Automatische Abwicklung von Klausuren mit dem MVC

Helmar Gust[+]
Universität Osnabrück
Institut für Kognitionswissenschaften
49069 Osnabrück

Philipp Hügelmeyer[**]
Universität Osnabrück
Zentrum virtUOS
49069 Osnabrück

Robert Mertens[*]
Universität Osnabrück
Zentrum virtUOS
49069 Osnabrück

Claus Rollinger[++]
Universität Osnabrück
Institut für Kognitionswissenschaften
49069 Osnabrück

Zusammenfassung

Bei dem Minimalen Virtuellen Campus (MVC) handelt es sich um ein an der Universität Osnabrück entwickeltes Online-Tutor-System, mit dem Übungsaufgaben und Klausuren über das Internet durchgeführt und ausgewertet werden können. Der folgende Beitrag gibt einen kurzen Überblick über den Aufbau und die Funktionsweise des MVC. Darüber hinaus werden generelle Konzepte anhand konkreter Aufgabentypen diskutiert und beschrieben, auf welche Weise das System die zur Verfügung gestellten Daten auswertet. In einem Vergleich mit kommerziellen Lehr-/Lernplattformen, wie sie zurzeit an zahlreichen Universitäten zum Einsatz kommen, werden die Vorteile des MVC bezüglich der Klausurfunktionalität herausgestellt. Der Beitrag schließt mit einem Blick auf die künftigen Entwicklungsziele und deren Realisierungsmöglichkeiten.

1. Einführung

Die Grundlage dieser Arbeit bildet der MVC (Minimaler Virtueller Campus). Der MVC ist eine Weiterentwicklung des internetbasierten VC-Prolog-Tutor-Systems [Pey99] das im Rahmen des Projektes „Virtueller Campus Hannover-Hildesheim-Osnabrück“ (VC) an der Universität Osnabrück entwickelt wurde. Das System wird aktuell in Lehrveranstaltungen der Studiengänge Cognitive Science, Betriebswirtschaftslehre, Computerlinguistik und Künstliche Intelligenz eingesetzt, kann aber auch als universeller Rahmen für internetbasierte Lehr- und Lernplattformen mit integrierter Verwaltung eines Übungsbetriebes dienen.

Dabei bietet das System neben allgemeinen Aufgabentypen wie Ankreuz- und Lückentextaufgaben auch Programmieraufgaben, die von dem System evaluiert und bewertet werden können. In diesem Papier soll vorrangig auf die automatische Aufgabenevaluation des vorgestellten Systems eingegangen werden. Die Flexibilität

[+] Helmar.Gust@uni-osnabrueck.de
[**] Philipp.Huegelmeyer@uni-osnabrueck.de
[*] Robert.Mertens@uni-osnabrueck.de
[++] Claus.Rollinger@uni-osnabrueck.de

der vorhandenen Aufgabentypen ermöglicht dabei auch eine Einordnung des Systems in den universitären Prüfungskontext.

2. Funktionalität des Systems

Der MVC ist ein Client-Server-System, das als universeller Rahmen einer internetbasierten Lehr- und Lernplattform konzipiert ist. Dabei dient den Studierenden der Browser als Client. Es werden keine weiteren Applikationen auf dem Client benötigt. Derzeit wird die komplette Logik des Systems auf einem zentralen Server pro Kurs abgebildet. Dabei ist es problemlos möglich, mehrere Kurse auf einem Server laufen zu lassen. Lehrenden bietet das System sowohl eine komplette Kursverwaltung, einschließlich Benutzer-, Aufgaben- und Gruppenverwaltung als auch eine Korrekturumgebung für die gelösten Aufgaben.

Dabei wird beim Anmelden des Benutzers an das System jedem Nutzer eine eigene Session zur Verfügung gestellt. Die Daten des Benutzers sowie die Aufgaben werden im Dateisystem abgebildet, es ist jedoch auch möglich, diese in einer SQL-Datenbank (PostgreSQL) abzulegen. Der Zugang zu den Aufgaben ist über personen- und gruppenspezifische Zugriffsrechte reglementiert.

