Automatische Bewertung von Übungsaufgaben in VIPS

Helmar Gust, Nadine Werner
IKW Universität Osnabrück, virtUOS Universität Osnabrück
helmar.gust@uos.de, nadine.werner@uos.de

Abstract

Vips ist ein virtuelles Prüfungssystem und dient zur Verwaltung, Durchführung und Auswertung von Online-Übungen und Online-Klausuren. Der Schwerpunkt der Vips-Entwicklung liegt auf der automatischen Auswertung von Aufgaben, um demjenigen, der die Aufgaben stellt und beurteilen muss, möglichst viel Arbeit abzunehmen. Darüber hinaus ermöglicht Vips Übungen und Klausuren für die Studierenden in einem Selbsttest-Modus anzubieten. Das System bietet verschiedene Möglichkeiten zur Entwicklung, Pflege und Auswertung virtueller Prüfungen sowie die Verwaltung des gesamten Übungsbetriebs eines Kurses.

1 Einleitung

Die Korrektur und Bewertung von Übungsaufgaben im Rahmen von Lehrveranstaltungen erfordert von den Dozenten einen nicht unerheblichen Aufwand an Zeit. Daher haben sich in vielen Bereichen stark schematisierte Aufgabentypen durchgesetzt. Typische Beispiele dafür sind Multible- und Single-Choice Aufgaben sowie Zuordnungsaufgaben. Auch für solche Aufgaben ist die händische Korrektur zeitaufwändig und lästig. Allerdings lässt sich die Auswertung dieser Aufgaben sehr leicht automatisieren. Immer dann, wenn freie Texteingaben erwartet werden (und dies gilt bereits bei Lückentextaufgaben), ist für die Beurteilung aber ein gewisses Verständnis der Aufgabe notwendig, zumindest dann, wenn von der vorgegebenen korrekten Lösung abgewichen wird. Bei weitem komplexer ist die Situation natürlich bei Freitextaufgaben. Aber auch bei diesen Aufgaben lässt sich die Korrektur und Bewertung von automatischen Systemen zumindest unterstützen. So hilft bereits das herausfiltern klarer Standardfälle, einen erheblichen Teil des Zeitaufwandes einzusparen, etwa wenn auf der einen Seite Musterlösungen erkannt werden können und auf der anderen Seite klare Fälle der Nichtbeantwortung herausgefiltert werden können. Programmieraufgaben nehmen eine Sonderstellung ein. Zum einen handelt es sich um Aufgaben, die sich weitgehend wie Freitextaufgaben verhalten. Zum andern gibt es aber starke formale Restriktionen an die eingegebenen Texte: Sie müssen als Programm-Code fehlerfrei kompilierbar sein und sie müssen das geforderte Programm-Verhalten produzieren. Die Korrektur und Bewertung von Programmieraufgaben ist normalerweise erheblich aufwändiger als bei Freitextaufgaben, insbesondere dann, wenn sie Fehler enthalten. Eine Unterstützung bei diesen Aufgaben ist also sehr hilfreich aber wegen der Komplexität solcher Aufgaben auch entsprechend schwierig.

Vips wurde entwickelt, um einfach online Übungen und Klausuren durchführen zu können und den Dozenten bei einer Reihe organisatorischer Aufgaben zu entlasten. Ein Schwerpunkt der Vips-Entwicklung liegt dabei auf der automatischen Auswertung von Aufgaben, um demjenigen, der die Aufgaben stellt und beurteilen muss, möglichst viel Arbeit abzunehmen. Darüber hinaus ermöglicht Vips Übungen und Klausuren für die Studierenden in einem Selbsttest-Modus anzubieten. Neben der Erstellung und Verwaltung von Übungsblättern und Klausuren umfasst Vips eine Arbeitsgruppenverwaltung, Punkte- und Notenübersichten für einzelne Teilnehmer und Arbeitsgruppen, sowie eine flexible Notenberechnung. Vips stellt eine Vielzahl unterschiedlicher Aufgabentypen zur Verfügung: Single Choice, Multiple Choice, Ja/Nein-Fragen, Zuordnungen, Lückentext, Freitext und Programmieraufgaben (z.B. Prolog, Lisp, Haskell oder Octave).

