aus

Oliver J. Bott, Peter Fricke, Uta Priss, Michael Striewe (Hrsg.)

Automatisierte Bewertung in der Programmierausbildung

Digitale Medien in der Hochschullehre Band 6
2017, 420 Seiten, br., 42,90 €, ISBN 978-3-8309-3606-0

Waxmann Verlag GmbH
www.waxmann.com info@waxmann.com

20 Integration automatisierter Programmbewertung in Stud.IP

Elmar Ludwig

Zusammenfassung
Das Lernmanagementsystem Stud.IP stellt – was ungewöhnlich ist – selbst noch keine Funktionen zur Erstellung von Aufgaben bereit, es gibt allerdings verschiedene Erweiterungsmodule, die in das System installiert werden können, um Aufgaben- und Bewertungsfunktionen nachzurüsten. Eine dieser Komponenten ist das „Vips“- Modul, das bereits eine Schnittstelle zu dem VEA Grader für Prolog (und einige andere Programmiersprachen) mitbringt. Die Einbindung dieser Schnittstelle in Vips als Stud.IP-Modul soll im Folgenden vorgestellt werden.

20.1 Stud.IP

Das Lernmanagementsystemen Stud.IP entstand 1999 an der Universität Göttingen als internetbasiertes System zur Unterstützung von Lehrveranstaltungen. Anfangs noch auf wenige Funktionen rund um die Themen Kommunikation (Forum), Datenaustausch (Dateiordner) und Organisation (Stundenplan) fokussiert, entstand im Laufe der Zeit ein umfangreiches Lernmanagementsystem, das mittlerweile von einer größeren Zahl von Hochschulen und Universitäten, aber auch außeruniversitäten Bildungseinrichtungen zur Organisation und Unterstützung des Lehrangebots eingesetzt wird.

Neben den klassischen Funktionen eines Lernmanagementsystems hinsichtlich der onlinebasierten Durchführung einer Lehrveranstaltung gibt es auch einzelne Funktionen aus dem Bereich des Campusmanagement – z. B. eine umfangreiche Raum- und Ressourcenverwaltung sowie Funktionen zur Personaldatenverwaltung, die Stud.IP ein breites Einsatzspektrum ermöglichen. Infolgedessen ist der Grad der Nutzung der einzelnen Komponenten an den verschiedenen Standorten zum Teil sehr unterschiedlich.

Zur tiefergehenden Unterstützung von elektronischen Lehrmaterialien gibt es noch eine Schnittstelle zum Lernmanagementsystem ILIAS (siehe Kapitel 21). Eine vergleichbare Schnittstelle zu Moodle ist gerade in der Entwicklung.

20.2 Das Vips-Modul in Stud.IP

Das onlinebasierte System Vips entstand aus einem Vorläufer, dem „Virtual Campus PROLOG Tutor“ ([Pey+00]), der ab 1997 an der Universität Osnabrück entwickelt und für die Ausbildung in der Programmiersprache Prolog eingesetzt wurde. Anfangs als reine Prolog-Umgebung konzipiert, wurde das System im Laufe der Zeit um die Funktionen eines vollwertigen E-Assessment-Systems erweitert, so dass heute eine breite Auswahl an unterschiedlichen Aufgabentypen und Einsatzszenarien unterstützt wird.

Seit dem Sommersemester 2004 ist es unter dem Namen „Vips“ (Abkürzung für „Virtuelles Prüfungssystem“) als onlinebasiertes Übungs- und Prüfungssystem in der Lehr- und Lernplattform Stud.IP verfügbar (siehe [Hüg+05]), in den ersten Versionen nur an der Universität Osnabrück, später dann auch an anderen Hochschulen. Als modulare Erweiterung kann es als Stud.IP-Plugin installiert und dann in einer Lehrveranstaltung von den Lehrenden als Inhaltselement aktiviert werden.

Das Modul bietet den Dozenten verschiedene Möglichkeiten zur Entwicklung und Verwaltung von Aufgabenblättern und Klausuren, inklusive der Unterstützung der eigenständigen Lernfortschrittskontrolle durch die Studierenden anhand von Selbsttests. Einige Aufgabentypen können vollautomatisch ausgewertet werden, bei anderen gibt es zumindest eine technische Unterstützung für den Auswertungsprozess, z. B. in Form von Bewertungsvorschlägen anhand der Ähnlichkeit einer abgegebenen Lösung zu einer erwarteten korrekten Lösung. Integriert ist auch eine Schnittstelle zum Import und Export von Aufgaben und Aufgabensammlungen in verschiedenen Formaten.

