ist dabei der Aufbau und die Durchführung der Übungen. Die Studenten bekommen nicht – wie sonst üblich – nur ein Übungsblatt, das sie bearbeiten müssen, sondern insgesamt drei Aufgabenbestandteile:

1. *Das Übungsblatt mit der Aufgabenstellung*: Hierbei werden keine abstrakten Programmieraufgaben gestellt, sondern das Übungsblatt enthält Anleitungen für Anwendungen, die eine bekannte Funktionsweise aus dem Alltag widerspiegeln: z.B. ein Lottospiel, ein Radio oder zu Weihnachten eine Lichterkette. Die Lösung der Aufgaben erfordert die inhaltlichen Kenntnisse aus der Vorlesung, jedoch müssen die Studenten nicht erst den abstrakten Anwendungsfall aus der Aufgabe und dessen Sinnhaftigkeit verstehen, um diese lösen zu können.
2. *Eine bereits implementierte graphische Oberfläche, in welche die Studenten ihre Aufgabenlösungen integrieren (Abbildung 1):* Bei konventionellen Programmierübungen müssten Studenten selbst ihre Benutzerschnittstelle implementieren. Da sie ihre Programmierkenntnisse erst erlangen, darf diese nicht zu komplex sein: meistens können sie nur einfache text-basierte Ein- und Ausgaben implementieren. Solche sind in der heutigen IT-Welt mit Smartphones, Tablets usw. gar nicht mehr anzutreffen.
   Mit einer graphischen Oberfläche verstehen die Studenten deutlich leichter die Aufgabenstellung, sehen die Zusammenhänge innerhalb der Programmieraufgabe und müssen sich nicht mehr mit veralteten und ungewohnten textuellen Ein- und Ausgaben für ihre Programme auseinandersetzen. Je ansprechender die graphische Oberfläche gestaltet ist, desto lieber beschäftigen sich die Studenten mit der Programmieraufgabe. Enthält die Aufgabenstellung zudem auch spielerische Aspekte (z.B. ein Lottospiel), steigert dies die Motivation zusätzlich.

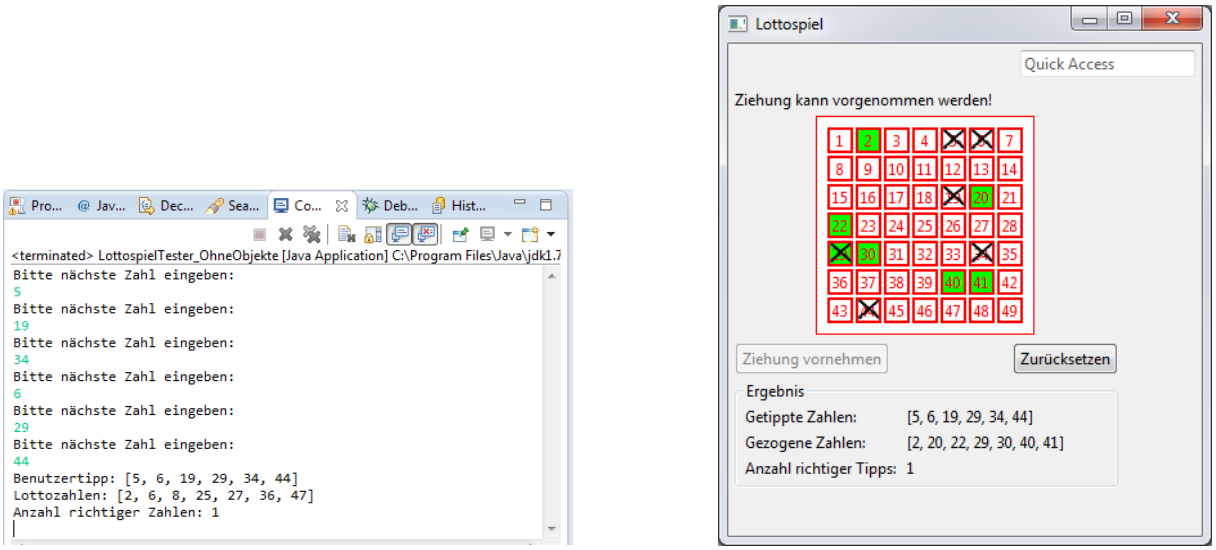

Abbildung 1: Konventionelle Ein- und Ausgabe für Programmierübungen (links) und die neu eingeführte graphische Oberfläche für die identische Programmieraufgabe "Lottospiel".

