Studiengang:	Informatik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuerin:	Dr. Waltraud Schweikhardt
Beginn am:	10.02.2003
Ende am:	12.08.2003
CR-Nummer:	H.5.2, H.5.4, I.7.2, K.3.1, K.4.2

Diplomarbeit Nr. 2077

Ein software-ergonomisches Werkzeug zum Erstellen von e-Learning-Kursen am Beispiel eines Lernprogramms zur Steigerung der Wahrnehmungsleistung sehbehinderter Kinder

Swen Mühlleitner

Universität Stuttgart
Fakultät Informatik, Elektrotechnik und Informationstechnik
Institut für Visualisierung und Interaktive Systeme
Universitätsstraße 38
70569 Stuttgart

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Die vorliegende Diplomarbeit ist als Ganzes im Zusammenhang mit der Studienarbeit Nr. 1852 [Co/Mü] und der Diplomarbeit Nr. 2076 [Conradt] zu sehen. Ziel dieser Arbeiten ist es, ein Autorensystem zu entwickeln, welches eine ergonomische Bedienung besitzt und des weiteren für das Zusammenstellen unterschiedlicher Lernprogramme geeignet ist.

In der Studienarbeit Nr. 1852 [Co/Mü] wurde hierbei festgestellt, dass die bestehenden Autorensysteme, entweder eine unhandliche Bedienung haben oder teilweise die nötige Funktionalität (wie z.B. das Erstellen von Prüfungen) vermissen lassen. Im Rahmen der drei Arbeiten sollen nun die Punkte einfache Bedienung und benötigte Funktionalität zusammengeführt werden, damit ein Autorensystem entsteht, welches nicht ausschließlich aber hauptsächlich in Bildungseinrichtungen Verwendung finden soll.
Hierzu wurde in der Studienarbeit Nr. 1852 [Co/Mü] der Kern des Autorensystems entwickelt, mit welchen es möglich ist, eine einzelne Lerneinheit eines Lernprogramms zusammenzustellen. Dabei wurde gleichzeitig das Autorensystem so entwickelt, das es ein offenes System darstellt. Dies bedeutet, dass es möglich ist, dass das Autorensystem mit Komponenten erweiterbar ist, ohne dabei den Quellcode des eigentlichen Autorensystems verändern zu müssen. Unter Komponente ist hierbei eine einzelne Frageart bzw. eine einzelne Präsentationsart für die Darstellung von Lerninhalten gemeint. Durch dieses Konzept können beliebige Komponenten entwickelt werden, die genau für eine Art von Lernprogramm bestimmt sind, was den Vorteil bietet, dass ein Autor sein Lernprogramm genau nach seinen Wünschen zusammenstellen kann, wenn die erforderlichen Komponenten zur Verfügung stehen.
Von diesem Konzept wurde im Rahmen dieser Diplomarbeit Gebrauch gemacht, indem für das Lernprogramm zur Steigerung der Wahrnehmungsleistung sehbehinderter Kinder eigene Komponenten entwickelt wurden, die zu einem den Lernstoff darbieten sollen und zum anderen wurde eine spezielle Bildfrage entwickelt, die in diesem Lernprogramm Verwendung findet.

Das Lernprogramm stellt dabei nur ein Beispiel dar, um die Funktionsweise des Autorensystems aufzeigen zu können. Die Hauptaufgabe dieser vorliegenden Diplomarbeit ist es hingegen, ein Werkzeug zur Festlegung des Ablaufs einer Lernsitzung zu entwickeln. Damit ist gemeint, dass dem Autor die Möglichkeit gegeben werden soll, die einzelnen Lerneinheiten in einer sinnvollen Reihenfolge zu einem Lernprogramm zusammenstellen zu können. Da das Autorensystem hauptsächlich auf die Zielgruppe von Lehrern abzielt, wird dabei darauf Wert gelegt, dass die Bedienung dieses Werkzeugs einfach gehalten wird, um die Autoren nicht durch unnötige Funktionalitäten zu verwirren.

Im nächsten Kapitel wird nun das Werkzeug vorgestellt, welches der Festlegung der Ablaufsitzung eines Lernprogramms dient. Das Werkzeug lehnt sich dabei an den Aufbau eines Lernprogramms an, welches aus Lektionen und Lerneinheiten besteht. Bei der Entwicklung wurde außerdem darauf acht gelegt, dass die einzelnen Elemente des Lernprogramms weitestgehend wieder verwendbar sind.

Anschließend wird im Kapitel 3 die technische Umsetzung erklärt, wobei hier verdeutlicht wird, dass es sich bei dem Autorensystem um zwei Programme handelt. Das erste Programm, das eigentliche Autorensystem, ist dabei für das Zusammenstellen des Lernprogramms zuständig, während das zweite Programm den Ablauf des Lernprogramms im Browser regelt.

In Kapitel 4 schließlich wird ein Lernprogramm vorgestellt, welches mit dem Autorensystem zusammengestellt wurde. Es handelt sich um ein Lernprogramm, welches für Kinder mit einer Sehbehinderung gedacht ist und welches die Wahrnehmungsleistung dieser Kinder steigern soll. Unter Sehbehinderung ist hierbei gemeint, dass das Kind Probleme bei der Verarbeitung visueller Reize hat.

Danach wird im letzten Kapitel eine kleine Schlussbetrachtung über das entwickelte Autorensystem gegeben. Außerdem werden Vorschläge für eine mögliche Weiterentwicklung aufgezeigt.

2. Konzeption zur Festlegung des Ablaufs einer Lernsitzung

In diesem Kapitel wird das Konzept zur Festlegung des Ablaufs einer Lernsitzung vorgestellt. Hierzu wird zuerst gezeigt, wie ein Lernprogramm bzw. Kurs aufgebaut ist. Danach wird der Begriff Autorensteuerung eingeführt und anschließend wird die Bedienung des Autorensystem im Hinblick auf software-ergonomische Kriterien dargestellt. Abschließend gibt es eine kleine Zusammenfassung über die Funktionsweise der Autorensteuerung.

2.1 Aufbau eines Lernprogramms

In der Regel besteht ein Lernprogramm bzw. ein Kurs aus mehreren Lektionen, die wiederum aus einzelnen Lerneinheiten bestehen. Durch diese Struktur ist es möglich, Wissensgebiete zu gruppieren und in geeigneten Portionen zu vermitteln. So kann in einer einzelnen Lerneinheit ein kleiner ausgewählter Bereich eines Wissensgebiet vermittelt werden. Gleichartige Lerneinheiten, die in denselben Bereich der Wissensvermittlung fallen, können in der übergeordneten Struktur Lektion zusammengefasst werden. Diese wiederum können dann zu einem Kurs zusammengebaut werden, der ein ganzes Wissensgebiet abdeckt. Als Beispiel kann man sich einen Kurs „Mathematik“ vorstellen, der aus mehreren Lektionen wie z.B. Grundrechenarten, Bruchrechnung, Mengenlehre, etc. besteht. In der Lektion Grundrechenarten könnten dann in den einzelnen Lerneinheiten Addieren, Subtrahieren, Multiplizieren und Dividieren behandelt werden. Natürlich ist auch eine andere Struktur möglich, je nachdem, was der Lehrer bzw. der Autor des Lernprogramms zu vermitteln wünscht.
In Abbildung 1 ist eine schematische Darstellung eines Kurses gegeben. Dabei ist zu sehen, dass der Ablauf eines Kurses durch die Bewertungen der Lektionen bestimmt wird. D.h. je nachdem wie viele Punkte der Lernende in den einzelnen Lerneinheiten einer Lektion erreicht hat, wird dieser zu einer anderen Lektion geleitet. Dadurch ist es dem Lehrer bzw. dem Autor des Kurses möglich, seinen Kurs so zu gestalten, dass gute Lernende mit einer anderen Lektion fortfahren, als weniger gute. Es ist also möglich den Kurs mit verschiedenen Schwierigkeitsstufen zu versehen.

Letztendlich gelangen aber alle Wege des Kurses zur Kurs-Bewertung, die den Abschluss eines jeden Kurses bildet. Ähnlich wie bei den Bewertungen der Lektionen liefert die Kurs-Bewertung eine Rückmeldung über die erbrachten Leistungen. Kurs und Lektion stellen demnach die zwei obersten Ebenen eines Kurses dar.