Studierende können Vips nutzen, um von Dozenten bereitgestellte Aufgabenblätter zu bearbeiten, entweder als Selbsttest mit unmittelbarem Feedback, in Form klassischer Hausaufgaben mit nachgelagerter (teils manueller) Bewertung und Freigabe durch Lehrende oder Tutoren oder auch in Form einer echten Online-Klausur – dann in der Regel unter Aufsicht in einem speziellen PC-Raum.

Vips bietet beim Erstellen von Tests eine Auswahl aus verschiedenen Aufgabentypen an:

Single Choice Eine Frage mit Auswahl aus einer Liste von vorgegebenen Antworten, nur eine Antwort kann gewählt werden.

Multiple Choice Eine Frage mit Auswahl aus einer Liste von vorgegebenen Antworten, mehrere Antworten können gewählt werden.

Ja/Nein Frage Eine spezielle Variante der Single Choice Aufgabe mit den Antwortmöglichkeiten „Ja“ und „Nein“.

Freitext Eine Frage mit freier Anwort (einzeilig oder mehrzeilig), die nur eingeschränkt automatisch ausgewertet werden kann.

Lückentext Ein Fragetext, in dem Lücken durch die Teilnehmer auszufüllen sind. Hier ist eine automatische Auswertung in den meisten Fällen möglich.

Mathematischer Term Eine Frage mit freier Anwort, die einen algebraischen Term erwartet, z. B. eine Formel. Die Antwort kann in den meisten Fällen automatisch ausgewertet werden.

Zuordnung Eine Aufgabe, bei der eine Liste von Begriffen, Texten oder Bilder einer Menge von Kategorien zugeordnet oder in eine bestimmte Reihenfolge gebracht werden soll.

Programmieraufgabe Aufgabe, die zur Lösung die Erstellung eines Programms erwartet. Aktuell werden von Vips nur Prolog Aufgaben unterstützt, weitere Grader zur Auswertung anderer Programmiersprachen sollen in Zukunft über die ProFormA-Middleware angebunden werden.

Das System bietet dabei umfangreiche Möglichkeiten zur Entwicklung, Pflege und Auswertung elektronischer Aufgaben sowie die Verwaltung des gesamten Übungsbetriebs eines Kurses. Dabei gibt es grundsätzlich Übungsblätter und Klausuren als Aufgabensammlungen innerhalb von Vips. 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 Selbsttestmodus, bei dem einem Kursteilnehmer nach dem Absenden einer Lösung sofort das Ergebnis der automatischen Auswertung präsentiert wird. Der Klausurmodus erlaubt eine Online-Überprüfung der Zugriffsparameter sowie eine Überwachung des Arbeitsfortschritts der Teilnehmer z. B. durch die Aufsicht.

Vips umfasst ebenfalls eine Arbeitsgruppenverwaltung (Aufgaben werden dann von mehreren Teilnehmern gemeinsam gelöst), Punkte- und Notenübersichten für einzelne Teilnehmer und Arbeitsgruppen im Kurs, sowie eine flexible Notenberechnung. Darüber hinaus gibt es auch statistische Übersichten über die Anzahl

der korrekten bzw. falschen Lösungen der Teilnehmer bei den einzelnen Aufgaben bzw. Aufgabenteilen. Aufgabenblätter und Klausuren können auch als Text- bzw. XML-Dateien importiert und exportiert werden. Darüber hinaus gibt es eine Druckansicht zum Ausdrucken der leeren Aufgabenformulare, der eigenen Lösung (für die Teilnehmer selbst) oder auch aller Lösungen im Kurs (für die Dozenten). Bild 20.1 zeigt die Ansicht zum Anlegen und Verwalten von Aufgabenblättern in Vips.