Die Bearbeitung der Aufgaben erfolgt vollständig über das MVC-System. Nach einer möglichen Korrektur durch einen menschlichen Tutor, wie zum Beispiel bei Aufgaben mit Freitexteingabe, können Ergebnisse, Kommentare und Korrekturen auf die gleiche Weise wieder empfangen werden. Die programmiersprachenspezifischen Aufgaben werden serverseitig mit einem Prozess der entsprechenden Programmiersprache ausgewertet. Derzeit unterstützt das System Prolog und Lisp, an der Unterstützung weiterer Sprachen wird gearbeitet.

3. Grundlagen der Implementation

Um den heterogenen Umgebungen gerecht zu werden, in denen Studierende mit Computern arbeiten, sollte die Lehrplattform auf möglichst vielen Plattformen laufen. “Your applications are portable accross multiple platforms. Write your applications once, and you never need to port them - they will run without modification on multiple operating systems and hardware architectures.” [Gos95] Im Rahmen des Forschungsprojektes Virtueller Campus wurden dazu mehrere Programmversionen entwickelt.

Die erste Version verwendete Client-seitig ein Java-Applet, das den Studierenden als Editor diente. Auf der Server-Seite wurde ein spezieller Server zur Datenbankkommunikation und Auswertung der Programme benötigt. Bei der Verwendung dieser Version traten eine Reihe von Problemen auf. So lief das eingesetzte Applet nur mit einigen Browserversionen korrekt und auf unterschiedlichen Plattformen wurde das Applet unterschiedlich dargestellt. Das Applet musste auf Seite des Clients auf zusätzliche Betriebssystemressourcen zugreifen. Die damit verbundene erweiterte Signatur des Applets führte zu Schwierigkeiten, wie in [Pey98] beschrieben. Als weiteres Problem kristallisierte sich die Größe des Applets heraus, die über die bei Studenten weit verbreiteten Modem/ISDN-Verbindungen zu sehr langen Downloadzeiten führte. Darüber hinaus brachte der selbst implementierte Server einen sehr hohen Wartungsaufwand mit sich.

Als Reaktion darauf wurde eine zweite Version mittels Java-Servlets implementiert. Der Betrieb dieser Version wurde eingestellt, nachdem Fehler auf der Server-Seite wiederholt zum kompletten Absturz des Servers geführt hatten.

Aus Stabilitätsgründen wurde schließlich eine dritte rein serverseitige Version konzipiert, die auf der Server-Seite nur auf Skriptsprachen setzt und damit eine Reihe

der zuvor erkannten Fehlerquellen ausschließt. Mit Hilfe eines gut durchdachten Sessionmanagements ist es gelungen, das System so aufzubauen, dass im laufenden Betrieb Änderungen vorgenommen werden können. Dies reduziert die Auswirkungen möglicher Implementationsfehler auf ein Minimum.

Im Rahmen einer Weiterentwicklung wird an einer Modularisierung der Plugins für Programmieraufgaben und andere Aufgabentypen, wie sie zum Beispiel im Fach Betriebswirtschaftslehre benötigt werden, gearbeitet. Der Server verwaltet dabei alle Daten eines Kurses in einem Verzeichnis. Dazu gehören die Übungs- und Klausuraufgaben, die Benutzerdaten, die Auswertungsmechanismen für Aufgabentypen, die speziell in diesem Kurs genutzt werden, und andere Kursmaterialien wie Skripte und Dokumentationen.

Nicht alle Benutzer haben umfangreiche Erfahrung mit Computern. Deshalb hat es sich bei Klausuren, die mit dem MVC geschrieben werden, bewährt, mit den Studierenden eine Testklausur zu schreiben, damit sie sich mit dem System und den Fragetypen auseinander setzten können. So misst eine Klausur dann wirklich die Leistungsfähigkeit der Studierenden und nicht die Kompetenz im Umgang mit dem System.