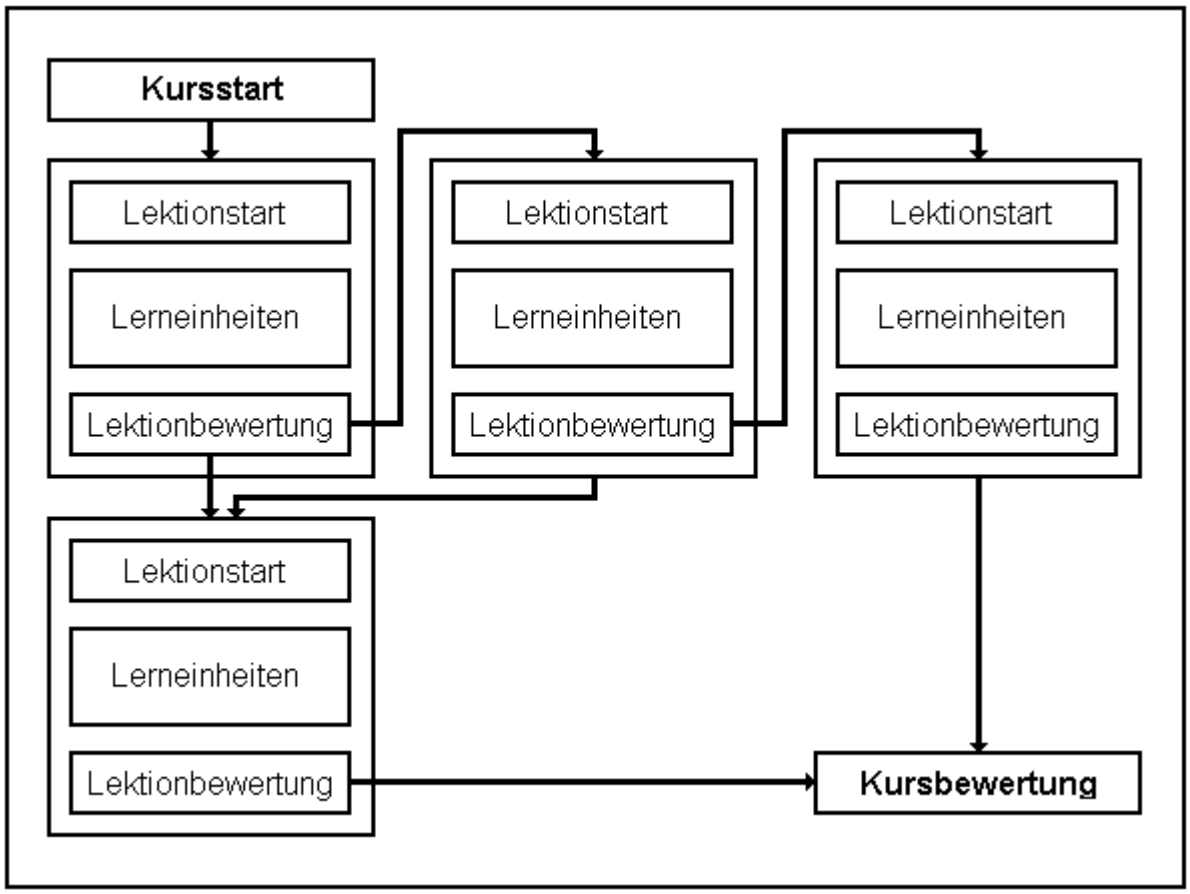

Abbildung 1: Schematische Darstellung eines Kurses

Die zwei untersten Ebenen sind die Lerneinheit und die Komponenten einer Lerneinheit. Die Ebene Lerneinheit wurde in der Studienarbeit Nr. 1852 [Co/Mü] „Übersicht und Untersuchung existierender Autorensysteme und Entwurf für ein System zum Erstellen multimedialer Lernprogramme“ entwickelt. Eine Lerneinheit besteht in der Regel aus einer variablen Anzahl von Komponenten. Unter Komponente ist dabei u.a. eine einzelne Frage oder ein normaler Text zu verstehen. Unterschieden werden die Arten Präsentationskomponenten, wie Text, Bilder, Sounddateien, usw., und Fragekomponenten, wie z.B. eine Bildfrage. Dabei wurde in der Studienarbeit eine Schnittstelle entwickelt so, dass es möglich ist, eigene Komponenten zu erstellen. Dadurch kann ein Entwickler seine eigenen Komponenten gestalten, um seinen Kurs auf seine individuellen Wünsche

abzustimmen. Beim Entwickler handelt es sich aber in der Regel nicht um den Autor eines Kurses, da für die Entwicklung neuer Komponenten Kenntnisse in der Programmiersprache Java verlangt werden. Als Beispiel neuer Komponenten wurden im Rahmen dieser Diplomarbeit verschiedene Komponenten entwickelt, die für den Beispiel-Kurs benötigt werden (siehe Kapitel 4). Mit Hilfe dieser Komponenten erstellt der Autor eine einzelne Lerneinheit.

Wenn der Autor hierbei Fragekomponenten verwendet, benötigt diese Lerneinheit eine Bewertungskomponente, die für die Auswertung zuständig ist. Auch hier sollte es für den Autor möglich sein, je nach Leistung des Lernenden, einen individuellen Lernweg vorsehen zu können. Dies ähnelt den Verzweigungen der Lektionen in Abhängigkeit der Bewertung einer Lektion, nur das sich dies nun eine Ebene tiefer abspielt. Wie auch eine Ebene höher alle Wege zur Kursbewertung führen, führen auch auf der Ebene der Lerneinheiten alle Lernwege zur Lektion-Bewertung.

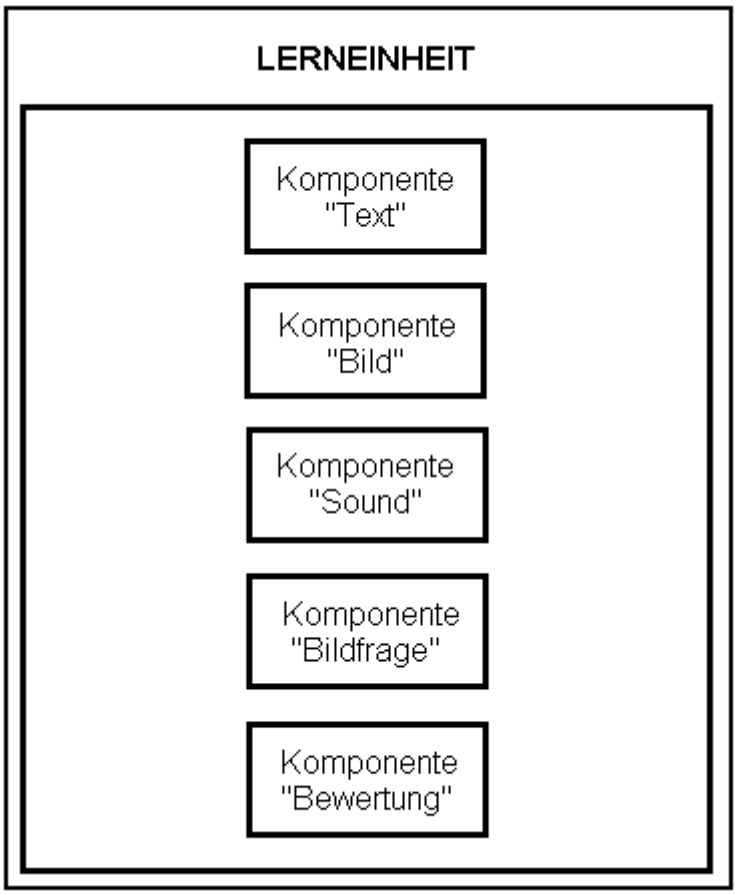

Abbildung 2: Schematische Darstellung einer Lerneinheit

In Abbildung 2 ist eine schematische Darstellung einer möglichen Lerneinheit abgebildet. Die Bewertungskomponente der Lerneinheit liefert, wie die anderen

Bewertungskomponenten auch, eine Rückmeldung über die Leistung des Lernenden. Sie nimmt dabei Bezug auch die erbrachten Leistungen des Lernenden innerhalb der jeweiligen Lerneinheit. Dies gilt sinngemäß auch für die Komponenten, die ihrerseits eine Rückmeldung geben. Dabei geben nur Fragekomponenten einen Leistungsnachweis, da dies für Präsentationskomponenten keinen Sinn macht.

2.2 Autorensteuerung

Ausgehend von der Struktur Kurs – Lektion – Lerneinheit – Komponenten wurde für das Autorensystem ein Konzeption entwickelt, mit welcher der Autor den Ablauf einer Lernsitzung festlegen kann. Oder mit anderen Worten gesprochen: Der Autor soll die Reihenfolge der Lektionen in Abhängigkeit der erbrachten Leistungen des Lernenden bestimmen können. Sinngemäß gilt dies auch für die Ebene der Lerneinheiten. Diese Festlegung des Ablaufs einer Lernsitzung wird auch als Autorensteuerung bezeichnet, da die Lerneinheiten bzw. Lektionen in einer vom Autor vorgesehenen Reihenfolge ablaufen, der Kurs also durch die vorgesehenen Wege des Autors gesteuert wird. Dabei ist die Autorensteuerung unter zwei Aspekten zu sehen. Zum einen wird die Autorensteuerung im Autorensystem durch den Autor festgelegt. Dieser Bereich wird im weiteren Verlauf auch Frontend genannt. Zum anderen wird die Autorensteuerung im Browser benutzt, um die entsprechenden Lerneinheiten anzuzeigen. Dieser Bereich wird im weiteren Backend genannt. In den nächsten Abschnitten wird auf die Funktionsweise des Frontends eingegangen und wegen des Backends sei auf das Kapitel 2.7 „Backend“ und das Kapitel 3 „Implementierung“ verwiesen.

Bei der Festlegung der Autorensteuerung wurden darüber hinaus software-ergonomische Kriterien beachtet, damit der Autor auf einfache Weise einen Kurs nach seinen Wünschen erstellen kann. Wie schon oben erklärt, gibt es bei einem Kurs verschiedene Ebenen, die durch die Autorensteuerung nachgebildet werden sollen. Bei der Entwicklung der Autorensteuerung wurde dabei darauf geachtet, dass die einzelnen Teile der Ebenen wieder verwendbar sind. Dies bedeutet, dass eine Lerneinheit, die in der einen Lektion verwendet wird, so gestaltet sein muss, dass sie ohne eine Veränderung auch in einer anderen Lektion verwendet werden kann. Aus diesem Grund darf die Lerneinheit keine Informationen darüber enthalten, mit welcher Lerneinheit in der Lektion fortgefahren werden soll, da sie sonst an eine

einzige Lektion gebunden wäre und für andere Lektionen nicht einsetzbar wäre. Informationen über den Verlauf der Lerneinheiten einer Lektion müssen demnach in der Lektion gespeichert werden und nicht in den einzelnen Lerneinheiten. Entsprechendes gilt auch hier für das Verhältnis Kurs zu Lektion. Nur in der Kursebene sind die Informationen über den Ablauf der Lektionen gespeichert, was eine Wiederverwendung von Lektionen in anderen Kursen ermöglicht. Die einzigen Informationen, die die Lektionsebene aus der Bewertungskomponente einer Lerneinheit benötigt, ist, wie viele Verzweigungen es von dieser Lerneinheit aus gibt. (Zur Implementierung siehe Kapitel 3). Die Verzweigungen stellen dabei die Leistungsbereiche dar, die der Autor für diese Lerneinheit vorgesehen hat. Der Autor unterteilt diese Leistungsbereiche mit Hilfe von Prozentangaben. So kann er z.B. vorsehen, dass bei bis zu fünfzig Prozent der erreichten Punkten ein anderen Weg eingeschlagen wird, als ab einfünfzig Prozent der Punkte. Die Einstellung ab null Prozent und ab einundfünfzig Prozent der Punkte ist hierbei die Standardeinstellung des Autorensystems für die Bewertungskomponenten aller Ebenen, die aber individuell veränderbar sind für die Bedürfnisse des Autors.
Abschließend sind noch die Startseiten der Kurs- bzw. Lektionebene erwähnt. Bei den Startseiten handelt es sich um spezielle Lerneinheiten, die jeweils beim Start der jeweiligen Ebene erscheinen. Der Autor besitzt hier die Möglichkeit den Lernenden mit allgemeinen Informationen über den Kurs bzw. die aktuelle Lektion zu versorgen, weswegen ihm hier der Einbau von Präsentationskomponenten ermöglicht ist. Da die Startseiten nur als Informationsseiten dienen soll, können hier noch keine Fragekomponenten eingebaut werden, was sie zu den anderen Lerneinheiten unterscheidet. Ein weiterer Unterschied besteht darin, dass die Startseiten die Informationsträger der Autorensteuerung sind, worauf im Kapitel 3 näher eingegangen wird.