In diesem Papier soll der Schwerpunkt auf Programmieraufgaben und deren automatischer Auswertung liegen. Single Choice, Multiple Choice, Ja/Nein-Fragen, Zuordnung und Lückentext-Aufgaben können einfach automatisch bewertet werden. Vips führt für diese Aufgaben standardmäßig eine komplett automatische Bewertung durch. Allerdings kann das Ergebnis auch bei diesen Aufgaben vom Dozenten nachträglich überschrieben werden. Für Freitext und Programmieraufgaben ist eine vollständig automatische Auswertung i.a. nicht möglich. Trotzdem kann eine Korrektur erheblich durch automatisierte Verfahren unterstützt werden. Vips stellt für Programmieraufgaben eine Runtime-Umgebung mit einer einfachen GUI zur Verfügung. Dies ermöglicht den Kursteilnehmern die Aufgaben ohne lokale Installationen der Programmiersprachen zu lösen. Allerdings stehen in-

teraktive Debug-Möglichkeiten in einer solchen Umgebung nicht zur Verfügung. Daher kann auf eine lokale Installation der Programmiersprachen insbesondere bei komplexen Aufgaben nicht immer verzichtet werden. Um lokale Installationen zum Aufgabenlösen benutzen zu können, gibt es Down- und Upload-Möglichkeiten für die Aufgaben. Die Runtime-Umgebung steht auch für die Aufgabenkorrektur zur Verfügung. Zusammen mit einem vorgegebenen Default-Aufruf der Hauptfunktion eines Programms ermöglicht dies sowohl dem Kursteilnehmer als auch dem Korrekteur sich mit einem Mausklick einen ersten Überblick über die Lauffähigkeit der Lösung zu verschaffen.

Zur Unterstützung dieser Programmieraufgaben gibt es ein eigenes Servermodul VEA (Vips Evaluation Assistent), das auf einem vom StudIP/Vips-System getrennten Rechner läuft. Die Kommunikation zwischen StudIP/Vips und der Programmierumgebung geschieht über HTTP. Mit einem Bowser kann diese Schnittstelle auch direkt benutzt werden[1], um VEA zu testen und zu konfigurieren. Vips entstand aus Ansätzen, die im Vorfeld und im Rahmen eines Projektes am Institut für Kognitionswissenschaft entwickelt wurden [1]. Vips ist ein Plugin für Stud.IP[2] und daher zur Zeit nur in Kombination mit Stud.IP verfügbar. Eine Überblick über die Architektur gibt Bild 1.