Im Prüfungsbetrieb treten beim Einsatz des Systems weniger Probleme auf, als bei einer Lösung per Zettelabgabe, da es eine einheitliche Schnittstelle zum System gibt und alle Verhaltensweisen klar definiert sind. Auch ist es nicht mehr erforderlich, zusätzliche Zeit darauf zu verwenden, den Studierenden die Ergebnisse zugänglich zu machen.

Dennoch besteht bei den Studierenden im Umgang mit einem solchen computergestützten System immer ein gewisser Vorbehalt, vor allem weil die Studierenden ihre Arbeitsweisen an einem Aufgabenzettel umstellen müssen. Sie können die Arbeit nur Online erledigen, was einen Internetzugang oder den Aufenthalt in einem Computerraum der Universität voraussetzt. In den betrachteten Kursen handelte es sich um die Kurse „Einführung in die Künstliche Intelligenz“ und „Programmieren in Logik“, in denen vor allem Progammieraufgaben mit Hilfe der Programmiersprache Prolog zu lösen waren. Deshalb fokussiert sich die Problembeschreibung hier etwas auf diesen Kontext.

Außerdem bedeutet die Verwendung eines Browsers zur Aufgabenbearbeitung**,** dass nur der im Browser eingebaute Editor zu Verfügung steht. Diese Browser bieten in der Regel kein Syntaxhighlighting oder anderweitige Unterstützung der Programmierung.

Aufgrund dieser Beschränkungen teilt sich die Benutzergruppe in zwei Fraktionen. Der einen Gruppe ist die Installation eines lokalen Prologsystems zu kompliziert, sie erledigen die Aufgaben vor allem in den von der Universität zur Verfügung gestellten Computerräumen. Eine andere Gruppe installiert sich den Prolog-Interpreter (SWI-Prolog) zuhause und erledigt dort ihre Aufgaben.

Der MVC stellt aus diesem Grund eine Up- und Downloadfunktionalität zur Offline-Bearbeitung der Aufgaben zur Verfügung. In einer kürzlich durchgeführten Umfrage waren die befragten Studierenden mit dem System sehr zufrieden. Es ist sehr stabil, was eine Grundvorrausetzung für die Akzeptanz seitens der Benutzer darstellt.

4. Bewertungsalgorithmen

In allen beschriebenen Versionen verfügt der Virtuelle Campus über automatische Auswertungsroutinen zur Kontrolle und Bewertung der Aufgabenlösungen der Studierenden. Das System bietet dabei zwei verschiedene Modi, um den unterschiedlichen Anforderungen im regulären Tutorium einerseits und in einer studienrelevanten Prüfungssituation andererseits gerecht zu werden.

Im regulären Tutoriumsbetrieb ist es wichtig, die Studierenden auf gemachte Fehler aufmerksam zu machen, um so den Lernprozess qualitativ zu unterstützen. Im regulären Übungsbetrieb wird eine Bewertung daher hinsichtlich eines konstruktiven Feedbacks vorgenommen. In einer Prüfungssituation dagegen soll gelerntes Wissen wiedergegeben und angewendet werden. Im Prüfungsmodus steht daher eine schnelle und präzise Korrektur der Abgabe im Vordergrund.

Eine Implementation zur Generierung von konstruktivem Feedback für Übungssituationen wurde 1999 im Rahmen des Studentenprojektes PLOT (Prolog Learinng Online Tutor) vorgenommen., das sich wie der Name vielleicht schon verrät, nur die Programmiersprache Prolog unterstützt. Genaueres zu den dort verwendeten Algorithmen und Analysemethoden sowie eine Zusammenfassung der mit dem System gemachten Erfahrungen lässt sich bei [Pey01] finden.