2.3 Bedienung des Autorensystems und Festlegung des Ablauf einer Lernsitzung

Neben der Funktionalität der Autorensteuerung soll es dem Autor leicht möglich sein, einen Kurs zusammenzustellen und den Ablauf einer Lernsitzung festzulegen. Aus diesem Grund wird ihm jede einzelne Ebene in einem eigenen Fenster präsentiert. Nach dem Start des Programms gelangt der Autor in das Hauptfenster des

Autorensystems. Hier bekommt der Autor einen Überblick über alle Kurse, die er bereits entwickelt hat und es stehen ihm verschiedenartige Manipulationsbefehle zur Verfügung mit Hilfe derer er einen Kurs neu erstellen, kopieren, umbenennen, löschen, exportieren und importieren kann.

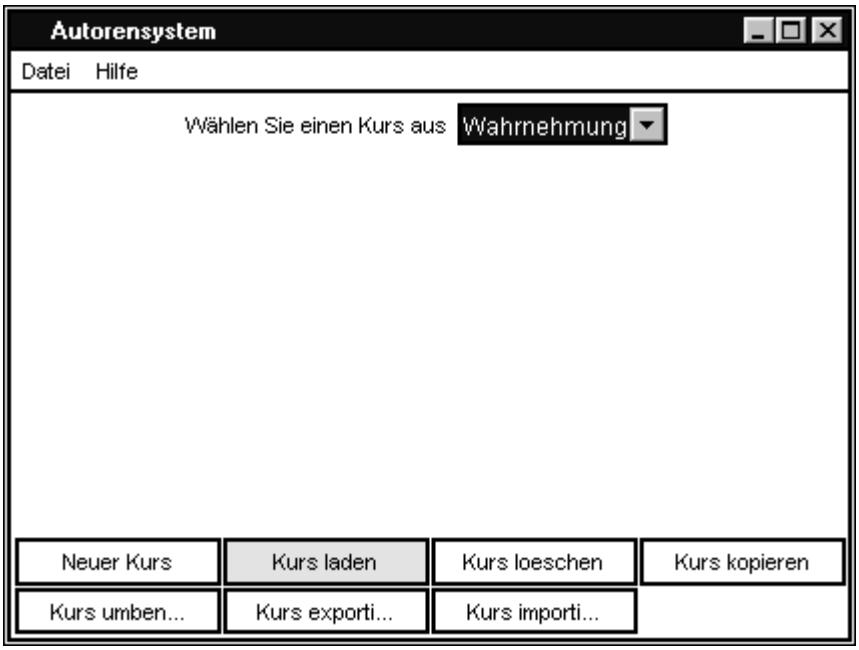

Abbildung 3: Hauptfenster des Autorensystems

Neben diesen Befehlen gibt es auch den Befehl Kurs laden, mit welchen der Autor einen ausgewählten Kurs laden kann, um diesen zu bearbeiten. Durch Auswahl der Laden-Schaltfläche gelangt der Autor auf die oberste Ebene des Kurses. In Abbildung 4 ist die Kursübersicht abgebildet, in der der Autor sich nun befindet. Unter Kursübersicht ist dabei die Übersicht eines einzelnen, ausgewählten Kurses zu verstehen und nicht die Übersicht über alle zur Verfügung stehenden Kurse.

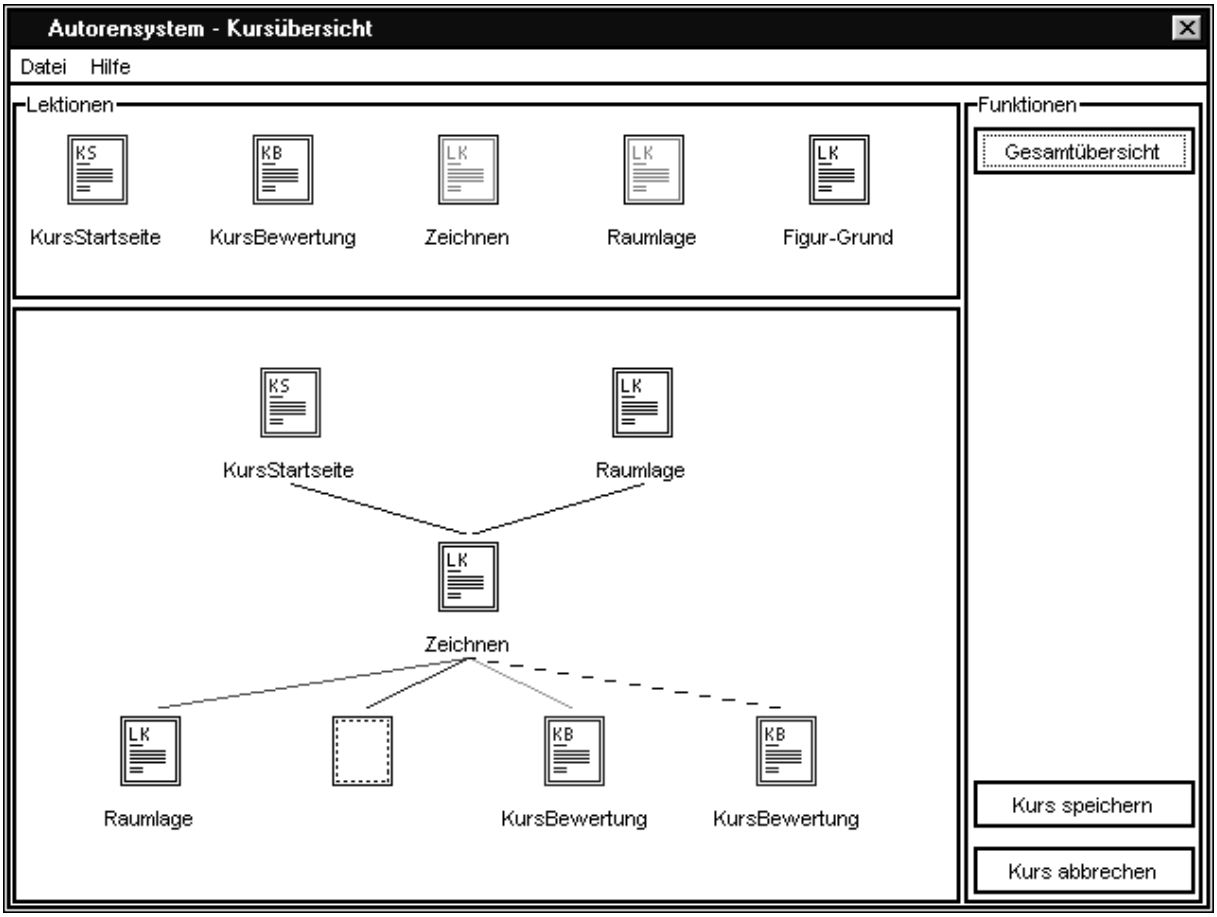

Abbildung 4: Fenster Kursübersicht

Die Kursübersicht unterteilt sich in drei Bereiche. Der erste Bereich links oben beinhaltet alle Lektionen, die in dem Kurs zur Verfügung stehen, was in der obigen Abbildung beispielhaft die Lektionen Zeichnen, Raumlage und Figur-Grund sind. Neben den Lektionen stehen auch die Kurs-Startseite (ganz links) und die Kurs-Bewertung (neben der Kurs-Startseite) zur Verfügung. Diese sind in einer anderen Farbe dargestellt, da diese den Anfang und das Ende eines Kurses darstellen. Im darunter liegenden Fenster wird dem Autor ein kleiner Ausschnitt der Verknüpfungen der Lektionen innerhalb des Kurses dargestellt. In der Mitte des Bereichs ist dabei die aktuell ausgewählte Lektion (in diesem Fall die Lektion „Zeichnen“) abgebildet. Darüber sind diejenigen Lektionen angegeben, von denen es einen Verweis bzw. eine Verzweigung auf die aktuelle Lektion gibt. Dies bedeutet in dem Beispiel, dass man von der Lektion Raumlage bzw. von der Kurs-Startseite auf die Lektion „Zeichnen“ gelangen kann. Aus diesem Grund werden diese Lektionen auch als Vorgänger-Lektionen bezeichnet. Hier sei kurz noch einmal erwähnt, dass es sich bei der Kurs-Startseite nicht um eine Lektion handelt, sondern um eine spezielle Lerneinheit, die einen Kurs einleitet. Gleiches gilt für die Kurs-Bewertung bei der es

sich um eine Komponente handelt und nicht um eine Lektion und die dazu dient eine Rückmeldung über die erbrachten Leistungen des Lernenden über den ganzen Kurs hinweg zu geben. Unterhalb der Lektion „Zeichnen“ sind nun diejenigen Verzweigungen angezeigt, die aus der Bewertungs-Komponente der Lektion (eine Ebene tiefer) ausgelesen wurden. Im obigen Beispiel bedeutet dies, dass die Bewertungskomponente der Lektion „Zeichnen“ drei Leistungsbereiche besitzt. Bei der vierten Verzweigung (gestrichelte Linie) handelt es sich um eine spezielle Verzweigung auf die im Kapitel 2.5 „Zyklus und Zyklusdurchbruch“ eingegangen wird. Auf die Bedeutung der Farben der einzelnen Verzweigungen sei auf das Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern“ verwiesen. Die Lektionen, die an den einzelnen Verzweigungen stehen, stellen demnach die nachfolgenden Lektionen dar, zu denen man gelangt, wenn man in den entsprechenden Leistungsbereich der Lektion „Zeichnen“ kommt. So stellt die linke Verzweigung immer denjenigen Leistungsbereich der aktuellen Lektion dar, der ab null Prozent der erzielbaren Punkte beginnt und bis zum Prozentbereich der nächsten Verzweigung geht. Die Prozentbereiche bzw. Leistungsbereiche werden hierbei, wie schon erwähnt, in der Bewertungskomponente der Lektionen eine Ebene tiefer festgelegt.

Auf der Kursübersicht muss es nun möglich sein, die einzelnen Verzweigungen der Lektionen mit anderen Lektionen zu verknüpfen und dadurch die Ablaufsteuerung, d.h. die Autorensteuerung, festzulegen. Dies geschieht mit Hilfe des Prinzips des Drag & Drop`s. Hierbei zieht der Autor die Lektionen aus der oberen Leiste auf offene Verzweigungen im Fenster der Ablaufsteuerung. Offene Verzweigungen werden durch ein spezielles Icon mit einem gestrichelten Rechteck und leerem Inhalt dargestellt. Außerdem ist das Feld unterhalb des Icons, in dem normalerweise der Name der Lektion angegeben ist, leer, da für die offene Verzweigung ja bisher noch keine nachfolgende Lektion ausgewählt wurde. So kann man die offene Verknüpfung des zweiten Leistungsbereiches (siehe Abbildung 4) mit der Lektion „Figur-Grund“ verbinden, indem man mit der linken Maustaste auf die Lektion in der oberen Leiste drückt, diese festhält und das in eine Lerneinheit veränderte Symbol des Mauszeigers auf die offene Stelle zieht. Im folgenden verändert sich das Symbol in das Symbol einer Lektion ab und unterhalb des Symbols erscheint der Name der Lektion. In diesem Fall wäre es „Figur-Grund“. Durch das Drag & Drop ist es also möglich die nachfolgenden Lektionen der aktuellen Lektion zu setzen und stellt damit

die Basis der Festlegung des Ablaufs einer Lernsitzung dar. Bleibt nur noch die Frage bestehen, wie man die gewünschte Lektion zur aktuellen Lektion macht.