3. *Automatische Tests*: Ohne langwierige und im Rahmen einer Übungsstunde nicht für jeden Studenten durchführbare manuelle Tests können Studenten nun mit Hilfe bereits vorbereiterer automatisierter Tests ihre eigenen Programme selbständig überprüfen. Zudem sehen sie sofort, ob sie die Aufgabe tatsächlich fertig gelöst haben oder ob die Lösung noch Fehler enthält. Eine zusätzliche Motivation entsteht durch eine graphische Aufbereitung der Testergebnisse: so lange der Testbalken nicht grün ist (Abbildung 2), sind nicht alle Tests fehlerfrei durchgelaufen und die Aufgabenlösung ist noch nicht fertig.
   Durch dieses Konzept entwickelt sich zusätzlich ein gewisser Ehrgeiz unter den Studenten und ihre Motivation wächst: der grüne Balken in der Testanzeige wird schnell zum selbstdefinierten Ziel.

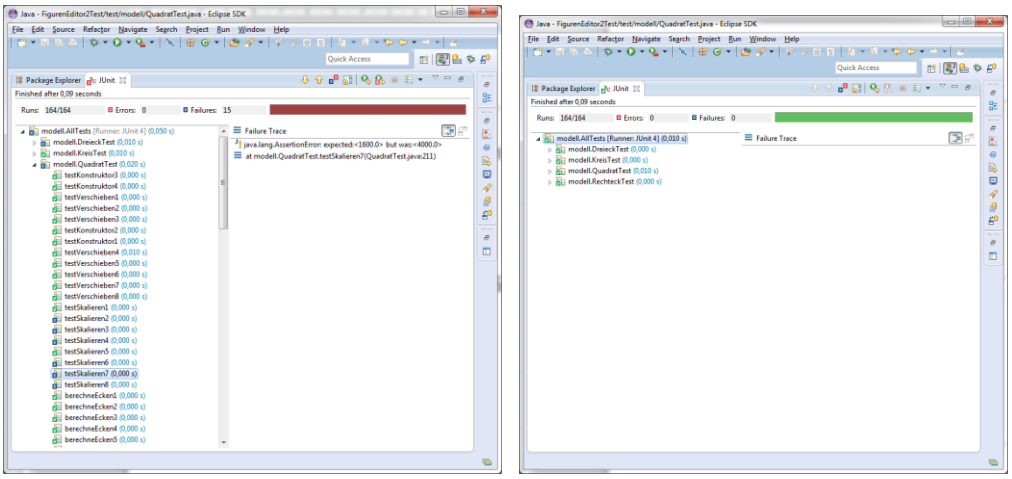

Abbildung 2: Graphische Darstellung der automatischen Tests. Links: der waagerechte Balken ist rot, der Student hat seine Aufgabe noch nicht fertig gelöst. Rechts: Der Testbalken ist grün, die Aufgabe ist fertig gelöst.

c. Historie des Konzeptes

Das Konzept (graphische Oberflächen) wurde im WS 11/12 zum ersten Mal eingesetzt, im Sommersemester 2012 wurden die ersten automatisierten Tests eingeführt. Seit dem Wintersemester 12/13 verfügen alle Aufgaben der Lehrveranstaltung über eine graphische Oberfläche und zugehörige automatische Tests.

d. Geplante Weiterentwicklung

Die Übungsumgebung soll künftig noch um weitere Funktionen erweitert werden, welche die Selbständigkeit und Motivation der Studenten noch stärker unterstützen sollen:

- Teilweise scheitern die Studenten beim Lösen von Programmieraufgaben an Kleinigkeiten, die durch einen kurzen Hinweis des Dozenten geklärt werden können. Um auch hier das selbständige Lernen ohne Anwesenheit des Dozenten zu fördern, sollen künftig in die Aufgabenstellung zusätzlich Musterlösungen integriert werden, die jedoch für die Studenten nicht vollständig sichtbar sind, sondern nur teilweise in die eigene Lösung eingebunden werden können. Dadurch können Studenten sich selbst an den Stellen des Programms weiterhelfen, an denen sie ohne Hilfe nicht weiterkommen, und sie sehen trotzdem nicht die gesamte Musterlösung auf einmal.
- Es wäre denkbar, die speziell für den Studiengang DT entwickelte Lösung – ggf. mit Anpassungen - auch anderen Studiengängen zur Verfügung zu stellen.

e. Zusammenhang mit dem Profil des Studiengangs DT

Die Lehrveranstaltung ist eine Pflichtveranstaltung im Studiengang DT und dient als Basis für weiterführende Veranstaltungen, z.B. diverse technologische Praktika, das Projektpraktikum oder die Veranstaltung „Entwicklung von Web-Anwendungen“. Studenten können ihre darin gewonnen Erkenntnisse in den weiterführenden Veranstaltungen einsetzen.