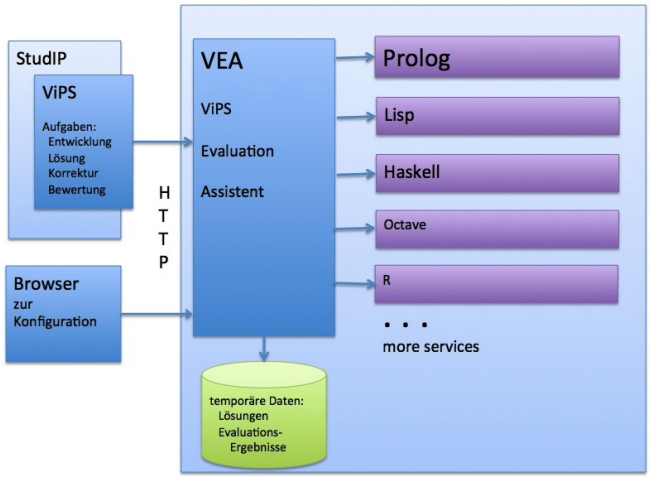

Abbildung 1: ViPS / VEA -Architektur

2 Das Vips Modul in StudIP

An der Universität Osnabrück ist das Übungs- und Prüfungssystem Vips seit Sommersemester 2004 als modulare Erweiterung der Lehr- und Lernplattform StudIP in einer ersten Version verfügbar. Es kann einer Lehrveranstaltung als Plugin hinzugefügt werden. Das System bietet verschiedene Möglichkeiten zur Entwicklung, Pflege und Auswertung virtueller Prüfungen sowie die Verwaltung des gesamten Übungsbetriebs eines Kurses. Vips stellt Übungsblätter und Kausuren zur Verfügung. Beide unterscheiden sich nicht in der Struktur der Aufgaben sondern nur im Modus des Ablaufes und der Zuordnung zu einzelnen Teilnehmern (bei Klausuren) bzw. zu Arbeitsgruppen (bei Übungsblättern). Zudem gibt es für Übungsblätter einen Selbsttest-Modus, bei dem dem Kursteilnehmer nach dem Absenden einer Lösung sofort das Ergebnis der automatischen Auswertung angezeigt wird. Der Klausur-Modus erlaubt eine online Überprüfung der Zugriffsparameter sowie ein Überwachung des Arbeitsfortschritts der Teilnehmer.

Vips umfasst eine Arbeitsgruppenverwaltung, Punkte- und Notenübersichten für einzelne Teilnehmer und Arbeitsgruppen, sowie eine flexible Notenberechnung. Arbeitsblätter und Klausuren können (zur Zeit) als Text-Dateien importiert und exportiert werden. Darüber hinaus gibt es eine Druckansicht.

Im Folgenden wird insbesondere auf Programmieraufgaben und die automatische Unterstützung deren Auswertung eingegangen.

3 Programmieraufgaben in Vips

Vips stellt für Programmieraufgaben eine Runtime-Umgebung mit einer einfachen GUI zur Verfügung. Die GUI ist über einen Reiter einer Lehrveranstaltung erreichbar. Sie ermöglicht für Programmieraufgaben den Programm-Code einzugeben und zu editieren sowie die Programme auszuführen. So können Kursteilnehmern die Aufgaben lösen, ohne die Programmiersprachen lokal auf ihrem Rechner installieren zu müssen. Allerdings stehen interaktive Debug-Möglichkeiten in einer solchen Umgebung nicht zur Verfügung. Um für komplexe Aufgabenstellungen auch lokale Installationen zum Aufgabenlösen benutzen zu können, gibt es Down- und Upload-Möglichkeiten für die Aufgaben.

Die Runtime-Umgebung steht auch für die Aufgabenkorrektur zur Verfügung. Zusammen mit einem vorgegebenen Default-Aufruf der Hauptfunktion ermöglicht dies sowohl dem Kursteilnehmer als auch dem Korrektor sich mit einem Mausklick einen ersten überblick über die Lauffähigkeit der Lösung zu verschaffen.

3.1 Anlegen eines Übungsblattes mit Programmieraufgaben

Um ein Übungsblatt mit Programmieraufgaben in Vips anzulegen, gibt es unter dem Reiter 'Vips' einer Lehrveranstaltung mehrere Aktionen. Der Dozent kann ein neues Übungsblatt anlegen, ein Übungsblatt auch aus einer Textdatei importieren oder die Daten aus bereits vorhandenen Übungsblättern importieren. Ist das Übungsblatt in der Oberfläche mit Titel und Beschreibung sowie Start und Endzeitpunkt versehen, können Übungsaufgaben neu hinzugefügt werden. Weiterhin hat der Dozent die Möglichkeit, Aufgaben aus bereits im StudIP vorhandenen Übungsblättern dieses oder anderer Kurse zu übernehmen. Vips stellt für eine Programmier-Übungsaufgabe, die einem Übungsblatt

[1] https://mvc.ikw.uni-osnabrueck.de/vips/vips.php
[2] www.studip.de

hinzugefügt wird, neben der Aufgabenstellung eine Reihe von Informations-Feldern bereit:

- Ein Lösungesfeld in das der Bearbeiter den zu entwickelnden Code eingeben kann. Dieses Felder kann vorbelegt werden, so dass dem Kursteilnehmer bereits ein Lösungsgerüst präsentiert werden kann, das z.B. vom Bearbeiter ergänzt bzw. geändert werden muss.
- Ein Feld für Musterlösungen. Dieses Feld kann mehrere Musterlösungen enthalten. zudem enthält diese Feld noch den Evaluations-Code. Dieser wird in Zukunft in einem eigenen Textfeld verwaltet.
- Ein Feld mit Aufruf der Hauptfunktion. In Prolog ist dies eine Folge von Subgoals durch Kommas getrennt. In anderen Programmiersprachen kann dies ein Funktionsaufruf oder ein arithmetischer Ausdruck sein.