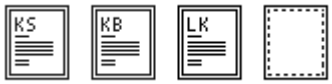

Abbildung 5: Symbole der Kursübersicht: Kurs-Startseite, Kurs-Bewertung, Lektion, offene Verknüpfung

Hierfür besitzt der Autor zwei Möglichkeit. Die erste Möglichkeit besteht darin, dass er auf einer der Vorgänger- oder Nachfolger-Lektionen einen Doppelklick ausführt. Dadurch wird automatisch diese Lektion zur aktuellen Lektion und der Autor bekommt die Vorgänger- bzw. Nachfolger-Lektionen der neuen, aktuellen Lektion zu sehen. Beispielhaft wurde auf die Nachfolger-Lektion „Raumlage“ geklickt (siehe Abbildung 6).

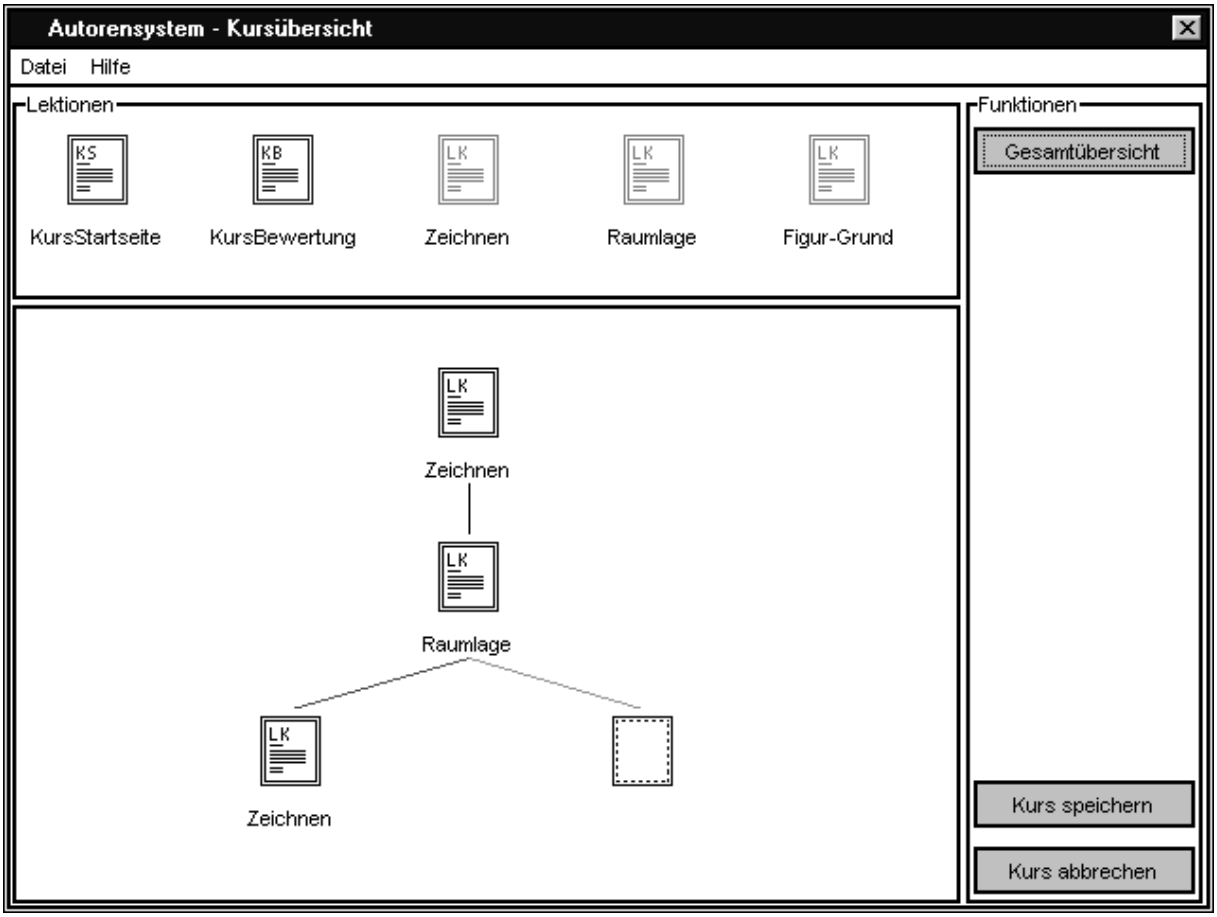

Abbildung 6: Aktuelle Lektion „Raumlage“

Wie zu sehen ist, besitzt diese neue, aktuelle Lektion, die Lektion „Zeichnen“ sowohl als Vorgänger- als auch als Nachfolger-Lektion, was bedeutet, dass der Lernende aufgrund seiner Leistungen zur Lektion „Zeichnen“ zurückkehren kann. Auf der Lektion „Zeichnen“ ist demnach ein Zyklus vorhanden. Zu den Zyklen sei auf das Kapitel 2.5 „Zyklus und Zyklusdurchbruch“ verwiesen. An dieser Stelle wird mit der weiteren Funktionsweise der Festlegung der Autorensteuerung fortgefahren.
Durch das Klicken auf Vorgänger- bzw. Nachfolger-Lektionen ist es dem Autor also möglich sich durch die Verknüpfungsstruktur seines Kurses zu bewegen. Dies ist zum Beispiel beim Aufbau eines neuen Kurses sehr hilfreich, wenn der Autor nach und nach die Lernwege erstellt und somit eine Lektion nach der anderen Lektion mit den richtigen Verknüpfungen versehen will. Im obigen Beispiel könnte dies bedeuten, dass der Autor zuerst alle Verknüpfung der Lektion „Zeichnen“ füllt und danach die Verknüpfungen der Lektionen „Raumlage“ und „Figur-Grund“. Wenn das Lernprogramm jedoch aus vielen Lektionen besteht oder wenn der Autor eine bestehenden Kurs bearbeiten möchte und sich erst durch den ganzen Verzweigungsbaum durchklicken muss, bis er zur gewünschten Lektion kommt, so kann dieser Weg recht mühsam sein. Außerdem muss sich der Autor genau in seinem Verzweigungsbaum auskennen, damit er auch den richtigen Weg zu der gewünschten Lektion findet. Hierfür steht dem Autor noch ein zweiter Weg offen:

Die Gesamtübersicht:

Die Gesamtübersicht ist ein hilfreiches Werkzeug für den Autor, um einen Gesamtüberblick über die ganze Verzweigungsstruktur des Kurses zu bekommen. Hierbei ist die Gesamtübersicht als eine Art Baum dargestellt. Ganz oben steht dabei immer die Kurs-Startseite, da sie der Beginn eines jeden Kurses ist. Von der Kurs-Startseite führt dann die erste Verzweigung zur ersten Lektion des Kurses, was im behandelten Beispiel die Lektion „Zeichnen“ ist. Anschließend werden die Verzweigungen der Lektion „Zeichnen“ und der darauffolgenden Lektionen angezeigt. So hat der Autor einen Überblick über alle Lernwege des Kurses und kann jeden einzelnen dieser Lernwege verfolgen. Die meisten Lernwege enden dabei mit der Kurs-Bewertung. Jedoch sieht man auf der Abbildung 7, dass ein Lernweg mit einer Lektion endet, was der obigen Behauptung zuwider laufen würde, dass jeder Lernweg mit der Kurs-Bewertung enden muss. So endet bei dem Lernweg

ganz links (Kurs-Startseite – Zeichnen – Raumlage – Zeichnen) dieser nicht mit der Kurs-Bewertung sondern mit der Lektion „Zeichnen“. Wenn man sich den Fall aber genauer betrachtet, handelt es sich hierbei wiederum um den schon erwähnten Zyklus auf der Lektion „Zeichnen“. Wenn der Lernende diesen Weg beschreiten sollte, so kommt er also nicht in einen Deadlock des Kurses, sondern er befindet sich sozusagen wieder auf der Lektion unterhalb der Kurs-Startseite. Und von dieser Lektion „Zeichnen“ gibt es mehrere Wege zur Kurs-Bewertung.

