LaplaceScript: Eine XML–basierte Skriptsprache für interaktive Übungsaufgaben

Barbara Grabowski, Susanne Gäng, Jörg Herter, Thomas Köppen

Hochschule für Technik und Wirtschaft des Saarlandes, Fachbereich GIS
Göbenstraße 40
66117 Saarbrücken, Germany
{ grabowski | sgaeng | pi.joerg.herter | ki.thomas.koeppen }@htw-saarland.de

Abstract: Das speziell auf mathematische Inhalte ausgelegte E-Learning-Werkzeug MathCoach besteht praktisch aus zwei Teilen: dem MathCoach-Servlet, das das Rahmenwerk bildet, und der Aufgabenbeschreibungssprache LaplaceScript, mit deren Hilfe Autoren ihre Übungsaufgaben verfassen.
Das MathCoach-Servlet oder einfach MathCoach bildet das eigentliche E-Learning-System. MathCoach stellt eine Umgebung zur Verfügung, in der in LaplaceScript erstellte interaktive Übungsaufgaben ausgeführt werden können. LaplaceScript ist eine kompakte, auf Praxistauglichkeit ausgelegte XML-Anwendung. Die Sprache besitzt alle Merkmale einer *echten* Programmiersprache, ist aber um Längen schneller erlernbar und intuitiver nutzbar als traditionelle Programmiersprachen.

1 Was ist LaplaceScript?

LaplaceScript entstand aus dem Wunsch heraus, möglichst flexible interaktive Übungsaufgaben beschreiben und implementieren zu können, ohne erst eine komplette Programmiersprache erlernen zu müssen. So bietet LaplaceScript zwar alle Möglichkeiten einer traditionellen Programmiersprache, minimiert aber den erforderlichen Lernaufwand.

LaplaceScript wurde speziell zur Beschreibung mathematischer Aufgaben und Inhalte konzipiert. So existieren in LaplaceScript zum Beispiel die mathematischen Datenstrukturen Vektoren und Matrizen sowie Sprachelemente zur direkten Kommunikation mit dem Computeralgebrasystem MuPAD und dem R-Statistik-Paket. Das Einbinden der unter Mathematikern bekannten Systeme R und MuPAD erweitert enorm die Möglichkeiten, die LaplaceScript Autoren bietet. Bereits bekannte Systeme aus LaplaceScript heraus nutzen zu können, trägt weiter auch dazu bei, die Akzeptanz des neuen E-Learning-Systems unter den zukünftigen Autoren zu erhöhen. Etwas Gewohntes und Bekanntes aus einer neuen Umgebung heraus zu nutzen, fällt sicherlich leichter, als noch mehr Neues zu lernen, um bereits Bekanntes neu umzusetzen.

Die Sprache selbst ist XML-basiert und wie XHTML oder MathML eine XML-Anwendung. XML-Kenntnisse erleichtern somit zwar den Einstieg in LaplaceScript, sind aber nicht unbedingt erforderlich, da die Sprache sehr einfach und intuitiv benutzbar gehalten wurde. Die Sprache auf XML-Basis zu implementieren erzwang mehr oder weniger ein weiteres Ziel, das für LaplaceScript gesteckt wurde: Autoren, die mit dieser Sprache arbeiten, programmieren keine Übungsaufgaben, sie beschreiben sie vielmehr.

Dennoch weist LaplaceScript auch alle wesentlichen Merkmale einer *echten* Programmiersprache auf, um den Autoren größtmögliche Freiheit beim Erstellen von Übungsaufgaben zu geben. In LaplaceScript sollte alles umsetzbar sein, was sich ein Autor als Übung ausdenken kann. Um dieser Anforderung gerecht zu werden, verfügt die Sprache über verschiedene Variablentypen wie skalare Variablen, Listen/Vektoren und Matrizen. Sie besitzt Sprachelemente zum Beschreiben von Schleifen und Verzweigungen und bietet Möglichkeiten der Codewiederverwendung, indem sie es gestattet, eigene wiederverwendbare Prozeduren zu definieren.

Ein weiterer Punkt, der für die Entwicklung der Sprache von großer Bedeutung war, ist die Performanz der gesamten Anwendung. Aus diesem Grund werden Laplace-Skripte nicht zur Laufzeit interpretiert, sondern aus den Aufgabenbeschreibungen werden Java-Klassen erzeugt, die wiederum innerhalb von MathCoach ausgeführt werden. Dadurch ist LaplaceScript auch um einiges schneller als vergleichbare Interpretersprachen.