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

20.3 Programmieraufgaben in Vips

Vips stellt für Programmieraufgaben eine Runtime-Umgebung mit einer einfachen, webbasierten Oberfläche zur Verfügung. Diese GUI ist für die Teilnehmer direkt über einen Reiter einer Lehrveranstaltung erreichbar. Sie ermöglicht für Programmieraufgaben den Programmcode einzugeben und zu editieren sowie die Programme in einer definierten Umgebung auszuführen. So können Kursteilnehmer die Aufgaben lösen, ohne die Programmiersprachen lokal auf ihrem Rechner installieren zu müssen. Auch muss bei Online-Tests in PC-Räumen der Hochschule keine spezielle Umgebung zur Ausführung der Programme bereitgestellt werden. Allerdings stehen interaktive Debug-Möglichkeiten in einer solchen Umgebung nicht zur Verfügung – es werden nur die Meldungen des Compilers bzw. Interpreters bei der Ausführung zurückgemeldet. Um für komplexe Aufgabenstellungen auch lokale Installationen zum Aufgabenlösen benutzen zu können, gibt

Abbildung 20.1: Anlegen und Verwalten von Aufgabenblättern

es Down- und Upload-Möglichkeiten für die Aufgaben. In diesem Fall passiert dann oft nur noch die finale Abgabe der Lösung in Vips. Hierbei sollte aber durch die Lehrenden sichergestellt werden, dass die Teilnehmer auch lokal mit einer kompatiblen Version des Interpreters arbeiten, da die spätere Auswertung in Vips auf dem zentralen Auswertungssystem passiert und sonst ggf. andere Ergebnisse liefern könnte.

Die gleiche Runtime-Umgebung wie den Teilnehmern steht auch den Dozenten und Tutoren für die Vorbereitung der Aufgaben sowie bei der Aufgabenkorrektur zur Verfügung. Zusammen mit einem vorgegebenen Default-Aufruf der Hauptfunktion des Programms ermöglicht dies sowohl dem Kursteilnehmer als auch dem Korrektor, sich mit einem Mausklick einen ersten Überblick über die Lauffähigkeit der Lösung zu verschaffen. Eventuelle Fehler bei der Übersetzung oder Auswertung werden dem Teilnehmer angezeigt.

20.3.1 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 aus einer Text- oder XML-Datei importieren oder die Daten aus bereits vorhandenen Übungsblättern übernehmen. Ist das Übungsblatt in der Oberfläche mit Titel und Beschreibung sowie Start- und Endzeitpunkt versehen, können Übungsaufgaben neu hinzugefügt werden. Alternativ hat der Dozent hier auch die Möglichkeit, Aufgaben aus bereits im Stud.IP vorhandenen Übungsblättern dieses oder anderer Kurse zu übernehmen. Bild 20.2 zeigt ein bereits angelegtes Übungsblatt mit mehreren Aufgaben. Vips stellt für eine Programmierübungsaufgabe, die einem Aufgabenblatt hinzugefügt wird, neben der Aufgabenstellung eine Reihe von Informationsfeldern bereit (siehe Abbildung 20.3):

- Ein Lösungsfeld, in das der Teilnehmer den zu entwickelnden Code eingeben kann. Dieses Feld kann vorbelegt werden, so dass dem Kursteilnehmer bereits ein Lösungsgerüst präsentiert werden kann, das z. B. vom Bearbeiter ergänzt bzw. geändert oder korrigiert werden muss.

- Ein Feld für Musterlösungen. Dieses Feld kann mehrere Musterlösungen enthalten. Zudem enthält dieses Feld noch zusätzliche Informationen für die automatische Auswertung durch den Prolog-Grader. Das kann z. B. ein Testprogramm sein oder spezifische Angaben dazu, wie die Auswertung er-

folgen soll – beispielsweise nur durch einen strukturellen Vergleich mit den Musterlösungen. In Zukunft werden diese Informationen zur Auswertung in Vips in einem eigenen Textfeld verwaltet.

- Ein Feld mit dem Aufruf der Hauptfunktion des Programms. In Prolog ist dies eine Folge von Subgoals durch Kommas getrennt. In anderen Programmiersprachen kann dies ein Prozedur- bzw. Funktionsaufruf oder ein arithmetischer Ausdruck sein.

Da das Erscheinungsbild einer Aufgabenspezifikation aus der Sicht der Kursteilnehmer nicht immer einfach vorhergesehen werden kann, gibt es eine direkte Möglichkeit zwischen der Darstellung für die Aufgabenentwicklung und der Darstellung für den Kursteilnehmer umzuschalten. In der Darstellung aus Teilnehmersicht kann der Aufgabenersteller – wie die Teilnehmer später auch – die integrierte Runtime-Umgebung zum Testen der Aufgabe verwenden.