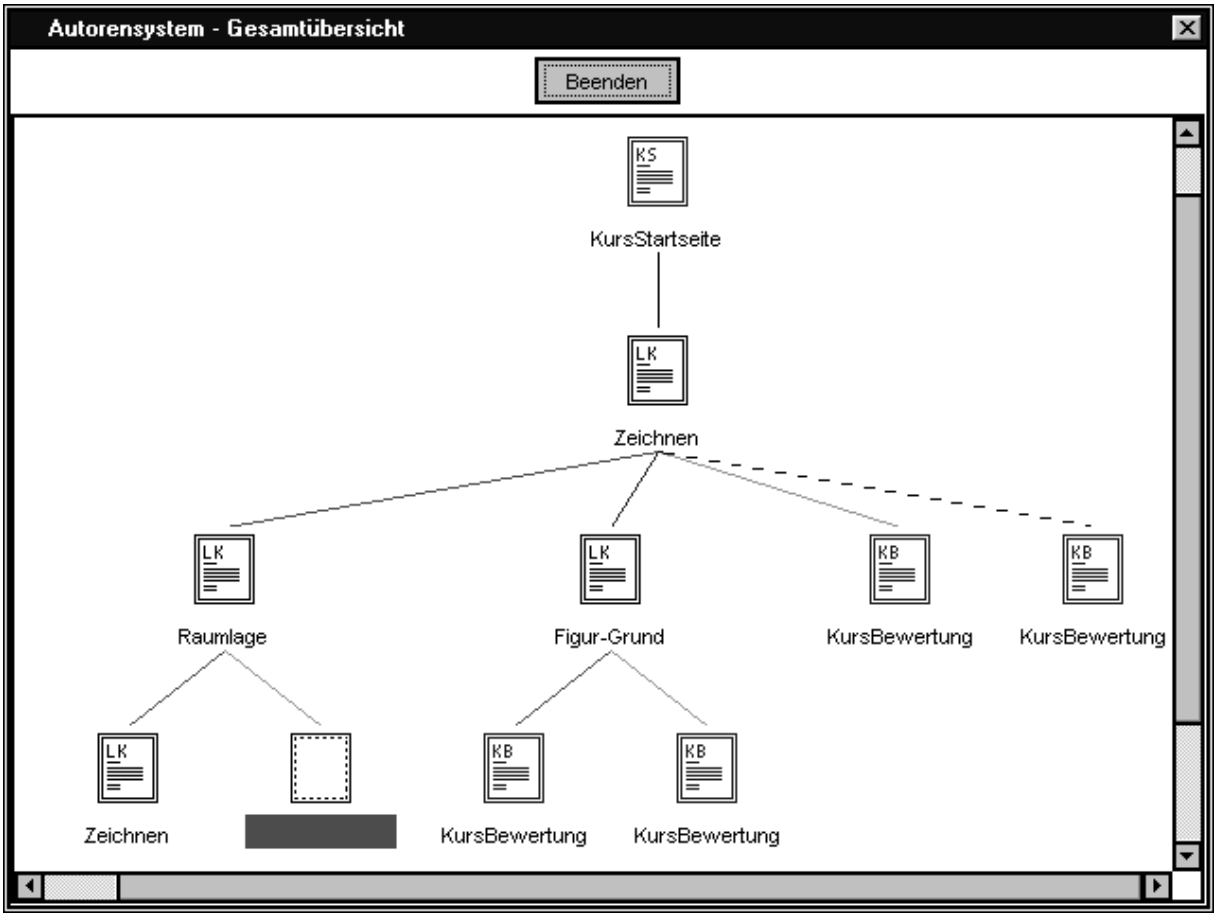

Abbildung 7: Gesamtübersicht

Auf diese Weise werden alle Zyklen in der Gesamtübersicht dargestellt, da eine andere Darstellung zu einer verwirrenden Anzahl von Verbindungslinien führen würde. Dem Autor muss nur klar sein, dass es sich bei Lektionen mit dem gleichen Namen immer um dieselbe Lektion handelt und dass bei mehrfachen Auftreten derselben Lektion die Verzweigungen dieser Lektion nur einmal abgebildet sind, um eine klaren Überblick zu behalten. Die Abbildung der Verzweigungen derselben Lektion befindet sich dabei immer beim ersten Auftreten der Lektion im

Verzweigungsbaum. Optisch bedeutet dies, dass es sich um die Lektion handelt, die am weitesten oben ist.
Die Gesamtübersicht kann bei einer Vielzahl von Lektionen ziemlich groß werden, weswegen der Autor mit Hilfe von Scrollbars sich den gewünschten Abschnitt des Kurses in das Anzeigefenster bewegen kann.

Was hat dies nun mit dem zweiten Weg der Navigation zu tun? Der Autor besitzt in der Gesamtübersicht die Möglichkeit auf irgendeine der angezeigten Lektionen einen Doppelklick zu machen. Dadurch schließt sich das Fenster der Gesamtübersicht und es erscheint wieder das Fenster der Kursübersicht mit dem Unterschied, dass die vom Autor angeklickte Lektion nun die aktuelle Lektion ist.
Dem Leser dürfte aufgefallen sein, dass unterhalb des Symbols der offenen Verzweigung ein Balken eingeblendet ist. Dieser rote Balken soll den Autor signalisieren, dass er hier noch eine Verknüpfung zu schließen hat (siehe Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern“). Sollte der Autor einen Doppelklick auf einer „offenen“ Lektion durchführen, so wird nicht diese „offene“ Lektion zur aktuellen Lektion, was natürlich keinen Sinn macht, sondern die darüber liegende Lektion. Im Beispiel wäre es die Lektion „Zeichnen“. Der letzte Weg die Gesamtübersicht zu verlassen, ist das Drücken der Schaltfläche „Beenden“. Da hierdurch keine Lektion ausgewählt wurde, erscheint wieder das Fenster der Kursübersicht mit der zuvor aktuellen Lektion. Mit Hilfe dieser beiden Wege sollte es dem Autor einfach möglich sein, einen Ablauf der Lernsitzung festzulegen und damit die Autorensteuerung zu erstellen. Auf die Gesamtübersicht wird in den nachfolgenden Kapitel nochmals kurz eingegangen, da diese noch einen weiteren Aspekt in einem weiteren Zusammenhang besitzt.

Auf der Ebene der Kursübersicht besitzt der Autor neben der Zusammenstellung der Lektionen zu einem Kurs, mehrere Möglichkeiten die einzelnen Lektionen zu manipulieren. So stehen ihm im Menü „Datei“ die zwei Menübefehle „Neue Lektion“ und „Lektion importieren“ zur Verfügung. Mit Hilfe des Befehls „Neue Lektion“ kann der Autor eine neue Lektion erstellen. Hierzu wird er erst aufgefordert, einen Namen der neuen Lektion einzugeben. Dabei darf kein Name einer Lektion verwendet werden, die bereits existiert. Anschließend öffnet sich das Fenster der Lektionübersicht, worauf in den nächsten Abschnitten eingegangen wird. Der zweite

Befehl „Lektion importieren“ ermöglicht es dem Autor eine ganze Lektion von einem anderen Speicherort in den Kurs zu importieren. Dies entspricht der bereits erwähnten Wiederverwendung von Lektionen. Wie bei dem Befehl „Neue Lektion“ erscheint die Lektion bei erfolgreichem Importieren in der oberen Anzeigeleiste der Lektionen und der Autor kann nun diese in seinen Verzweigungsbaum des Kurses einfügen. Zusätzlich stehen dem Autor weitere Befehle zur Verfügung, die über ein Popup-Menü zu erreichen sind. Hierzu muss der Autor nur die rechte Maustaste auf einem der Lektion-Symbole in der oberen Anzeigeleiste bzw. im Fenster links unten klicken. Dabei erscheinen nicht immer alle Befehle. Hierauf sei auf das Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern“ verwiesen. Die im allgemeinen zur Verfügung stehenden Befehle sind „Lektion löschen“, „Lektion kopieren“, „Lektion exportieren“ und „Lektion umbenennen“. Auf diese Befehle wird aber hier nicht näher eingegangen, da die Bedeutung der Befehle nahe liegt. Der Befehl „Zyklusduchbruch“, welcher in manchen Popup-Menüs des Fensters links unten erscheint, jedoch nicht in der oberen Anzeigenleiste der Lektionen, wird im Kapitel 2.5 „Zyklus und Zyklusdurchbruch“ erklärt. Neben dem Befehl „Zyklusdurchbruch“ gibt es zwei weitere Befehle, die nur im linken unteren Fenster erscheinen. Dies sind die Befehle „Lösche Link“ und „Lösche Teilbaum“.

Bisher wurde nur erklärt, wie der Autor eine Verzweigung setzen kann. Dabei kann er natürlich einen Fehler machen und um diesen rückgängig zu machen, stehen ihm diese zwei Befehle zur Verfügung. Dabei muss er sich des Unterschieds zwischen den beiden Befehlen klar sein. Mit Hilfe des Befehls „Lösche Teilbaum“ wird der ganze Teilbaum gelöscht, der die ausgewählte Lektion als Wurzel-Lektion besitzt. Dies bedeutet beispielhaft folgendes. Der Autor hat als aktuelle Lektion die Lektion „Zeichnen“ ausgewählt und geht auf die Nachfolger-Lektion „Figur-Grund“ um hier den ganzen Teilbaum zu löschen. Dies bewirkt, dass die Lektion „Figur-Grund“ mitsamt ihren Verzweigungen zur Kurs-Bewertung aus der Verzweigungsstruktur gelöscht wird (siehe Abbildung 7). Dabei wurde die Lektion nicht physikalisch gelöscht und steht weiterhin in der oberen Anzeigeleiste zu einem späteren Einbau zur Verfügung.

Beim Befehl „Lösche Link“ wird hingegen nicht der ganze Teilbaum der Verzweigungsstruktur gelöscht, sondern es wird nur die Verzweigung zwischen der aktuellen Lektion und der ausgewählten Nachfolger-Lektion geöffnet. Sollte die bisherige Nachfolger-Lektion noch eine andere Vorgänger-Lektion als die aktuelle

Lektion besitzen, bzw. sollte die Nachfolger-Lektion noch an einem anderen Zweig der aktuellen Lektion hängen, so bleibt diese mitsamt ihrer nachfolgenden Lektionen im Verzweigungsbaum existent. Lediglich die einzelne Verzweigung ist gekappt. Handelte es sich jedoch um die einzigste Verbindung, bzw. war die aktuelle Lektion die einzigste Vorgänger-Lektion, so wird automatisch ein „Lösche Teilbaum“ durchgeführt. Damit sind nun alle Befehle der Festlegung der Autorensteuerung auf der Ebene des Kurs erklärt. Jedoch besteht der Kurs nicht nur aus Lektionen, sondern die Lektionen selbst bestehen noch aus Lerneinheiten und für diese Lerneinheiten innerhalb einer Lektion müssen ebenfalls noch die Lernwege festgesetzt werden. Dazu muss der Autor auf die nächst tiefere Ebene, die Lektionübersicht gelangen. Dazu stehen ihm wiederum zwei Wege zur Verfügung, die einen kleinen aber feinen Unterschied besitzen. Zum einen kann der Autor einen Doppelklick auf eine der Lektionen in der oberen Anzeigeleiste durchführen, oder er kann einen Doppelklick auf der aktuellen Lektion machen. Der Effekt ist derselbe und es öffnet sich ähnlich wie bei dem Befehl „Neue Lektion“ die Lektionübersicht mit der ausgewählten Lektion. Der Unterschied besteht darin, dass die aktuelle Lektion bereits in die Verzweigungsstruktur des Kurses eingebaut ist und dies bei einer Lektion in der Anzeigenleiste nicht unbedingt der Fall sein muss. Jedenfalls muss bei eingebundenen Lektionen beachtet werden, dass der Autor durch Veränderung der Lektion-Bewertung die Verzweigungsstruktur auf Ebene der Kursübersicht verändern kann. Sollte er z.B. einen weiteren Auswertungsbereich in der Lektion-Bewertung einführen, so erscheint dieser nach Bearbeiten der Lektion in der Kursübersicht als weitere offene Verzweigung der Lektion. Ähnliches gilt für das Löschen von Bewertungsbereichen, wodurch der Autor die Verzweigung öffnet, die an diesem Bewertungsbereich hängt. Sollte der Autor letztlich die Bewertungskomponente ganz löschen und eine neue einfügen, so wurden alle Verzweigungen, die an der Lektion hingen, geöffnet, da das Programm davon ausgeht, dass der Autor mit einer neuen Lektion-Bewertung auch eine neue Verzweigungsstruktur wünscht.

Bei Lektionen, die nicht in die bisherige Verzweigungsstruktur eingebunden sind, müssen diese Manipulationen an den Verzweigungen nicht beachtet werden, und es öffnet sich lediglich die Lektionübersicht mit der ausgewählten Lektion zur Bearbeitung.