Der augenblickliche Entwicklungsschwerpunkt des Systems liegt jedoch in der automatisierten Auswertung und Benotung von Online-Klausuren. Im Folgenden wird daher zunächst ein Überblick über repräsentative Aufgabentypen und Ansätze zu deren Bewertung gegeben. Im Anschluss daran wird das zur Verfügung stehende Aufgabenspektrum mit denen der Lehr-/Lernplattformen Blackboard Learning System und WebCT verglichen. Abschließend wird ein Ausblick auf geplante Weiterentwicklungen und Perspektiven der automatisierten Bewertungsfunktion gegeben.

4.1 Vorhandene Aufgabentypen

Die im MVC implementierten Aufgabentypen erstrecken sich von einfachen Ankreuzaufgaben über Lückentexte bis hin zu Programmieraufgaben. Die automatisierte Auswertung von Ankreuzaufgaben ist dabei ohne großen Aufwand realisierbar. Bei der Bewertung von Lückentextaufgaben treten verschiedene Probleme wie Rechtschreibfehler oder die Verwendung von anfangs nicht bedachten Lösungen auf. Programmieraufgaben sind oft auf verschiedene Arten lösbar, was die Beschränkung auf simples Pattern-Matching[1] bei der Auswertung ausschließt.

Der vorliegende Abschnitt betrachtet die einzelnen Aufgabentypen exemplarisch und beschreibt die augenblicklich verwendeten Bewertungsstrategien.

4.1.1 Ankreuzaufgaben

Der MVC stellt verschiedene Typen von Ankreuzaufgaben zur Verfügung. Die Bewertung der abgegebenen Lösung wird hier anhand einer vorgegebenen Punktemaske vorgenommen. Diese Bewertungsmaske wird dabei direkt in die Aufgabe hineincodiert. Bei der späteren Auswertung vergibt der MVC die Punkte dann automatisch. Die Bewertung erfolgt also ähnlich einer manuellen Korrektur mittels einer Schablone, allerdings mit dem Unterschied, dass der Rechner – und nicht ein menschlicher Korrekteur – die vom Studenten erreichten Punkte zusammenzählt.

Da Ankreuzaufgaben in der deutschen Bildungslandschaft eher ungebräuchlich sind, treten sowohl von Seiten der Lehrenden als auch der Studierenden oft Zweifel beim Einsatz dieses Aufgabentyps auf. Um diesen entgegenzuwirken verfügt der MVC über ein optionales Kommentarfeld, in dem Anmerkungen während des Tests schriftlich festgehalten werden können.

Ankreuzaufgaben stellen damit den einfachsten Fall der automatischen Bewertung dar, ermöglichen jedoch bei einer unklaren Fragestellung den Rückgriff auf ein Kommentarfeld. Die Einführung dieses Kommentarfeldes wurde von den Studenten als erhebliche Verbesserung empfunden.

---

[1] Pattern-Matching stellt einen Mustervergleich, ähnlich dem Auflegen einer Schablone, dar.

4.1.2 Lückentexte

Beim Erstellen von Lückentexten ist es dem Dozenten möglich, mehrere Antwortalternativen pro Lücke anzugeben. Damit kann die Verwendung von Synonymen und alternativen Lösungen von vornherein berücksichtigt werden. Da jedoch auch nicht antizipierte, aber dennoch richtige Antworten auftreten können, ist es zurzeit noch nötig, alle als falsch klassifizierten Antworten von Hand zu überprüfen.

Bei Rechtschreibfehlern kann das System die Ähnlichkeit zur vorgegebenen Lösung erkennen. Diese Möglichkeit ist allerdings optional, da beispielsweise bei Fremdsprachenkursen Rechtschreibfehler explizit als Fehler erkannt und gewertet werden sollen. Für alle übrigen Aufgaben wir eine Bewertung durch erweitertes Pattern-Matching berechnet. Es ist dabei erstaunlich, wie gut Rechtschreibfehler von falschen Antworten unterschieden werden.

Die Lückentextanalyse eliminiert damit alle definitiv richtigen Lösungen und auch alle nicht beantworteten Fragen aus dem zu sichtenden Material und verringert so den Korrekturaufwand erheblich.