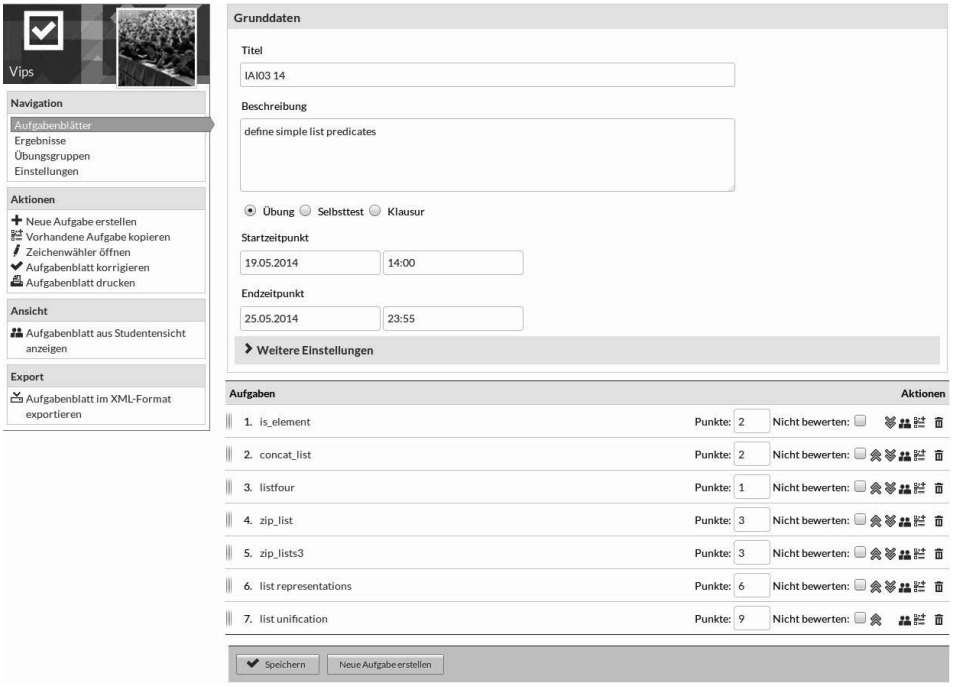

Abbildung 20.2: Übungsblatt mit mehreren Aufgaben

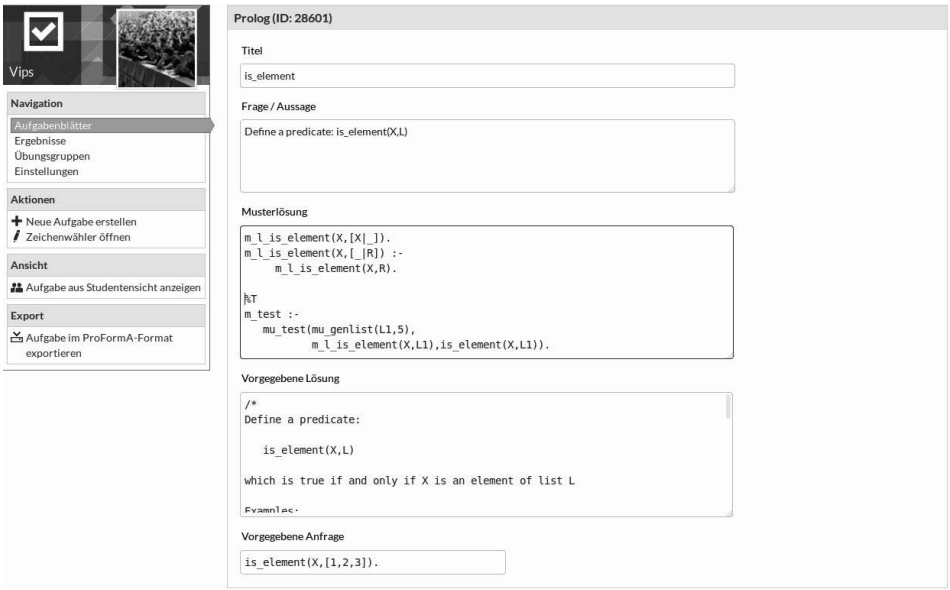

Abbildung 20.3: Erstellung einer Programmierübungsaufgabe

20.3.2 Bearbeiten eines Übungsblattes mit Programmieraufgaben

Generell können Programmieraufgaben, wie alle anderen Aufgabentypen auch, innerhalb des vom Dozenten beim Erstellen des Aufgabenblattes festgelegten Zeitraums bearbeitet werden. Aus Kursteilnehmersicht bietet die GUI (siehe Abbildung 20.4) dabei folgende Möglichkeiten:

- Der Kursteilnehmer kann seine Lösung in ein Textfeld eintragen. Dieser Bereich ist eventuell bereits mit einem Lösungsgerüst vorbelegt, das vom Teilnehmer erweitert oder korrigiert werden soll.