Die Lektionübersicht bietet dem Autor nun keine Überraschung mehr, da sie prinzipiell genau gleich aufgebaut ist, wie die Kursübersicht, weswegen hier auf eine Abbildung verzichtet wird. Lediglich die Symbole der Ebene haben sich geändert, um den Autor zu zeigen, dass er sich auf einer neuen Ebene befindet. (siehe Abbildung 8).

Abbildung 8: Symbole der Lektionübersicht: Lektion-Startseite, Lektion-Bewertung, Lerneinheit, offene Verknüpfung.

Die Unterschiede bei der Lektionübersicht bestehen darin, dass der Autor hier nun auf dieselbe Weise wie eine Ebene höher, die Verknüpfungen der Lerneinheiten untereinander setzen kann und damit die Lernwege der ausgewählten Lektion erstellen kann. Dabei gelten sinngemäß alle Regeln aus der oberen Ebene. So muss zum Beispiel jeder Lernweg zur Lektion-Bewertung führen. Auch ist es hier möglich nun die Lektion-Bewertung zu bearbeiten, die im vorigen Abschnitt schon erwähnt wurde, wie auch, dass deren Manipulation zu Veränderungen in der Verzweigungsstruktur der Kursebene führt. Die Lektion-Bewertung, sowie die Lektion-Startseite werden in der Anzeigeleiste der Lerneinheiten ganz links angezeigt und können durch einen Doppelklick bearbeitet werden. Wie auch die anderen Lerneinheiten bei einem Doppelklick ein Fenster öffnen, in der die jeweils ausgewählte Lerneinheit mitsamt ihren Komponenten dargestellt wird und hier auch bearbeitet werden kann. Dieses Fenster mit der Übersicht über alle Komponenten wurde in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt. Die Funktionsweise sei deswegen nur kurz angerissen. Mit Hilfe der Schaltflächen „Präsentation“, „Frage“ bzw. „Bewertung“ ist es dem Autor möglich Komponenten dieser Art einzubauen, die von Entwicklern für das Programm zur Verfügung gestellt wurden. Wie auf den vorigen Ebenen besitzt auch diese Ebene die Möglichkeit, die einzelnen Komponenten mit Hilfe eines Editors zu manipulieren, worauf aber auf das Kapitel 4 verwiesen sei, in dem ein paar neue Komponenten vorgestellt werden. Die Lerneinheit kann ebenfalls, ähnlich zur Lektion, bzw. zum Kurs eine Bewertungskomponente besitzen, die für die Bewertung der ausgewählten Lerneinheit zuständig ist. Auch hier gilt, dass Manipulationen an dieser

Bewertungskomponente Auswirkungen auf die Verzweigungsstruktur eine Ebene höher (Lektionübersicht) hat. Als Besonderheit kommt hier noch hinzu, dass eine Lerneinheit, wenn sie nur aus Präsentationskomponenten besteht keine Bewertungskomponente besitzt und dies auf der Ebene der Lektionübersicht durch eine einzelne schwarze Linie der entsprechenden Lerneinheit gekennzeichnet wird. Diese hat die Bedeutung, dass es nur einen Weg der Fortsetzung von dieser Lerneinheit aus gibt. Damit sollte dem Leser nun klar sein, wie es ihm möglich ist, einen kompletten Kurs aus einzelnen Lerneinheiten zu erstellen. Dieser muss natürlich noch gespeichert werden, wozu dem Autor auf den einzelnen Ebene eine entsprechende Speichern-Schaltfläche bzw. ein entsprechender Menüpunkt im Menü „Datei“ zur Verfügung steht. Sollte der Autor seine Arbeit nicht speichern wollen, so besitzt er die Möglichkeit die Arbeiten auf jeder Ebene abzubrechen, was dazu führt das keine der Änderungen gespeichert werden. Für die Befehle „Neuer Kurs“, „Neue Lektion“ und „Neue Lerneinheit“ bedeutet ein Abbrechen, dass keine dieser Struktur angelegt wurde, und das Programm den gleichen Zustand besitzt, als wäre dieser Befehl nicht ausgeführt worden. Bei dem hier verwendeten Beispiel handelt es sich um einen Beispiel-Kurs, der die Festlegung der Autorensteuerung veranschaulichen soll und nicht um den Kurs, der in Kapitel 4 erstellt wurde.

2.4 Bewertungskomponenten

Im vorigen Abschnitt wurde schon mehrfach auf die Bedeutung der Bewertungskomponenten innerhalb der Autorensteuerung eingegangen. Hier sei noch einmal die genaue Funktionsweise dieser wichtigen Komponenten dargestellt. Eine Bewertungskomponente ist der Abschluss jeder Ebene. So wird die Kursebene mit einer Kurs-Bewertung abgeschlossen, entsprechendes gilt für die Lektionebene und eine Lerneinheit wird mit einer Lerneinheiten-Bewertung abgeschlossen. Bei der letzten Bewertungskomponente gibt es die Besonderheit, dass eine Lerneinheit ohne Fragekomponenten keine Bewertungskomponente besitzen muss. Auf den anderen Ebenen ist dies aber zwingend erforderlich, und der Autor kann die jeweilige Ebene nicht abspeichern, bevor eine Bewertungskomponente eingefügt ist (siehe Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern). Zur Bearbeitung einer Bewertungskomponente muss der Autor lediglich auf das entsprechende Symbol in der Anzeigenleiste klicken und den Editor über ein Popup-Menü aktivieren. In diesem

Editor, der in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt wurde, besitzt der Autor die Möglichkeit verschiedene Bewertungsbereiche einzugeben. Hierfür muss er lediglich die Prozentzahl angeben, ab wie viel Prozent der maximal erreichbaren Punkte der Lernende in diesen Bereich kommt. Für jeden Bereich besteht die Möglichkeit, dass der Autor dem Lernenden eine individuelle Rückmeldung gibt, wie er abgeschnitten hat. Der Autor muss dabei nur beachten, dass er die Bereiche in aufsteigender Reihenfolge angibt. Dies wird beim Abspeichern geprüft und gegebenenfalls muss der Autor die Bereiche anpassen, wenn er eine Falscheingabe gemacht hat. Ebenso kann er nur Zahlen von null bis hundert eingeben, da der Lernende maximal hundert Prozent der Punkte erreichen kann. Zur Berechnung der Punkte sei auf das Kapitel 2.7 „Backend“ verwiesen. In diesem Kapitel steht auch, wie die Rückmeldungen im Browser angezeigt werden. Im allgemeinen sind die Bewertungskomponenten der verschiedenen Ebenen gleich aufgebaut, besitzen aber kleine Unterschiede. So ist es in der Bewertungskomponente einer Lerneinheit möglich, eine nachfragende Lerneinheit zu aktivieren, wenn der Lernende in den entsprechenden Bereich kommt. Das Konzept der nachfragenden Lerneinheiten wurde in der Diplomarbeit Nr. 2076 [Conradt] „Entwicklung von Fragekomponenten zur Lernkontrolle von e-Learning-Kursen am Beispiel eines Kurses zum Erlernen der Blindenschrift“ entwickelt und ist dort nachzulesen. Die Bewertungskomponente der Lektionebene hat denselben Aufbau, wie die Bewertungskomponente einer Lerneinheit, bis auf den Unterschied, dass hier keine nachfragenden Lerneinheiten aktiviert werden können, da es eine derartige Struktur auf der Lektionebene nicht gibt. Beide Bewertungskomponenten haben, wie im vorigen Kapitel schon erwähnt, Auswirkungen auf die nächst höhere Ebene. So werden die Verzweigungsmöglichkeiten einer Lerneinheit durch die Anzahl der Bewertungsbereiche der Bewertungskomponente der Lerneinheit bestimmt und entsprechendes gilt für die Lektion-Bewertung und die Kursebene. Die Bereiche von null Prozent ab werden dabei in der gleichen Reihenfolge von links nach rechts in der Lektionübersicht bzw. Kursübersicht angezeigt.

Die Kurs-Bewertung besitzt gegenüber den anderen beiden Bewertungskomponenten einen kleinen Unterschied. Dieser besteht darin, dass die Kurs-Bewertung keine Auswirkung auf die nächst höhere Ebene besitzt, da sich die Kurs-Bewertung schon auf der höchsten Ebene befindet. Die Kurs-Bewertung ist ausschließlich dafür da, um eine Rückmeldung an den Lernenden zu geben

entsprechend der von ihm erreichten Punkten. Alle drei Bewertungskomponenten wurden dabei so entwickelt, dass sie die Schnittstelle für Komponenten erweitern und entsprechen damit dem Prinzip des offenen Autorensystem, was in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt wurde. Im nächsten Kapitel wird nun die letzte Verzweigungsmöglichkeit vorgestellt, die unabhängig ist von den Bewertungskomponenten.

2.5 Zyklus und Zyklusdurchbruch

Mehrfach wurde schon auf dieses Kapitel verwiesen, wenn es darum ging, dass ein Zyklus auf einer Lektion entstanden ist. Was hat es nun auf sich mit diesen Zyklen. Zuerst sein noch mal geklärt, was genau ein Zyklus ist. Ein Weg ist ein Zyklus, wenn der Anfangspunkt eines Weges gleich dem Endpunkt eines Weges ist. D.h. der Weg führt zurück zum Ursprung. Im obigen Beispiel (siehe Abbildung 7) ist dies beispielhaft der Weg Zeichnen – Raumlage – Zeichnen. Der kleinste mögliche Zyklus auf einer Lektion, ist der Zyklus auf sich selbst, d.h. die Lektion hat als Nachfolger-Lektion sich selbst. Daneben sind Zyklen verschiedener Größe, d.h. mit einer variablen Anzahl von Lektionen zwischen Anfang und Ende des Weges möglich. Dies gilt wie immer sinngemäß auch für die Lerneinheiten. An sich ist dies aber nichts besonderes, wenn der Autor vorsehen will, dass ein Lernender eine oder mehrere Lektionen wiederholen soll, wenn dieser eine gewisse Punktzahl erreicht hat. Dies kann zum Beispiel der Fall sein, wenn der Lernende eine Lektion falsch beantwortet hat, der Autor aber der Meinung ist, dass der Lernende erst fortfahren kann, wenn er diese erste Lektion richtig beantwortet hat. Jedoch kann es dann zu einem Problem werden, wenn der Lernende in diesem Zyklus stecken bleibt, weil er immer wieder dieselben Antworten gibt. Hierfür besitzt der Autor einen Möglichkeit, diesen Zyklus zu durchbrechen, was die gestrichelte Linie in der Abbildung 4 von der aktuellen Lektion „Zeichnen“ zur Kurs-Bewertung erklärt. Bei dieser Linie handelt es sich um eine weitere Verzweigungsart von Lektionen untereinander, dem Zyklusdurchbruch. Da eine Lektion nicht wissen kann, ob sie sich in einem Zyklus befindet, kann diese Verzweigungsart nicht in der Lektionen-Bewertung gesetzt werden, da ansonsten die Lektionen Informationen von außen besitzen würden, was dem Konzept der Wiederverwendung entgegenlaufen würde. Aus diesem Grund kann ein Zyklusdurchbruch nur in der Kursebene gesetzt werden. Jedoch nicht auf