f. Vermittlung von Kompetenzen

- Fachkompetenzen:
  - Softwareentwicklung
  - Softwaretest
  - Umgang mit Softwareentwicklungstools, die in der Industrie eingesetzt werden.
- Selbstkompetenz:
  - Kritische Auseinandersetzung mit eigenen Ergebnissen: durch die automatischen Tests lernen die Studenten, dass ihre eigene Einschätzung ihrer Lösung mit der objektiv gemessenen Qualität der Lösung (Anzahl der gefundenen Fehler) nicht immer übereinstimmt.
  - „Hilfe zur Selbsthilfe“: die Studenten lernen, ihre Aufgaben selbständig zu bearbeiten und die Ergebnisse ohne fremde Hilfe zu überprüfen.
  - Sorgfalt und Genauigkeit: Die automatischen Tests sind so ausgelegt, dass sie auch die kleinsten Details einer Lösung auf Korrektheit überprüfen. Dadurch lernen die Studenten, ihre Aufgaben nicht nur oberflächlich zu lösen, sondern sich auch mit Details auseinanderzusetzen.
  - Zielstrebigkeit: die Studenten lernen, hartnäckiger an ihren Lösungen zu arbeiten und geben nicht so schnell auf wie bei Aufgaben ohne Tests.

2. Reflexion zur Lehrkonzeption

a. Gesellschaftlicher Gegenwartsbezug und Zukunftsbezug

- In der Industrie besteht immer noch hoher Bedarf an Absolventen der MINT-Fächer. IT-Fächer gelten aber als trocken und schwierig. Mit meinem innovativen Lehrkonzept lernen die Studenten spielend die Welt der Informatik kennen und gewinnen dadurch oft zusätzlichen Spaß an der IT.
- Studenten werden durch die besondere Form der Übungen auf ein inzwischen in der Industrie vorausgesetztes selbständiges „lebenslanges Lernen“ vorbereitet.

b. Teilnahme der Studierenden am Innovationsgehalt der Veranstaltung

Studenten arbeiten von der ersten praktischen Übung an mit dem innovativen Konzept, das sich deutlich von bisherigen konventionellen Programmierübungen abhebt. Durch die

graphische Oberfläche wird der Zugang zum Programmieren erleichtert. Die vollständig automatisierten Tests steigern zudem die Programmierperformance erheblich. Die Vorteile des Ansatzes aus Sicht der Studenten lassen sich in kurzen Worten zusammenfassen:

- Schnelles, zeitgemäßes Erlernen einer Programmiersprache.
- Steigerung der Programmierqualität durch automatisierte Tests.

c. Umgang mit unterschiedlichen Voraussetzungen der Studierenden

- Jeder Student kann in seinem Tempo arbeiten und die Aufgaben auch zu Hause selbständig bearbeiten und seine Ergebnisse überprüfen.
- Zusätzlich behalte ich als Dozentin in den Präsenzveranstaltungen immer den Überblick über den aktuellen Stand und kann bei Bedarf einspringen. Da die Studenten meine Unterstützung zur Überprüfung ihrer Ergebnisse nicht mehr unmittelbar benötigen, kann ich mich besser auf die Unterstützung der schwächeren Studenten konzentrieren und bin Ansprechpartner für wesentliche Probleme.
- Es gibt – neben den regulären Aufgaben – weitere Zusatzaufgaben, die auf freiwilliger Basis in der gleichen Art und Weise wie die regulären Aufgaben durch die Studenten bearbeitet werden können. Somit können die besseren Studenten ihre Kenntnisse vertiefen und die schwächeren Studenten zusätzlich üben.

d. Sicht auf die eigene Rolle als Dozentin

- Impulse geben, wie man das theoretisch Gelernte in die Praxis umsetzen kann. Dazu dienen die praxisorientierten Übungsbeispiele.
- Persönliche Unterstützung für die schwächeren Studenten, Förderung der besseren Studenten durch Zusatzaufgaben.
- Vorbildfunktion: was ich praktisch von Studenten verlange, muss ich auch selbst - mit Spaß und Freude an der Aufgabe - machen können.

3. Modellwirkung der Lehrkonzeption

a. Weitergabe eigener Erkenntnisse

- Vorstellung des Lehrkonzeptes auf dem Tag der Lehre im Dezember 2012
- Tipps an Studenten, wie sie das Gelernte und die Tools auch in anderen, weiterführenden Veranstaltungen einsetzen können.
- Einsatz eines abgewandelten (vereinfachten) Konzeptes bei den geplanten Kinder-forschertagen im Sommer 2013.

b. Mögliche Weiterentwicklung des innovativen Lehrkonzeptes

- Weiterentwicklung der heute existierenden Rahmensoftware für die graphische Oberfläche und die automatischen Tests durch Hiwis und Praktikanten.
- Umsetzung der weiteren (bereits im Kapitel 2d erwähnten) Erweiterungen durch Hiwis und Praktikanten.
- Einsatz des Ansatzes in ähnlichen Veranstaltungen durch andere Dozenten, um mehr Feedback aus der Lehrpraxis zu erhalten.