LaplaceScript versucht vor allem für Personen ohne Programmiererfahrung intuitiv nutzbar zu sein. Es ist fehlertoleranter als die meisten bekannten Programmiersprachen, z.B. was das Anlegen von Matrizen mit falscher Anzahl von Initialwerten betrifft und versucht wo immer möglich implizit vorhandenes Wissen zu nutzen. Ein Beispiel dazu ist die Variablenbehandlung. Hier wird selbständig je nach Situation erkannt, welchen Inhalt eine Variable haben muss (Zahlenwerte oder Zeichenketten), ohne dass dies explizit im LaplaceScript-Code angegeben werden muss. Durch seinen im Vergleich zu traditionellen Programmiersprachen eher beschreibenden Charakter unterstützt LaplaceScript Autoren in soweit, als dass die Programmierung in LaplaceScript mehr einer natürlichsprachigen Beschreibung als einer Umsetzung einer Aufgabe in eine Maschinensprache entspricht.

2 Aufbau einer LaplaceScript-Datei

Ausgangspunkt für die Entwicklung von LaplaceScript war ein Ansatz zur Integration von Interaktionen in OMDoc, der bei ActiveMath [BGOM04] verfolgt wurde. Die dort vorhandene, strukturelle Gliederung in *startup-*, *interaction-* und *support-*Teil wurde übernommen. Allerdings waren die Sprachelemente für unsere Anwendungen nicht hinreichend, was zur Entwicklung von LaplaceScript als eigenständige Beschreibungssprache für interak-

tive Übungsaufgaben und MathCoach als Laufzeitumgebung für diese Aufgaben führte. So erweitert LaplaceScript den ursprünglichen Ansatz z.B. um Prozeduren und Kontrollstrukturen.
LaplaceScript-Dateien haben immer den folgenden Aufbau:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<laplace>
    <procedure name="...">
        ...
    </procedure>

    ...