allen Lektionen, sondern nur auf den Lektionen, die sich innerhalb eines Zyklus befinden. Dabei muss sich der Autor nun keine Gedanken machen, welche Lektionen sich in einem Zyklus befinden, da das Autorensystem automatisch feststellt, ob es sich um eine Lektion in einem Zyklus handelt. Diese Überprüfung geschieht immer dann, wenn eine neue Lektion in die Verzweigungsstruktur eingebaut wird, oder eine Lektion daraus entfernt wird. In diesen Fällen kann sich nämlich einer oder mehrere Zyklen schließen bzw. öffnen. Der Autor weiß, es handelt sich um eine Lektion innerhalb eines Zyklus, wenn der Befehl „Zyklusdurchbruch“ im Popup-Menü der aktuellen Lektion oder einer der Nachfolger-Lektionen erscheint. Führt der Autor nun diesen Befehl aus, erscheint ein kleines Eingabefenster, indem er zwei Dinge angeben kann. Zum einen verlangt das Programm eine Zahl, die angibt, wie oft die Lektion maximal besucht werden darf. Die vorgegebene Einstellung für die maximale Anzahl von Besuchen ist dabei unendlich. Und zum anderen kann der Autor eine kleine Rückmeldung für den Lernenden geben, wenn der Fall eintritt, dass der Lernende die Lektion öfters besucht hat, als erlaubt. In diesem Fall wird der Lernende auf diese Lektion weitergeleitet, die der Autor unter der Verzweigung Zyklusdurchbruch angegeben hat. Im obigen Beispiel ist dies die Kurs-Bewertung. Auf diese Weise kann der Autor einen Lernenden bestimmte Lektionen ein paar mal wiederholen lassen, bevor er sich erbarmt und den Lernenden zu einer anderen, vielleicht leichteren Lektion oder zur Kursbewertung leitet. Da aber in diesem Autorensystem der freie Fall programmiert wurde, d.h. dem Autor sind alle Möglichkeit für die Gestaltung seines Kurses offen, kann er diesen Zyklusdurchbruch setzen, er muss ihn aber nicht setzen und wird deswegen nicht aufgefordert einen derartigen Zyklusdurchbruch bei jedem Zyklus einzugeben.

Jedoch besteht eine weitere Gefahr bei der Erstellung derartiger Zyklusdurchbrüche. Hierzu sei auf die Abbildung 9 verwiesen. Man sieht hier einen Ausschnitt einer Verzweigungsstruktur von Lerneinheiten einer Lektion. Interessant ist hierbei der Zyklus „aufgabe1 – Praesentation1 – Praesentation2 – aufgabe2“. Die Lerneinheit Praesentation1 und Praesentation2, bei denen es sich im Beispiel um Lerneinheiten ausschließlich mit Präsentationskomponenten handelt, besitzen beide eine Zyklusdurchbruch, der auf die jeweils andere Lerneinheit verweist. Durch eine ungeschickte Wahl der maximalen Besuche dieser Lerneinheiten, kann es passieren, dass beide Lerneinheiten ihre maximale Besuchsanzahl erreicht haben und die Lerneinheit in ihrer Verzweigung Zyklusdurchbruch als nächstes von der

Autorensteuerung angezeigt werden soll. Da die maximale Anzahl der Besuche dieser Lerneinheit aber auch überschritten ist, würde der Lernende in einen dynamischen Zyklus geraten, woraus er nicht mehr herauskommt. Aus diesem Grund ist ein derartiger dynamischer Zyklus nicht erlaubt und wird bei der Überprüfung der Verzweigungsstruktur einer Lektion beim Speichern der Lektion erkannt.

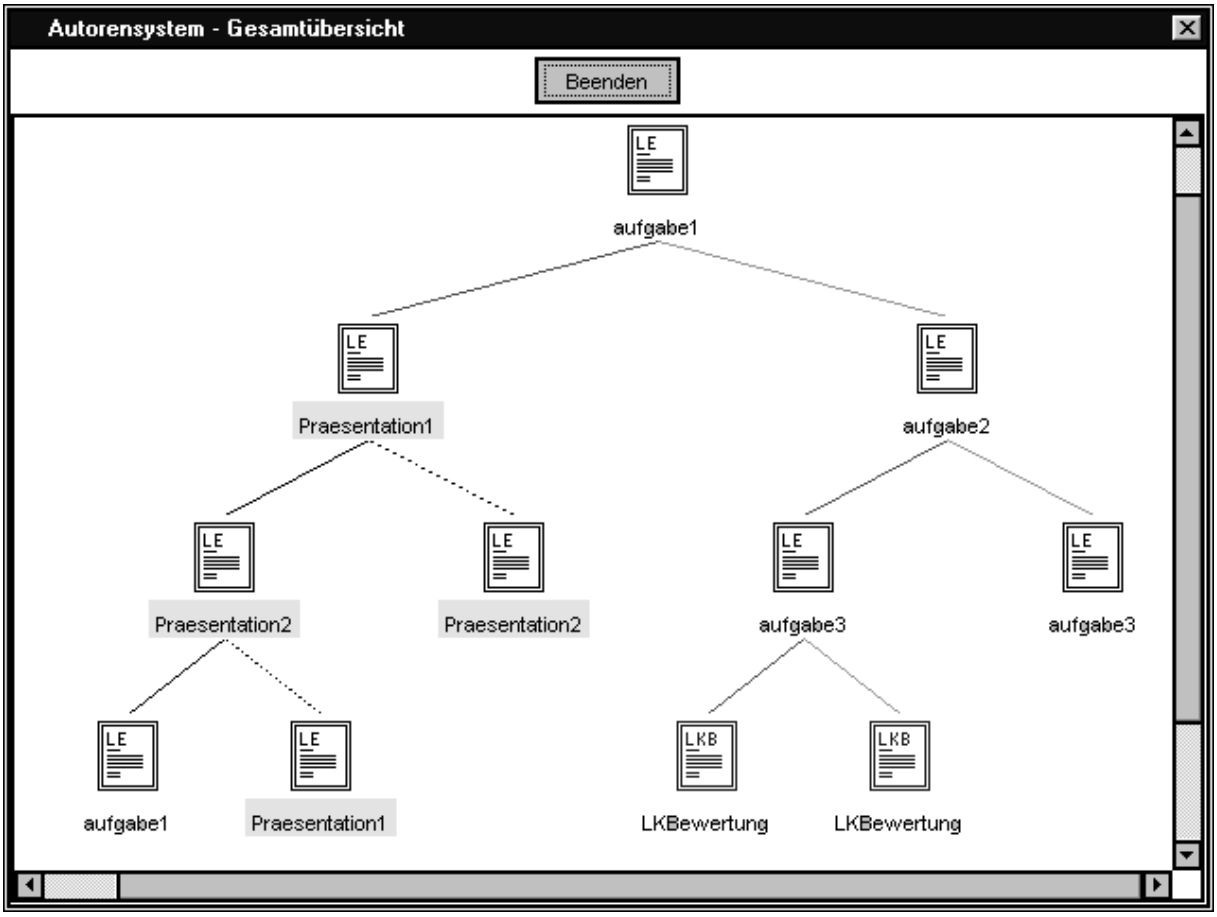

Abbildung 9: Darstellung eines dynamischen Zyklus

Und der Autor wird darauf hingewiesen, dass er eine derartige Struktur aufbrechen muss, bevor die Lektion abgespeichert wird. Zu den weiteren Überprüfung der Verzweigungsstruktur der Lektion siehe nächstes Kapitel. Damit sind nun alle Möglichkeiten aufgezählt, wie der Autor eine Autorensteuerung nach seinem Willen festlegen kann.

Das Erzeugen von Zyklen ist hierbei nur unterhalb von Lerneinheiten bzw. Lektionen möglich und das Durchbrechen dieser Zyklen dementsprechend nur auf den Ebenen der Lektionübersicht bzw. der Kursübersicht. Innerhalb einer Lerneinheit gibt es keine Zyklen, da die einzelnen Komponenten linear nacheinander angezeigt werden.

2.6 Strategie zur Vermeidung von Bedienfehlern

Bei der Arbeit des Autors im Autorensystem können dabei verschiedene Fehler auftreten, die durch Unachtsamkeit seitens des Autors hervorgerufen werden können. Diese Fehler sollten natürlich vom Autorensystem abgefangen werden und es sollte versucht werden, diese Fehler durch Strategien von vorneherein zu vermeiden. Dazu werden in dem vorliegenden Programm drei Strategien verwendet, die dies erfüllen sollen.

So hat z.B. das Popup-Menü, welches auf den verschiedenen Lektionen in der Kursübersicht aufgeht, nicht immer das gleiche Aussehen. So kann in der oberen Anzeigeleiste der Befehl „Lektion löschen“ bei einigen Lektionen fehlen und bei anderen ist dieser vorhanden. Wenn man auf die Kurs-Startseite bzw. die Kurs-Bewertung geht, so steht sogar nur ein Befehl, der Befehl „Lektion bearbeiten“, zur Verfügung. Dies hat auch seinen Sinn. Denn wenn ein Befehl nicht angezeigt wird, so darf dieser Befehl auch nicht auf dieser Lektion ausgeführt werden. So fehlen bei allen eingebundenen Lektionen der Befehl „Lektion löschen“, da der Autor keine Lektion löschen darf, die sich in der Verzweigungsstruktur des Kurses befindet, da dies zum einem ein Fehlverhalten beim Ablauf der Lerneinheiten im Backend hervorrufen würde und zum anderen nicht vom Autor gewünscht sein dürfte, da eine fehlende Lektion innerhalb der Verzweigungsstruktur ein Loch aufreißen könnte, so dass lose Verknüpfungen entstehen. Die erste Strategie setzt also darauf, dass der Autor nur diejenigen Befehle, Menüpunkte etc. zu Gesicht bekommt, die auch zum jeweiligen Zeitpunkt Sinn machen. Darunter fällt unter anderem auch, dass im Fenster mit der Abbildung der Verzweigungsstruktur der aktuellen Lektion, der Befehl „Zyklusdurchbruch“ nur bei Lerneinheiten, die sich in einem Zyklus befinden, angezeigt werden. Des weiteren sind die Befehle „Lösche Link“ und „Lösche Teilbaum“ nur auf Nachfolger-Lektionen möglich, da derselbe Befehl auf einer der Vorgänger-Lektionen zu einem ungewollten Ergebnis führen könnte, da z.B. bei einer Teilbaum-Löschung einer Vorgänger-Lektion automatisch auch die aktuelle Lektion mit gelöscht werden würde. Unter diese Strategie fallen im ganzen Autorensystem noch etliche weitere Punkte. So wird bei der Bearbeitung der Kurs-Startseite dem Autor nur die Möglichkeit gegeben, dass er Präsentationskomponenten einbauen kann, was dadurch erreicht wird, dass in der Lerneinheit-Übersicht nur die Schaltfläche „Präsentation“ zur Verfügung steht. Entsprechendes gilt bei der Kurs-

Bewertung mit der Schaltfläche „Bewertung“. Ein weiteres Beispiel ist dadurch gegeben, dass die Filter der Dateiauswahldialoge so gestaltet sind, dass nur Dateien des entsprechenden Typs angezeigt werden. Und als letztes Beispiel sei noch genannt, dass der Autor die Startseiten nicht in die Verzweigungsstruktur ziehen kann, da diese nur einmal am Anfang des Kurses bzw. der Lektion erscheinen dürfen. Allgemein kann man diese Strategie dadurch bezeichnen, dass dem Autor nichts angeboten wird, was zu einem fehlerhaften Zustand führen würde.