- Basierend auf der aktuellen Lösung im Textfeld kann eine Testanfrage gestellt werden. Diese wird auf dem Auswertungsserver ausgeführt und das Ergebnis des Programmlaufs angezeigt. Gab es Fehler bei der Ausführung, werden diese ebenfalls hier angezeigt.

- Der Inhalt des Lösungsbereichs kann in eine lokale Datei heruntergeladen werden, so dass eine lokale Installation der Programmiersprache und/oder

ein umfangreicherer Editor oder eine IDE zur Programmentwicklung benutzt werden kann.

- Anschießend kann die fertige Lösung wieder in das Bearbeitungsfeld hochgeladen werden.

- Über den Button *Abschicken* wird die eigene Lösung abgegeben. Übungsaufgaben können im Rahmen der Bearbeitungszeit mehrfach abgegeben werden, die als letztes abgegebene Lösung zählt.

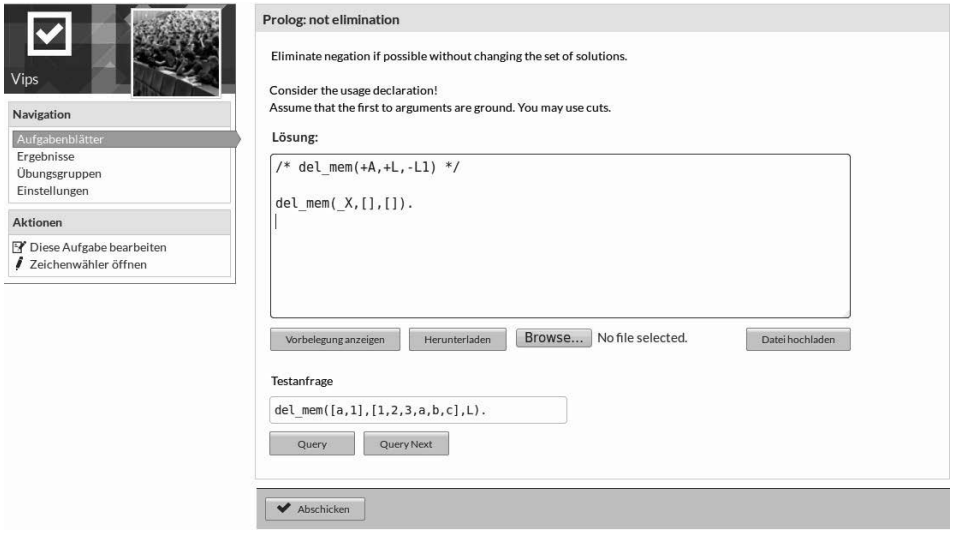

Abbildung 20.4: Bearbeitung einer Programmieraufgabe in Vips

Kursteilnehmer können in Arbeitsgruppen organisiert sein. Dann gilt bei einem Aufgabenblatt eine abgegebene Lösung für die gesamte Arbeitsgruppe, wobei jeder Teilnehmer der Gruppe die Gruppenlösung einsehen kann. Die jeweils letzte Lösung, die einer der Gruppenteilnehmer dabei abgegeben hat, zählt bei der Auswertung. Bei einer Klausur werden keine Gruppen berücksichtigt, hier arbeitet immer jeder Teilnehmer für sich.

Im Selbsttestmodus wird nach der Abgabe die Aufgabe automatisch ausgewertet und die Auswertung sowie die (erste) Musterlösung angezeigt. Ansonsten erfolgt die Auswertung durch den Dozenten und ist erst einsehbar, wenn die Bewertungen freigegeben wurden.