Da das Erscheinungsbild einer Aufgabenspezifikation für die Kursteilnehmer nicht immer einfach vorhergesehen werden kann, gibt es eine einfache Möglichkeit zwischen der Darstellung für die Aufgabenentwicklung und der Darstellung für den Kursteilnehmer umzuschalten.

3.2 Bearbeiten eines Übungsblattes mit Programmieraufgaben

Aus Kursteinehmer-Sicht bietet die GUI folgende Möglichkeiten:

- Der Kursteilnehmer kann seine Lösung in ein Textfeld eintragen. Dieser Bereich ist eventuell bereits mit einem Lösungsgerüst vorbelegt.
- Basierend auf der aktuellen Lösung kann eine Testanfrage gestellt werden. Das Ergebnis des Programmlaufs wird angezeigt.
- Der Inhalt des Lösungsbereichs kann in eine lokale Datei heruntergeladen werden, so dass eine lokale Installation der Programmiersprache zur Programmentwicklung benutzt werden kann.
- Anschießend kann die fertige Lösung wieder in das Bearbeitungsfeld hochgeladen werden.
- Über den Button 'abschicken', wird die Übungsaufgabe abgegeben. Übungsaufgaben können im Rahmen der Bearbeitungszeit mehrfach abgegeben werden. Die als letztes abgegeben Lösung zählt.

Kursteilnehmer können in Arbeitsgruppen organisiert sein. Dann gilt bei einem Aufgabenblatt eine abgegebene Lösung für die gesamte Arbeitsgruppe. Bei einer Klausur gilt sie immer nur für einzelnen Teilnehmer. Im Selbsttestmodus wird nach der Abgabe die Auswertung und die (erste) Musterlösung angezeigt.

3.3 Auswertung eines Übungsblattes mit Programmieraufgabe

Im Korrektur- und Bewertungsmodes von Vips stellt die Oberfläche folgende Funktionen zur Verfügung

- Im Programm-Code kann korrigiert und kommentiert werden.
- In einem Textfeld können generelle Kommentare hinzugefügt werden.
- Punkte können für die Lösung vergeben werden. Dabei werden zunächst die automatisch vergebenen Punkte angezeigt.
- Der *query*-Knopf kompiliert den Lösungsfeld-Inhalt und ruft die Hauptfunktion auf. Die Ergebnistexte der Kompilation und der Programmausführung werden ausgegeben. Für nichtdeterministische Sprachen wie Prolog gibt es den Knopf 'query next', der bei jeder Betätigung eine weitere Lösung ausgibt.
- Der *eval*-Knopf stößt die automatisceh Bewertung an (siehe Abbildung 2). Der Bewertungprozess kann über Tags in der Musterkösung gesteuert werden. Als Ergebnis werden u.a. zwei Zahlen zwischen 0 und 1 angezeigt: Ein Score, der die Güte der Lösung repräsentiert und ein Validitätswert, der die Verlässlichkeit der Bewertung signalisiert. Bei einem Validitätswert von 1 kann die Bewertung als sicher angesehen werden. Als Bewertungkriterien können die IO-Relation (also das Verhalten) des Programms sowie textuelle Vergleiche des Lösungscodes mit den Musterlösungen auf verschiedenen Normalisierungsebenen herangezogen werden.

Die abgegebenen Übungsblätter werden dem Dozenten in einer Übersicht präsentiert. Von hier aus gelangt er zu den einzelnen Übungsaufgaben.