    <startup>
        ...
    </startup>

    <interaction>
        ...
    </interaction>

    <support>
        ...
    </support>

</laplace>
```

Die erste Zeile, die XML-Deklaration, ist nur optional. Sie ermöglicht jedoch die Angabe der benutzten Text-Kodierung. In obigem Beispiel ist dies die Oktettkodierung ISO-8859-1, die den US-ASCII-Zeichensatz um verschiedene europäische und andere Schriftzeichen, unter anderem auch um die deutschen Umlaute, erweitert.
Das Wurzelelement *laplace* enthält alle in dem Skript definierten Funktionen bzw. Prozeduren. Neben beliebigen eigenen Prozeduren müssen mindestens die Prozeduren *startup*, *interaction* und *support* vorhanden sein.
In der *startup*-Prozedur beschreibt der Autor, was beim Starten (oder Neustarten) des Skripts geschehen soll. Diese Stelle ist zum Beispiel für Variableninitialisierungen oder zur Ausgabe der Aufgabenstellung besonders geeignet.
Die Prozedur *interaction* wird jedesmal abgearbeitet, wenn der Benutzer eine Eingabe tätigt. In ihr beschreibt der Autor also, wie auf Benutzereingaben reagiert bzw. wie mit diesen verfahren werden soll. Grundsätzlich enthalten *startup* und *interaction* beliebige Elemente der Gruppe *Statements*. Es sind jedoch einige wenige *Statement*-Elemente wie z.B. *restart* für den *startup*-Bereich gesperrt.
Im *support*-Element wird das Hilfesystem der Aufgabe beschrieben. Die Hilfestellungen je Lösungsstufe sind hierbei hierarchisch in einer Baumstruktur organisiert. So enthält das *support*-Element als Kindelemente zunächst pro Teilaufgabe ein *subtask*-Element. Innerhalb eines *subtasks* wird weiter in verschiedene Niveaustufen unterteilt, die die einzelnen

Teillösungen bzw. korrekte Lösungsschritte der Aufgabe repräsentieren. Hierzu dient das *level*-Element. Innerhalb eines *levels* können mittels *define-help* beliebig viele Hilfestellungen angegeben werden, die später in der angegebenen Reihenfolge ausgegeben werden können.

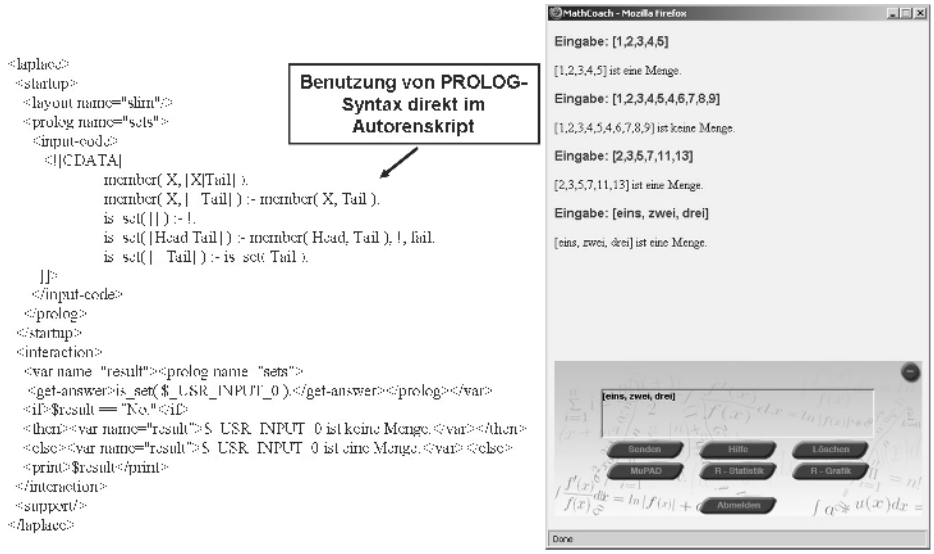

Abbildung 1: Beispiel eines Laplace-Skripts mit integriertem Prolog-Code und ein Screenshot der zugehörigen MathCoach-Konsole.

3 Editoren für LaplaceScript

LaplaceScript ist eine eigene Sprache, die erst einmal gelernt werden muss. Potentielle Autoren könnten folglich davon abgeschreckt werden, eine Sprache lernen zu müssen. Daher ist der Einsatz von möglichst transparenten Autorenwerkzeugen empfehlenswert. Ein solches Autorenwerkzeug sollte in der Lage sein, den Autor nur dann mit dem eigentlichen XML-Code zu konfrontieren, wenn dieser dies möchte oder es nicht anders möglich ist. Es sollte Autoren Schritt für Schritt durch die Erstellung eines Laplace-Skripts begleiten und allgemeine Aufgabentypen als vordefinierte Schablonen, sogenannte Templates, bereit halten. Auch sollten größere LaplaceScript-Konstrukte wie zum Beispiel Schleifen als Templates vorhanden sein. Weiter sollten Hilfen zu jedem LaplaceScript-Element von einem solchen Autorenwerkzeug jederzeit angezeigt werden können.