20.3.3 Auswertung eines Übungsblattes mit Programmieraufgaben

Alle abgegebenen Übungsblätter werden dem Dozenten in Vips in einer Übersicht präsentiert. Von hier aus gelangt er zu den einzelnen Übungsaufgaben bzw. den Lösungen der Teilnehmer.

Die Auswertung eines Übungsblattes beginnt normalerweise mit dem Aufruf der Autokorrekturfunktion, die alle automatisch auswertbaren Aufgabentypen zu bewerten versucht. Bei Programmieraufgaben liefert die automatische Auswertung aber oft nur einen Indikator für die Bewertung, daher stellt die Vips-Oberfläche im Korrektur- und Bewertungsmodus ähnliche Funktionen wie bei der Bearbeitung durch die Studierenden zur Verfügung – inklusive der Runtime-Umgebung zur Nutzung durch den Dozenten, hier aber noch erweitert um die Möglichkeiten zur Annotation der Lösung und zur Punktevergabe. Innerhalb des Auswertungsmodus einer Programmieraufgabe gibt es folgende Funktionen:

- Im abgegebenen Programmcode kann korrigiert und kommentiert werden. Dies kann beispielsweise nützlich sein, wenn die vom Teilnehmer abgegebene Lösung einen Syntaxfehler enthält, der eine automatische Bewertung des Progamms verhindert. Der Dozent oder Tutor kann dann diesen Fehler korrigieren und die Bewertung anschließend noch mal durchführen.

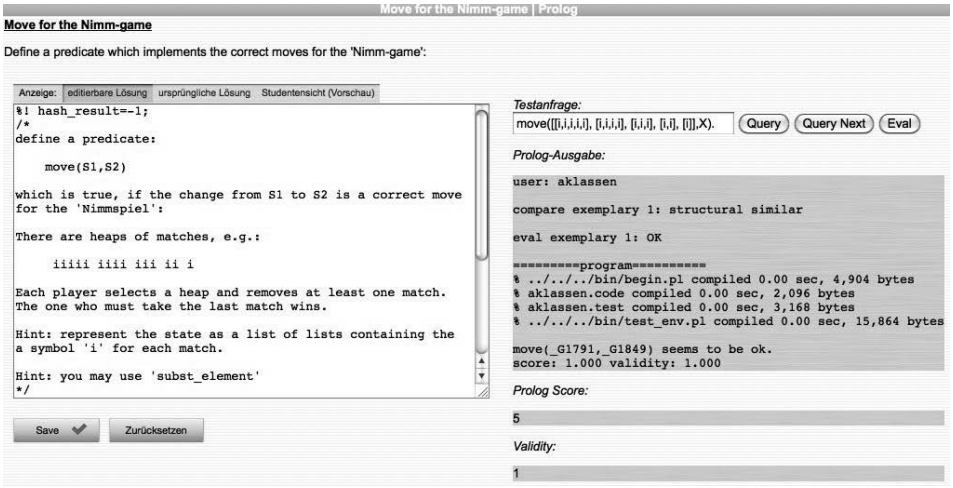

Abbildung 20.5: Auswertungsmodus einer Programmieraufgabe in Vips

Die Originallösung des Teilnehmers wird dabei aber nicht verändert – diese bleibt zur Dokumentation der abgegeben Lösung unverändert, die annotierte Lösung wird immer separat gespeichert.

- In einem Textfeld können allgemeine Kommentare zur Lösung oder Erläuterungen zur Bewertung hinzugefügt werden.

- Für die Lösung können Punkte vergeben werden. Dabei werden zunächst als Vorschlag die von der automatischen Auswertung vergebenen Punkte angezeigt.

- Der *Query*-Knopf sendet den Inhalt des Lösungsfelds an den Auswertungsserver und ruft dort die Hauptfunktion auf. Die Ergebnistexte der Kompilation und der Programmausführung werden anschließend als Text angezeigt. Für nichtdeterministische Sprachen wie Prolog gibt es noch den Knopf *Query Next*, der bei jeder Betätigung eine weitere Lösung des Programms abfragt und ausgibt.

- Der *Eval*-Knopf stößt die automatische Bewertung an (siehe Abbildung 20.5). Der Bewertungprozess selbst kann dabei über spezielle Schlüsselworte in der Musterlösung gesteuert werden, so kann z. B. die Bewertung nur auf einen Textvergleich beschränkt oder durch ein Testprogramm vorgenommen werden – die Details dazu sind im Kapitel 14 beschrieben.