4.1.3 Programmieraufgaben

Von den verwendeten Aufgabentypen stellen Programmieraufgaben die höchsten Anforderungen an die Bewertungsalgorithmen des MVC. Programmieraufgaben sind im Gegensatz zu Lückentexten in der Regel nicht durch einfaches Pattern-Matching zu korrigieren, da es in einer Programmiersprache unendlich[2] viele Möglichkeiten gibt, eine Lösung, so sie denn existiert, zu formulieren.

Um einen Lösungsvorschlag dennoch bewerten zu können, wird neben der Form der abgegebenen Lösung die Korrektheit des Algorithmus zur Bewertung herangezogen. Dazu wird das Verhalten des Programms mit dem einer Musterlösung verglichen. Dieser Vergleich geschieht auf einer Menge von Testfällen, die sich aus vom Dozenten festgelegten kritischen Fällen und vom System zufällig generierten Eingaben zusammensetzen. Das Zufallselement soll dabei auch diejenigen Fälle abdecken, die der Lehrende nicht bedacht hat, da eine vollständige Korrektheitsanalyse mit diesem Ansatz nur möglich ist, wenn für alle Klassen von Eingaben ein repräsentativer Fall getestet wird. Um das Verhalten von studentischer Lösung und Musterlösung bezüglicher der zufälligen Eingabe vergleichen zu können, wird dabei der Unterschied zwischen der Ausgabe der Musterlösung und der studentischen Lösung betrachtet. Stimmen beide Ausgaben überein, wird die Lösung bezüglich dieser Eingabe als korrekt bewertet.

Diese Bewertungsmethode ermöglicht die Punktevergabe nach einer kontinuierlichen Bewertungsskala, im Gegensatz zu einer diskreten Bewertung, wie sie bei den zuvor betrachteten Aufgabentypen verwendet wird. Da eine Programmieraufgabe in der Regel nur als Ganzes und nicht als Summe ihrer Teile bewertet werden kann,[3] bietet dieser Ansatz damit die Grundlage zur automatisierten Bewertung von Programmieraufgaben.

Ein besonderer Vorteil dieser Bewertungsstrategie ist die Möglichkeit, dass sie unabhängig von der verwendeten Programmiersprache eingesetzt werden kann, da das zu bewertende Programm als Black Box angesehen wird und nur das Verhältnis von Eingabe zu Ausgabe betrachtet wird.

---

[2] Der Begriff *unendlich* ist hierbei nicht als Sprachfigur zu verstehen. Nach [Rap99] lässt sich ein Programm als logisches Modell einer abstrakten, axiomatisch darstellbaren Beschreibung verstehen. Da eine widerspruchsfreie Menge von Axiomen durch unendlich viele Modelle realisiert werden kann, gibt es hier also tatsächlich unendlich viele Möglichkeiten.

[3] Bei der Bewertung einer Programmieraufgabe können selbstverständlich auch Teilschritte eines konkreten Lösungskonzeptes bewertet werden. Eine solche Bewertung deckt jedoch nicht alle möglichen Lösungen ab und ist erfahrungsgemäß nur dort nötig, wo die Aufgabe nicht vollständig gelöst worden ist.

Dies stellt andererseits aber auch einen nicht zu vernachlässigenden Nachteil dar. Weder die Struktur, noch die Kommentierung des Programms wird bewertet. Auch die Effizienz des betrachteten Algorithmus bleibt bei der augenblicklich verwendeten Strategie unberücksichtigt.

In der aktuellen Version bietet die Bewertungsfunktion für Programmieraufgaben also ausschließlich eine Korrektheitsanalyse des betrachteten Algorithmus. Da dies jedoch die aufmerksamkeitsintensivste Teilaufgabe bei der Bewertung darstellt, ist es bereits mit der jetzigen Version des Systems gelungen, die Bewertung von Programmieraufgaben erheblich zu beschleunigen.