4 Der Vips Evaluation Assistent VEA

Zur Unterstützung der Auswertung von Programmieraufgaben gibt es ein eigenes Server-Modul VEA (Vips Evaluation Assistent), das auf einem vom StudIP-System getrennten Rechner läuft. Sinnvoll ist hier eine dedizierter Rechner, der nur für diese Aufgabe zur Verfügung steht, da die Ausführung von fremden Programmcode immer auch mit Sicherheitsrisiken verbunden ist. VEA stellt Runtime-Umgebungen für Programmieraufgaben in ausgewählten Programmiersprachen zur Verfügung. Diese Umgebungen können zur Entwicklung der Lösungen und zur Bewertung dieser Lösungen benutzt werden. Die Kommunikation zwischen StudIP/Vips und der Programmierumgebung geschieht über HTTP, so dass diese Schnittstelle mit einem Bowser auch direkt benutzt kann. Dieses Interface gestattet neben der Konfiguration auch den Test aller Funktionen, sowie die Erweiterung des Systems um weitere Programmiersprachen.

4.1 Kommunikation mit Vips

Zur Zeit gibt es zwei Modi für die Kommunikation zwischen Vips und VEA:

1. Alle Datenfelder werden als eigene Post-Felder übertragen (wird zur Zeit von Vips benutzt).
2. Die Aufgabenspezifikation wird als XML-Strutur übertragen und nur die abgegebene Lösung wird

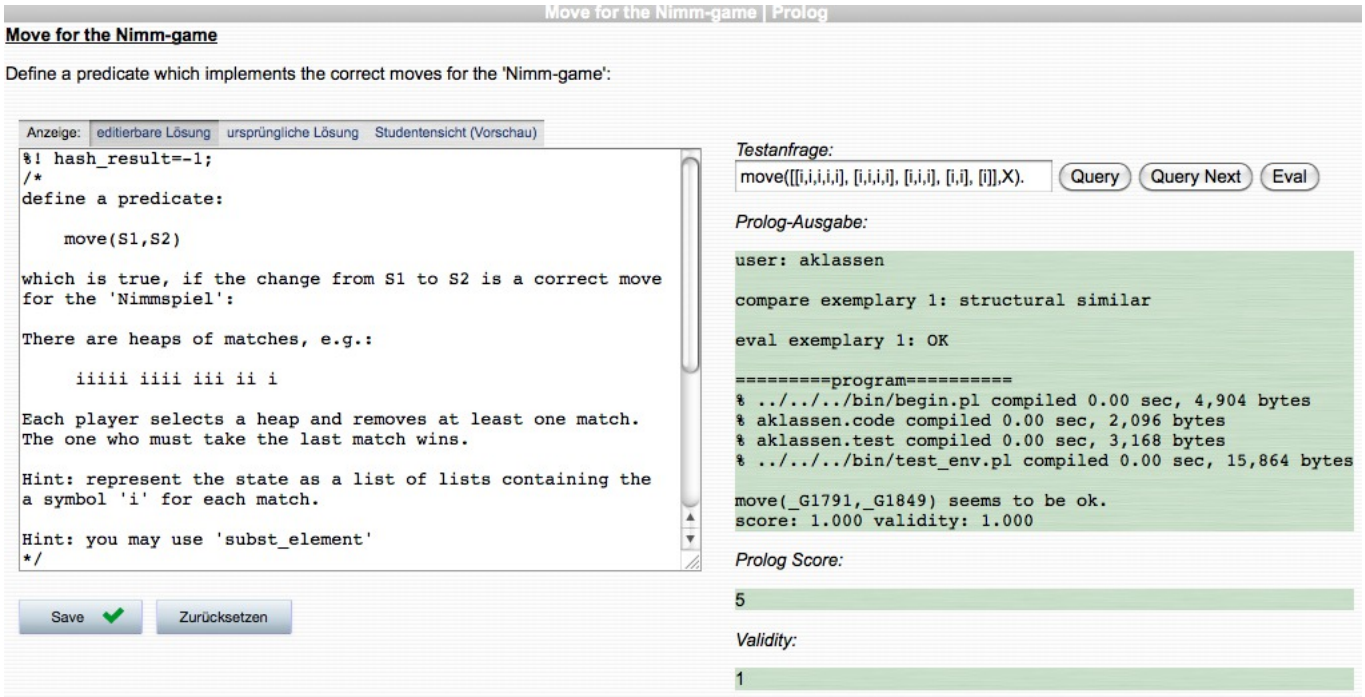

Abbildung 2: Auswertungsmodi einer Programmier-Übungsaufgabe

als eigenes Feld übertragen. Die XML-Struktur entspricht dem Aufgaben-Austauschformat, das im Rahmen von eCULT[3] entwickelt wird.