  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 Bewertungskriterien können die I/O-Relation (also das Verhalten) des Programms sowie textuelle Vergleiche des Lösungscodes mit den Musterlösungen auf verschiedenen Normalisierungsebenen herangezogen werden.

20.4 Im- und Export

Zusätzlich zu der Möglichkeit, Aufgaben direkt im System zwischen Nutzern auszutauschen, lassen sich auch alle Aufgaben in verschiedenen Formaten importieren und exportieren. Für den Im- und Export von Programmieraufgaben werden im wesentlichen zwei Formate unterstützt:

Vips-Format

Das Vips-Format ist ein XML-basiertes Format speziell für den Datenaustausch von Aufgaben zwischen verschiedenen Vips-Installationen. Daneben kann auch das ältere, nicht XML-basierte Textformat weiterhin eingelesen werden, auf das hier nicht weiter eingegangen wird.

ProFormA-Task

Das ProFormA-Format ist ein Austauschformat speziell für Programmieraufgaben, das auch von anderen Werkzeugen unterstützt wird. Bisher ist Vips allerdings das einzige System, das Grader-Unterstützung für Aufgaben in Prolog enthält.

Daneben gibt es in Vips noch einen Export in das QTI 2.1 Format, der allerdings nicht alle Aufgabentypen darstellen kann – insbesondere werden von QTI keine Programmieraufgaben unterstützt.

20.4.1 Vips-Format

Das Vips-Format ist ein eigenes, XML-basiertes Austauschformat für alle Aufgabentypen. Das Schema ist auf Basis eines Aufgabenblattes definiert (d. h. einer Aufgabensammlung), der Export einzelner Aufgaben ist dabei eigentlich nicht vorgesehen. Daher eignet es sich auch nur bedingt zum Austausch isolierter Aufgaben mit anderen Werkzeugen bzw. Gradern – hierfür ist das im nächsten Kapitel dargestellte ProFormA-Format besser geeignet.

Generell besteht die Darstellung einer Programmieraufgabe aus verschiedenen Metadaten (Titel, Aufgabentext, Aufgabentyp) sowie dem enthaltenen Programmcode. Im Einzelnen sind das:

- Die Textvorgabe, die Grundlage für die Lösung der Teilnehmer ist.
- Eine oder mehrere Musterlösungen mit Angaben dazu, wie gut eine solche Lösung zu bewerten ist.
- Sowie optional auch Tests, die in die automatische Bewertung eingehen (in dem folgenden Beispiel nicht zu sehen).

Schließlich gibt es noch die Möglichkeit, einen Programmaufruf zu hinterlegen, mit dem die Teilnehmer in der interaktiven Runtime-Umgebung in Vips während der Entwicklung ihre Lösungen ausprobieren können.

Abbildung 20.6 zeigt exemplarisch den Export einer Programmieraufgabe in Prolog.

```
<exercise id="exercise-37636">
  <title>
    Reverse Elements of a List
  </title>
  <description>
    <text>
      Define a predicate 'rev(L1,L2)' which is true if L2
      contains the elements of L1 in reverse order.
    </text>
  </description>
  <items>
    <item type="program-prolog">
      <answers>
        <answer score="0" default="true">
          <text>
            rev( )
          </text>
        </answer>
        <answer score="1">
          <text>
            m_l_rev([], []).
            m_l_rev([X | Rest], Result) :-
            m_l_rev(Rest, RevRest),
            append(RevRest, [X], Result).
          </text>
        </answer>
      </answers>
      <evaluation-hints>
        <input-data type="prolog-query">
          time(rev([1,2,3,4,5,6,7,8,9,0], L)).
        </input-data>
      </evaluation-hints>
    </item>
  </items>
</exercise>
```