4.2 Vergleich mit marktführenden Lehr-/Lernplattformen

Im Vergleich mit WebCT und Blackboard, zwei marktführenden Lehr-/Lernplattformen, stellen sich schnell die Vorteile des MVC heraus. In der folgenden Übersicht wird das Aufgabenspektrum des MVC kurz mit dem der beiden Systeme verglichen.

Weiterführende Informationen zu beiden Systemen sowie eine Übersicht über weitere Lehr-/Lernplattformen sind bei Baumgartner, Häfele & Maier-Häfele [Bau02] zu finden.

4.2.1 WebCT

WebCT ist eine Lehr/Lernplattform, mit der Kurse verwaltet, Online-Kursmaterial zur Verfügung gestellt werden kann und in der Werkzeuge zur kursinternen Kommunikation bereitgestellt werden. Darüber hinaus beinhaltet WebCT eine so genannte Quiz-Funktion. Im Rahmen dieser Quiz-Funktion bietet WebCT vier verschiedene Aufgabentypen:

- Multiple Choice
- Matching
- Fragen mit einer möglichen Antwort
- Rechenaufgaben

Bei Multiple Choice Aufgaben handelt es sich dabei um Ankreuzaufgaben, wie sie auch im MVC implementiert sind. Bei Matching Aufgaben handelt es sich um einfache Zuordnungsaufgaben. Die automatische Bewertung dieser beiden Aufgabentypen ist aufgrund der binären Struktur ihrer Antworten[4] sehr einfach realisierbar.

Bei Fragen mit einer möglichen Antwort[5] handelt es sich bezüglich der Bewertung um denselben Typ, wie bei Lückentexten im MVC. Die Auswertung ist daher ähnlich gehalten. Wie der MVC kann auch WebCT mit Synonymen umgehen. Rechtschreibfehler kann WebCT jedoch nur dann ignorieren, wenn diese zuvor vom Lehrenden explizit angegeben werden.

Rechenaufgaben verlangen die Eingabe einer Formel unter Verwendung von Variablen. Diese Variablen initialisiert das System zufällig und vermeidet es so, jedem Studierenden dieselbe Aufgabe zu stellen. Bei einem Aufruf lautet die Frage dann „Wie viel ist 2 + 3“ und beim nächsten „Wie viel ist 5+1“. Die Antwort steht aufgrund der Formel bereits zu dem Zeitpunkt fest, an dem das System die Aufgabe stellt. Die Bewertungslogik dieses Aufgabentyps entspricht daher einer Lückentextaufgabe ohne Berücksichtigung von Rechtschreibfehlern.

WebCT umfasst also ein ähnliches Aufgabenspektrum wie der MVC, mit der Ausnahme, das WebCT nicht über eine Auswertungsfunktion für Programmieraufgaben verfügt.

4.2.2 Blackborad Learning System

---

[4] richtig oder falsch

[5] Die Antwort besteht dabei aus einer genau definierten Zeichenkette.

Blackboards Learning System verfolgt eine ähnliche Zielsetzung wie WebCT. Eine Besonderheit ist das Building Block Programm, das die benutzerseitige Erweiterung des Learning Systems erlaubt. Auch dieses System verfügt über eine Testkomponente mit automatischer Aufgabenbewertung.

Das Learning System bietet dabei eine ganze Reihe von unterschiedlichen automatisch auswertbaren Aufgabentypen, die jedoch allesamt keine freie Eingabe erlauben. Freitextaufgaben sind möglich, müssen aber zu 100% von Hand ausgewertet werden.

Hinsichtlich der Bewertungslogik sind diese Aufgabentypen daher mit einem einfachen Typ wie dem der Ankreuzaufgaben des MVC vergleichbar.

4.2.3 Zusammenfassung des Vergleichs

Hinsichtlich der Komplexität der zur Verfügung stehenden Aufgabentypen hat der MVC vor allem durch die Möglichkeit, Programmieraufgaben zu stellen, einen klaren Vorsprung. Auch der Lückentextalgorithmus stellt hier einen Pluspunkt dar, der aber in einem der beiden Referenzsysteme auch implementiert ist.