Die wichtigsten Felder, die VEA erwartet, sind in Tabelle 1 aufgeführt: Im Fall, dass eine XML-Struktur übertrage wird, werden diese Felder teilweise aus der XML-Struktur gefüllt.

Tabelle 1: Liste der wichtigsten Felder

Feldname	Beschreibung	Werte
adm	Administrationsebenen	0|1|2|3|4|5|6
typ	Auswertungsmodus	query | eval
mgc	Password	
srv	Service	Prolog | Lisp ....
qry	Query / Call	
exp	Musterlösungen	
ini	Vorbelegung des Lösungsfeldes	
exc	ausführbare Lösung	
xml	XML-Spezifikation einer Aufgabe	

4.2 Implementierte Funktionen

Die Schnittstelle erlaubt neben der Konfiguration die Ausführung eine Reihe von Funktionen: *list*, *hash*, *query* und *eval*.

[3]http://www.ecult-niedersachsen.de/

list Es wir eine Liste der verfügbaren Services zurückgegeben. Zur Zeit ist nur der Service Prolog und Lisp komplett realisiert. Haskel, Oktave und R sind nur rudimentär vorhanden.

query Der Auswertungsmodus *query* kompiliert den Lösungsfeld-Inhalt und ruft die Hauptfunktion auf. Der Ergebnistext (Compiler-Protokoll und das Ergebnis der Ausführung, falls das Programm fehlerfrei kompiliert werden konnte) wird an Vips übertragen und ausgegeben. Im Falle einer Fehlermeldung oder einer Warnung wird zusätzlich der mit Zeilennummern versehene Code ausgegeben, um die Fehlersuche zu erleichtern. Damit wird eine rudimentäre Entwicklungsumgebung realisiert, in der Programmentwürfe und Lösungsansätze getestet werden können.

eval Der Auswertungsmodus *eval* testet die abgegebene Lösung und versucht, eine automatische Bewertung durchzuführen. In diesem Modus liefert VEA im Wesentlichen zwei Zahlen s (score) und v (validity) zwischen 0 und 1 zurück. $v = 1$ bedeutet, dass eine valide Bewertung vorgenommen werden konnte. $s = 0$ bedeutet in diesem Fall, dass die Lösung vollkommen falsch ist und $s = 1$ entsprechend, dass sie vollkommen richtig ist (z.B. wenn sie mit einer Musterlösung übereinstimmt). Eine Überprüfung durch einen Korrekteur sollte in diesem Fall nicht notwendig sein. Werte von v kleiner 1 deuten darauf hin, dass keine sichere Bewertung vorgenommen werden konnte, der Score s also nur ein Anhaltspunkt für den Korrekteur sein sollte. Es stehen mehrere Testverfahren zur Verfügung. VEA iteriert die beiden folgenden

Verfahren jeweils über alle Musterlösungen m.

- Vergleich der Ergebnisse der Lösung und der Musterlösung bei verschiedenen Eingaben. Der Auswertungs-Modus 'eval' kompiliert sowohl den Lösungsfeld-Inhalt als auch die Musterlösung sowie ein Stückchen Test-Code, dass die Ergebnisse des Lösungsvorschlags und den Musterlösungen vergleicht. Die Werte für s und v für diesen Testteil werden folgendermaßen berechnet:

$$s^m_{comp} = \frac{1}{1 + \#compiler\_fehler} \quad (1)$$

$$s^m_{eval} = \frac{0.5 * \#pos}{\#pos + \#neg)} + 0.5 * s_{comp} \quad (2)$$

$$v^m_{eval} = s^m_{eval} \quad (3)$$

  $\#pos$ ist dabei die Anzahl der Testfälle, für die korrekte Ergebnisse geliefert wurden, und $\#neg$ entsprechend die Fälle, für die falsche Ergebnisse geliefert wurden. Die letzte Gleichung ist in sofern plausibel, als dass ein niedriger Score-Wert nicht bedeuten muss, dass das Programm komplett falsch ist: Ein kleiner Fehler kann etwa eine Reihe von Compiler-Fehlermeldungen evozierten. In diesem Fall sollte also auch ein entsprechend niedriger Validitätswert angesetzt werden. Auf der anderen Seite bedeutet eine hoher Score-Wert, dass das Programmverhalten korrekt ist, was ein relativ sicherer Hinweis darauf sein sollte, dass auch das Programm korrekt ist.

- Textueller Vergleich des Lösungsvorschlags mit der Musterlösung und der Vorbelegung des Lösungsfeldes. Auf der Basis eines Ähnlichkeitsmaßes[4] sim werden ein Score s_{sim} und ein Validitätswert v_{sim} nach folgender Formel berechnet

$$s' = sim(EXC, EXP) * \frac{1 - sim(INI, EXC)}{1 - sim(INI, EXP_m)} \quad (4)$$

$$s^m_{sim} = s'^2 * s^m_{max} \quad (5)$$

$$v^m_{sim} = max(1 - s', s^m_{sim})^2 \quad (6)$$

  EXC und INI referieren auf die entsprechenden Feldinhalte (Lösungsvorschlag und Vorbelegung) und EXP_m auf die m-te Musterlösung. s' liefert also 0, falls die Vorbelegung nicht geändert wurde, und 1, falls eine der Musterlösungen eingegeben wurde. Formel (5) realisiert eine pessimistische Sichtweise für den Score-Wert und Formel (6) entsprechend für den Validitätswert. s^m_{max} ist der vorgegebene maximal Score für die m-te Musterlösung[5]. Kritisch ist natürlich die Wahl der Funktion sim. Das Mögliche Spektrum reicht von spezifischen Funktionen für die einzelnen Programmiersprachen, die die syntaktische Struktur berücksichtigen können bis zu robusten Textvergleichen unabhängig von der konkreten Programmiersprache. In der gegenwärtigen VEA-Version wird die zweite Möglichkeit benutzt:

$$sim(t_1, t_2) = \frac{2 * similar\_text(t_1, t_2)}{|t_1| + |t_2|} \quad (7)$$

  Dabei ist $similar\_text$ eine PHP-Funktion, die die die übereinstimmenden Zeichen zählt.

- Textueller Vergleich des Lösungsvorschlags und der Musterlösungen auf der Basis verschieden starker Normalisierungsstufen. Die wesentlichen Stufen sind folgende:
  - In der Standard-Stufe (wird grundsätzlich angewendet) werden nur Kommentare am Anfang und Ende der Lösung sowie doppelte Zeilenumbrüche und Blanks am Zeilenende entfernt.
  - In der schwachen Stufe werden zusätzlich Blanks vor und nach Operatoren und Trennzeichen sowie mehrfache Blanks und Kommentare entfernt.
  - In der mittleren Stufe werden zusätzlich alle Blanks und Zeilenümbrüche entfernt.
  - In der starken Stufe werden zusätzlich alle kleingeschriebenen Symbole auf 'a' und alle großgeschriebenen Symbole auf 'V' reduziert, so dass nur die Programmstruktur selbst erhalten bleibt. [6]

  Für die ersten drei Stufen wird auf Identität der normalisierten Texte getestet. Für die stärkste Stufe wird ein toleranter Textmatch, der maximale gemeinsame Teilstrings berücksichtigt, verwendet. Diese Vergleiche werden u.a. dazu benutzt, um ein textuelles Ergebnis ausgeben zu können: 'literally same', 'same with similar layout', 'same up to layout' und 'stucturally similar', je nachdem, auf welcher Stufe die Gleichheit erkannt wurde. Für diese Fälle können auch feste vorgegebene Werte für den Score s und den Validitätswert v vergeben werden. In diesem Fall überschreiben diese die Werte aus dem vorigen Verfahren.