Abbildung 20.6: Beispiel für den Export einer Aufgabe im Vips-Format

20.4.2 ProFormA-XML

Das Exportformat ProFormA-XML wird in Vips primär zum Austausch von Aufgaben mit anderen Lernplattformen und zur Kommunikation mit der Middleware bei der zukünftigen Anbindung weiterer Grader verwendet – die Schnittstelle zum Grader VEA benutzt noch nicht das ProFormA-XML-Format. Dieses Format ist immer aufgabenbezogen, d. h. es kann (anders als beim zuvor beschriebenen Vips-Format) auch immer nur jeweils eine Aufgabe in einer Datei beschrieben werden.

```
<task xmlns="urn:proforma:task:v0.9.4" lang="en">
  <description>
    <h1>Reverse Elements of a List</h1>
    Define a predicate 'rev(L1,L2)' which is true if L2
    contains the elements of L1 in reverse order.
  </description>
  <proglang version="SWI-Prolog 5.10.1">
    prolog
  </proglang>
  <files>
    <file id="answerDefault" class="template" filename="ad.pl"
    type="embedded">
      rev( )
    </file>
    <file id="queryDefault" class="inputdata" filename="qd.pl"
    type="embedded">
      time(rev([1,2,3,4,5,6,7,8,9,0], L)).
    </file>
    <file id="m_l_0" class="internal" filename="m_l_0.pl" type="embedded">
      m_l_rev([], []).
      m_l_rev([X | Rest], Result) :-
      m_l_rev(Rest, RevRest),
      append(RevRest, [X], Result).
    </file>
    <file id="test" class="internal" filename="test.pl" type="embedded">
      m_test :-
      mu_test(mu_genlist(L1,5),
      m_l_rev(L1,L2),rev(L1,L2)).
    </file>
  </files>
  <model-solutions>
    <model-solution id="m_s_0" filename="m_s_0.pl" type="embedded">
      rev([], []).
      rev([X | Rest], Result) :-
      rev(Rest, RevRest),
      append(RevRest, [X], Result).
    </model-solution>
  </model-solutions>
  <tests>
    <test id="test_0" validity="1">
      <title>Test</title>
      <test-type>
        prolog-eval-similarity
      </test-type>
      <test-configuration>
        <filerefs>
          <fileref refid="answerDefault" />
          <fileref refid="m_l_0" />
          <fileref refid="test" />
        </filerefs>
      </test-configuration>
    </test>
  </tests>
</task>
```

Abbildung 20.7: Beispiel für den Export einer Aufgabe im ProFormA-Format

Weitere Informationen zum ProFormA-XML-Format selbst sind in Kapitel 24 zu finden.

Neben den bereits im Vips-Format gezeigten Metadaten zur Aufgabe kann in ProFormA-XML noch die Information zur Version des verwendeten Interpreters (hier: *SWI-Prolog 5.10.1*) explizit hinterlegt werden.

Dies ist wichtig, damit bei der Auswertung sichergestellt werden kann, dass auch ein mit dem Code in der Aufgabe kompatibler Interpreter verwendet wird.

Nach den Metadaten folgt eine Liste von Programmen, denen jeweils unterschiedliche Rollen zugewiesen sein können – eine Vorbelegung für die Teilnehmer, eine Liste von Musterlösungen sowie ein Beispielaufruf für die interaktive Runtime-Umgebung in Vips. Anschließend folgt die Liste der Tests für die automatische Programmbewertung. Generell sind bei aus Vips exportierten Aufgaben diese Programme immer in die XML-Struktur eingebettet (*type="embedded"*), auch wenn das Format die Auslagerung in separate Dateien ermöglichen würde.

Jeder aufgelistete Test kann dabei noch spezifische Optionen für die Auswertung enthalten, so gibt es z. B. folgende Prolog-spezifische Werte für den *test-type*:

prolog-similarity Auswertung ausschließlich anhand der Textähnlichkeit zu einer der Musterlösungen.

prolog-eval Auswertung ausschließlich anhand der Ausgabe des Testprogramms.

prolog-eval-similarity Auswertung anhand beider obiger Kriterien.

Abbildung 20.7 zeigt den Export einer Prolog-Programmieraufgabe in ProFormA-XML. Dieses Format wird zukünftig auch für die Anbindung an den ProFormA-Server (siehe Kapitel 22) verwendet.

Literatur für dieses Kapitel

[Hüg+05] Philipp Hügelmeyer u. a. „Integration des Virtuellen Prüfungssystems ViPS in die Lehr-/Lernplattform Stud. IP“. In: *Proceedings of the Workshop on e-Learning*. 2005, S. 187–196.

[Pey+00] Christoph Peylo u. a. „A Web-based intelligent educational system for PROLOG“. In: *Proceedings of the International Workshop on Adaptive and Intelligent Web-Based Education Systems held in conjunction with ITS 2000 Montreal, Canada.* 2000, S. 85–96.