Konkret werden zur Zeit drei mögliche Ansätze verfolgt, ein solches Autorenwerkzeug umzusetzen.

Editor-Plug-Ins Ein erster Ansatz ist die Umsetzung des Autorenwerkzeugs als Plug-In für einen XML-fähigen Editor wie z. B. Eclipse oder jEdit. Ein solches Plug-In würde die benötigten Funktionalitäten für diese Editoren nachrüsten. Derzeit wird daran gearbeitet, ein solches Plug-In für Eclipse umzusetzen.

Oxygen Die Benutzung des Editors bietet eine einfache XML-Unterstützung auf Basis des LaplaceScript-XML-Schemas. Dieser Editor kann also automatisch Pflichtelemente und -attribute ins Dokument einfügen und an jeder Stelle des Skripts eine Auswahl derjenigen Elemente anzeigen, die an dieser Stelle einsetzbar sind. Autoren arbeiten aber weiterhin auf XML-Code-Basis und benötigen eine kostenpflichtige Lizenz für Oxygen. Auch Templates können hiermit nicht umgesetzt werden. Abbildung 2 zeigt einen Screenshot dieses Editors.

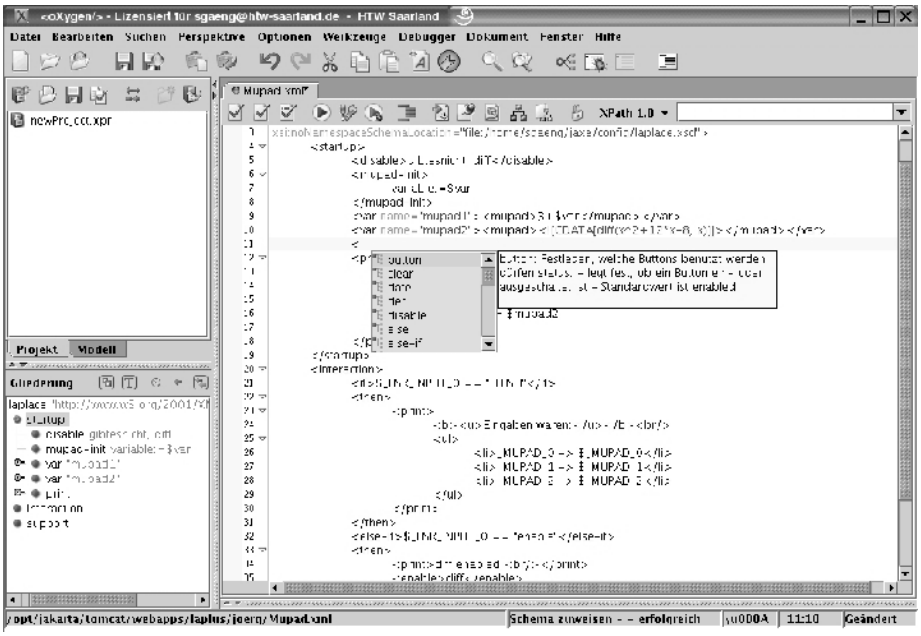

Abbildung 2: Bearbeitung einer LaplaceScript-Datei in Oxygen.

Jaxe Jaxe ist ein XML-Editor, der sich an eine gegebene XML-Sprache anpasst, indem er die DTD bzw. das XML-Schema der Sprache einliest und seine Oberfläche entsprechend aufbaut. Jaxe ist als OpenSource unter der GPL veröffentlicht und somit kostenlos verfügbar und beliebig erweiter- und anpassbar. Der eigentliche XML-Code bleibt bei Jaxe dem Autor verborgen. Templates und die Nutzung von XHTML und MathML zusätzlich zu LaplaceScript müssen bei Jaxe noch nachträglich implementiert werden. Abbildung 3 zeigt einen Screenshot dieses Editors umgesetzt für LaplaceScript.

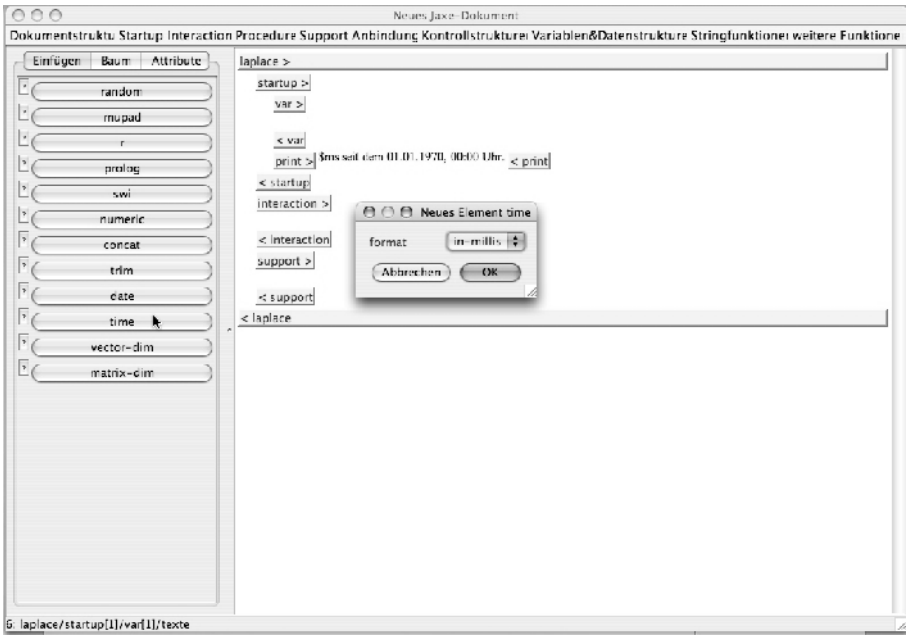

Abbildung 3: Bearbeitung einer LaplaceScript-Datei in Jaxe.

4 Zusammenfassung und Ausblick

LaplaceScript ist eine leicht zu erlernende Beschreibungssprache für interaktive Übungsaufgaben. Durch Kopplung an externe Systeme erweist sich LaplaceScript im praktischen Einsatz als mächtiges Autorenwerkzeug. Momentan konzentriert sich die Weiterentwicklung auf Werkzeuge zur Unterstützung der Autoren beim Erstellen von Übungsaufgaben. Weiter wird derzeit an der Erstellung einer Sammlung von in LaplaceScript verfassten Übungsaufgaben zum Propädeutikum Mathematik gearbeitet.

Literatur

[BGOM04] P. Baumgartner, B. Grabowski, W. Oevel und E. Melis. In2Math – Interaktive Mathematik- und Informatikgrundausbildung. Publikation auf der Projekt–Homepage: http://www.in2math.de, 2004.

[Her05] Jörg Herter. Konzeptionierung und Implementierung eines E-Learning-Werkzeugs einschließlich Autorensprache, 2005. Diplomarbeit an der Hochschule für Technik und Wirtschaft des Saarlandes.