Von allen betrachteten Aufgabentypen sind es allerdings Programmieraufgaben oder Aufgaben mit großer Texteingabe, welche jedoch noch nicht adäquat automatisch korrigiert werden können, die am ehesten in das Selbstverständnis der deutschen Hochschullandschaft passen. Mit ihnen ist es möglich, Transferleistungen statt bloßem Faktenwissen abzufragen.

Der MVC stellt also als einziges der drei Systeme eine Möglichkeit dar, in einer Prüfung auch Material mit einem Transferanteil zu stellen und automatisch zu bewerten.

4.3. Ausblick auf geplante Bewertungsstrategien

In den Bereichen Lückentextaufgaben und Programmieraufgaben gibt es, wie bereits beschrieben, bei falschen und als falsch klassifizierten Antworten einen nicht unerheblichen Bedarf an menschlicher Nachkorrektur. Diese Nachkorrektur auf ein Mindestmaß zu beschränken, ist eines der Ziele der weiteren Entwicklung des Virtuellen Campus.

Im Folgenden werden die Entwicklungsperspektiven der beiden Aufgabentypen Lückentexte und Programmieraufgaben dargestellt. Darüber hinaus wird die Bewertungsstrategie für den geplanten Aufgabentyp Rechnersimulation skizziert.

4.3.1 Lückentexte

Bei diesem Aufgabentyp ist eine dynamisch erweiterbare Fallbasis von richtigen und falschen Lösungen geplant. Jedes Mal, wenn der Dozent während der Korrektur eine abgegebene Lösung klassifiziert hat, wird diese Klassifikation in der Fallbasis gespeichert. Bei wiederholtem Auftreten derselben Lösung muss der Dozent dann nicht mehr gefragt werden, da die Bewertung bereits im System gespeichert ist.

Das System wird also lernen, wie eine Aufgabe zu bewerten ist, und nicht wie bisher dem Korrekteur mehrmals dieselbe Frage stellen.

4.3.2 Programmieraufgaben

In der bisherigen Implementation bewertet der MVC eine Programmieraufgabe nicht hinsichtlich ihrer Effizienz. Da die meisten Programmiersprachen Analysewerkzeuge für Laufzeitanalysen zur Verfügung stellen, ist eine solche Effizienzanalyse jedoch relativ leicht zu realisieren.

Ein zweiter Schritt ist die automatische Generierung von Testfällen mittels der Abstract State Machine Language (Asml), wie in [Gri01] beschrieben. Dies

erfordert allerdings eine Spezifikation der Aufgabe in Asml. Da dies nicht von jedem Dozenten zu verlangen ist, wird zunächst die bereits beschriebene Generierung zufälliger Testfälle mit einer dynamischen Fallbasis kombiniert. Dazu werden die Ausgaben von Musterlösung und abgegebenen Programmen verglichen. Immer dann, wenn ein zufällig generierter Testfall unterschiedliche Ausgaben erzeugt, obwohl alle in der Fallbasis gespeicherten Testfälle in beiden Programmen identische Ausgaben erzeugen, wird er in die Fallbasis aufgenommen. Nach einem Bewertungsdurchlauf über alle abgegebenen Aufgaben werden alle Aufgaben mit der erweiterten Fallbasis ausgewertet. Diese Methode gewährleistet eine Gleichbehandlung aller abgegebenen Lösung und reduziert das Risiko von nicht angegebenen Testfällen.