Hieraus berechnen sich die Werte (v_{sim}, s_{sim}) als $max_m((v^m_{sim}, s^m_{sim}))$ und (v_{eval}, s_{eval}) als $max_m((v^m_{eval}, s^m_{eval}))$ bezüglich der lexikalischen Ordnung. Für jede Aufgabe kann festgelegt werden, ob nur der Lösungsmengenvergleich, nur der Textvergleich oder eine Kombination von beiden (default) in die endgültigen Werte von s und v eingehen, die zur Berechnung eines Bewertungsvorschlages benutzt werden. Diese relativ groben heuristischen Verfahren zur Berechnung der Werte s und v haben sich insbesondere für Prolog als durchaus brauchbar erwiesen.

[4] Für ein Ähnlichkeitsmaß sim muss gelten $0 \leq sim(x, y) \leq sim(x, x) = 1$

[5] Musterlösungen können unterschiedliche Güte haben. Für die erste Musterlösung gilt immer $s^m_{max} = 1$

[6] Insbesondere die stärkste Normalisierungsstufe sollte abhängig von der Programmiersprache sein. Es ist daran gedacht, für alle Stufen in der Konfiguration parametrisierbare Pattern zu erlauben.

hash Es werden vier MD5-Hash-Werte zurückgegeben, die den vier Code-Normalisierungen entsprechen. Diese Werte können in Vips benutzt werden, um Ähnlichkeiten zwischen den Lösungen unterschiedlicher Teilnehmer zu erkennen. So kann z.B. festgestellt werden, ob eine Lösung schon als Lösung eines anderen Teilnehmers korrigiert wurde.

4.3 Konfiguration

Zur Konfiguration eines Services sind zwei Dateien notwendig:

config enthält eine Reihe von Attribut-Wert-Paare zur Konfiguration des Systems. Hier werden u.a. die Auswertungsparameter eingestellt, die Kommentarspezifikation der Programmiersprache festgelegt sowie die erlaubten Funktionen und Symbole aufgelistet.

exec ist ein Shell-Script und Implementiert für die Funktionen *query* und *eval* den Compiler-Aufruf. Hier ist darauf zu achten, dass der Resourcen-Verbrauch des Prozesses limitiert und die Priorität niedrig eingestellt wird, damit der Prozess die Machine nicht blockiert. Zudem sind Sicherheitsmaßnahmen gegen den Missbrauch dieser Schnittstelle zu bedenken.

Darüber hinaus muss für die Funktion *eval* ein Programm zur Verfügung gestellt werden, dass die IO-Relation des Lösungsvorschlag überprüft. In den implementierten Services wird dazu die erste Musterlösung benutzt und für eine Reihe von Testfällen die Ergebnisse beider Programme verglichen.

5 Zusammenfassung und Ausblick

Das Modul Vips (Virtuelles PrüfungsSystem) übernimmt in StudIP die komplette Verwaltung des Übungsbetriebs einer Veranstaltung. Vips wurde um eine Komponente VEA (Vips Evaluation Assistent) zur automatischen Bewertung von Programmieraufgaben erweitert. VEA stellt sowohl semantische (Programmverhalten) als auch syntaktische (Beurteilung des Programm-Codes) Verfahren zur Bewertung von Programmieraufgaben zur Verfügung. VEA ist immer noch in einem experimentellen Stadium. Sowohl die Funktionalität, als auch die Kommunikation zwischen Vips und VEA sollen weiterentwickelt werden.

Literatur

[1] Peylo, C.; Gust, H.; Rollinger, C.; Thelen, T. (2000): *A web-based intelligent educational system for PROLOG*. In: C. Peylo (Ed.), Proceedings of the International Workshop on Adaptive and Intelligent Web-Based Education Systems held in conjunction with ITS 2000 Montreal, Canada. Technical Report of the Institute for Semantic Information Processing, Osnabrück 2000.