4.3.3 Rechenaufgaben

Die bereits für Programmieraufgaben eingesetzte Black-Box-Betrachtungsweise lässt sich ohne Änderungen des Bewertungsalgorithmus für Rechnersimulationen adaptieren. Die Erweiterung des MVC durch Simulationsaufgaben ist erklärtes Ziel der Entwicklung, da in Fachbereichen wie Wirtschafts- und Gesellschaftswissenschaften Transferaufgaben oft in Aufsatzform gestellt werden. Derartige Freitextaufgaben zumindest teilweise durch automatisch auswertbare Simulationen zu ersetzen, würde den Korrekturbedarf erheblich reduzieren und auch die oft unverhältnismäßig langen Korrekturzeiten verkürzen.

5. Zusammenfassung und Ausblick

5.1 Ausblick

Die in den verschiedenen Versionen gemachten Erfahrungen erlauben einen technisch sicheren und unterbrechungsfreien Betrieb des Systems. Darauf aufbauend ist zunächst eine Überarbeitung des Benutzerinterfaces unter Berücksichtigung psychologischer Gesichtspunkte mit paralleler Evaluation geplant. Auch ist die Einbindung in eine Lehr-/Lernplattform angedacht.

Auf der technischen Seite wird an einer Modularisierung gearbeitet, die es ermöglicht, beliebige Aufgabentypen zu integrieren. Außerdem sollen weitere Schnittstellen zur Authentifizierung gegen externe Quellen eingebaut werden (ldap), um die Verwaltung der Benutzer zu erleichtern und die Zugangsdaten zu vereinfachen.

Durch die Erweiterung des Aufgabenspektrums um Simulationsaufgaben und die Schaffung weiterer Schnittstellen als Alternative zum Browser wie beispielsweise WebDAV sollen die Grundlagen gelegt werden, um das System in einem breiteren Rahmen einsetzen zu können.

5.2 Fazit

Der MVC ist eine virtuelle Übungsumgebung, die aus einem langjährigen Erfahrungsschatz schöpfen kann und mit der Zeit an den Bedürfnissen der Nutzer und den Erfahrungen aus dem praktischen Einsatz des Systems gewachsen ist. Sie hat sich als stabil, robust und flexibel bewährt und ist auf dem Übungs- und Klausurenteil vielen kommerziellen Systemen überlegen. Außerdem besitzt sie eine automatische Auswertung von Programmieraufgaben und in absehbarer Zukunft Simulationen, die von kommerziellen Lernplattformen nicht unterstützt werden.

[Bau02] Baumgartner, P., Häfele, H. & Maier-Häfele, K. (2002). Auswahl von Lernplattformen Marktübersicht – Funktionen – Fachbegriffe. Innsbruck.

[Gos96] Gosling, J. & Mc Gilton. H. (1996) A White Paper. The Java Language Environment, Santa Clara.

[Gri01] Grieskamp, W., Gurevich, Y., Schulte, W. & M. (2001). Testing with Abstract State Machines. In: Moreno-Díaz, R. & Quesada-Arencibia, A. (Hrsg.). Formal Methods and Tools for Computer Science (Proceedings of Eurocast 2001*)*, Las Palmas de Gran Canaria, 257-261.

[Pey98] Peylo, C. & Teiken, W. (1998). Ein internetbasiertes tutorielles System als Java-Applet. Erfahrungen mit einer plattformunabhängigen Sprache unter verschiedenen Plattformen, Osnabrück.

[Pey99] Peylo, C., Teiken, W, Rollinger, C. & Gust, H. (1999) Der VC-Prolog-Tutor, eine Internetbasierte Lernumgebung. In: Künstliche Intelligenz, Bremen, 32-36.

[Pey01] Peylo, C., Rollinger, C. & Thelen, T. (2001) Intelligenter Support für Übungen und den Übungsbetrieb, als Unterkapitel von Hauenschild, C., Neijdl, W., Rollinger, C., Wagner, E. & Womser-Hacker, C., Der Entwicklungsverbund: Sprache – Wissen – Information. In Wagner, E. & Kindt, M : Virtueller Campus: Strategien – Szenarien – Studium. Münster: Waxmann, S462-506. 2001

[Rap99] Rapaport, William J. (1999) Implementation Is Semantic Interpretation. In: The Monist 82, 109-130.