Jedoch kann man dem Autor nicht alles verbieten oder nicht anzeigen, was einen Fehler machen würde. Bei ein paar Handlungen sollten dem Autor vorübergehende Inkonsistenzen erlaubt sein, was zum Beispiel des oben erwähnten dynamischen Zyklus führt. Während des Erstellens der Autorensteuerung sollte es dem Autor vorübergehend erlaubt sein, einen derartigen, dynamischen Zyklus kurzfristig anlegen zu können, um ihn nicht durch ein Verbot einer derartigen Struktur beim Erstellen des Kurses zu behindern. Denn es könnte durchaus möglich sein, dass er diese Struktur in seiner weiteren Arbeit wieder aufbricht. So ist es dem Autor also erlaubt, vorübergehend eine falsche Struktur in seinem Kursaufbau zu haben. Jedoch muss vor dem Abspeichern des Kurses dies überprüft werden und wenn der Autor vergessen hat, die Struktur aufzubrechen, so muss das Programm eine entsprechende Warnmeldung ausgeben. Dies ist nun die zweite Strategie, die zur Anwendung kommt. Der Autor wird beim Abspeichern des Kurses gewarnt, wenn Fehler in der Struktur der Verzweigung des Kurses sind. Dabei gibt es folgende Fehler: Der erste Fehler des dynamischen Zyklus wurde bereits im vorigen Kapitel angesprochen. Ein weiterer Fehler besteht darin, dass der Autor einen Deadlock erzeugt hat.

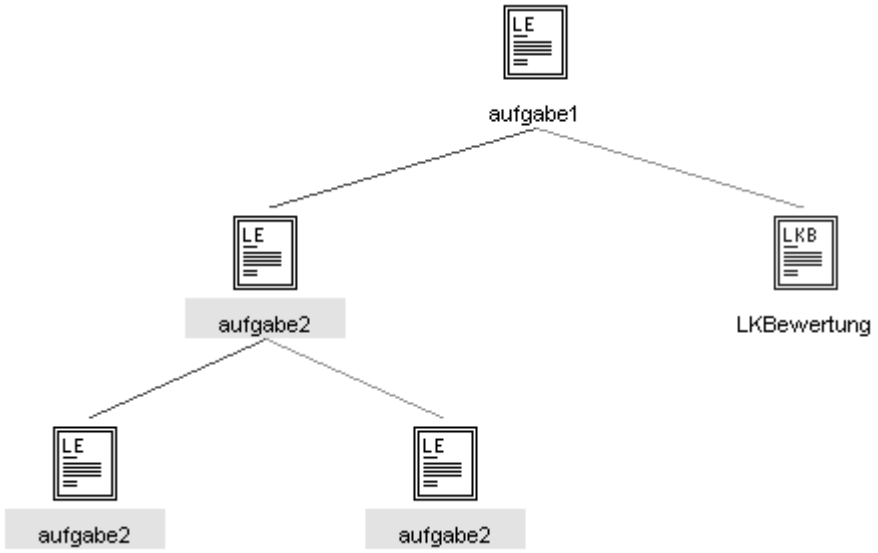

Abbildung 10: Der linke Teilbaum ist ein Deadlock

In Abbildung 10 ist ein derartiger Deadlock anhand der Verzweigungsstruktur einiger Lerneinheiten angezeigt. Der Lernende kann ausgehend von der Lerneinheit „aufgabe1“ zur Lerneinheit „aufgabe2“ kommen und von dieser nicht mehr zur Lektion-Bewertung, da alle Verzweigungen wiederum auf die Lerneinheit „aufgabe2“ verweisen. Beim einem Deadlock handelt es sich im Prinzip um einen Zyklus ohne Zyklusdurchbruch. Besser gesagt es handelt sich um Lerneinheiten, von denen es kein Weg zur Lektion-Bewertung gibt.

Wie bei einem dynamischen Zyklus, erscheint auch bei einem Deadlock für den Autor eine entsprechende Fehlermeldung und als visuelle Hilfe geht das Fenster der Gesamtübersicht auf und zeigt dem Autor durch farbige Balken unter den jeweiligen Namen der Lerneinheiten bzw. Lektionen die fehlerhaften Stellen der Struktur. Bei diesen beiden Prüfungen handelt es sich um die komplexesten Prüfung auf der Verzweigungsstruktur (zur Implementierung siehe Kapitel 3). Davor werden noch zwei weitere Prüfung durchgeführt, die eine einfachere Natur besitzen. Die erste Prüfung, die auf der Verzweigungsstruktur nämlich gemacht wird, ist diese, dass geprüft wird, ob der Autor alle Verzweigung geschlossen hat und es keine weiteren offenen Links mehr gibt. Sollten nämlich offene Links vorhanden sein, so könnte der Lernende in einen Lernweg mit einer offenen Verzweigung geraten, der ihn geradewegs auf einen Absturz des Backends führen würde, was mit Sicherheit nicht vom Autor gewollt sein dürfte. Wie bei den vorherigen Fehlerfällen auch, geht hier ebenfalls das Fenster der Gesamtübersicht auf und die fehlerhaften Stellen werden mit einem roten Balken gekennzeichnet, wie dies in Abbildung 7 zu sehen ist. Die

zweite Prüfung, die anschließend durchgeführt wird, ist die Prüfung, ob die Verzweigungsstruktur der Kursebene eine Kurs-Bewertung bzw. die Lektionebene eine Lektion-Bewertung besitzt. Sollte dies nicht der Fall sein, so würde kein Lernweg einen Abschluss finden. Die Prüfung der Deadlocks bzw. der dynamischen Zyklen werden bei einer fehlenden Bewertungskomponente nicht durchgeführt, da ohne eine Bewertungskomponente sich jede Lektion bzw. Lerneinheit in einem Deadlock befinden würde. Erst wenn die beiden ersten Prüfungen erfolgreich sind, werden die Prüfungen zu den Deadlocks und zu den dynamischen Zyklen durchgeführt. Die Prüfung der Verzweigungsstruktur erfolgt hierbei immer beim Speichern der jeweiligen Ebene. Für die Ebene der Komponenten innerhalb einer Lerneinheit braucht natürlich keine derartige Prüfung auf der Verzweigungsstruktur erfolgen, da die Komponenten linear hintereinander angezeigt werden. Jedoch müssen hier andere Prüfungen erfolgen. Hierauf sei auf die Studienarbeit Nr. 1852 [Co/Mü] verwiesen.

Dies sind jedoch nicht die einzigsten Prüfungen der Eingaben des Autors im Programm. So wird bei der Eingabe des Namens einer neuen Lektion bzw. beim Kopieren einer Lektion, diese dahingehend geprüft, dass der Autor seine Lektion nicht mit einem vordefinierten Namen wie KursStartseite belegt, bzw. ein ungültiges Zeichen für Dateinamen (\, /, :, *, ?, <, > |) benutzt. Oder es wird beim Abspeichern einer Bewertungskomponente überprüft, ob der Autor eine aufsteigende Reihenfolge für die Bewertungsbereiche eingeben hat. Vorübergehend kann der Autor auch hier einen inkonsistenten Zustand eingeben, der spätestens beim Speichern korrigiert wird. Entsprechend gibt es für die anderen Komponenten Fehlerüberprüfungen, die sich auf die Eingaben in diesen Komponenten beziehen (siehe hierzu auch Kapitel 4).

Damit kommen wir zur letzten Strategie zur Vermeidung von Bedienfehlern, die im Autorensystem zur Anwendung kommt. Hierbei handelt es sich um den sinnvollen Einsatz von Farben und dem entsprechenden Positionieren der Elemente. So besitzen die Verzweigungen auf die nachfolgenden Lektionen der aktuellen Lektion verschiedene Farben, wie das z.B. in Abbildung 6 zu sehen ist. Außerdem befindet sich die rote Linie immer links und die grüne Linie rechts. Diese beiden Farben sollen den Falsch- bzw. den Richtig-Fall anzeigen. Dabei geht der Falsch-Fall immer bei

null Prozent der Punkte der jeweiligen Einheit los und der Richtig-Fall geht bis zu 100 Prozent der Punkte. Dazwischen kann sich eine variable Anzahl von weiteren Fällen befinden, die blau gekennzeichnet sind und intern im Programm als undefiniert-Fälle bezeichnet werden, da der Autor für diese Fälle eine beliebigen Prozentbereich eingeben kann. Die Farben rot und grün wurden deswegen so gewählt, da diese Farben von den Korrekturen der Arbeiten in der Schule den meisten bekannt sein dürften. So wurde ein Fehler meistens mit rot gekennzeichnet und wenn alles richtig war, gab es einen grünen Haken. Bekannt ist diese Farbkombination auch von manchen OK-Dialogen, bei denen die OK-Schaltfläche mit einen grünen Haken zusätzlich gekennzeichnet ist, und die Abbrechen-Schaltfläche mit einem roten X. Die weiteren Verzweigungsarten wie „Weiter“ (schwarz) im Falle einer Präsentations-Lerneinheit und einer weißen Linie für den Zyklusdurchbruch, besitzen diese Farben, um sie von den anderen Verzweigungsarten abzugrenzen und sie so für den Autor zu kennzeichnen. Der Autor sieht also anhand der Farben um welchen der Bewertungsbereiche der entsprechenden Bewertungskomponente es sich handelt und kann so seinen Kurs entsprechend einfacher planen, und vermeidet unter anderem solche Fehler, dass er die Verzweigung des Falsch-Falls nicht mit der Verzweigung des Richtig-Falls verwechselt. Die Wahl der Farben wurde dabei in jedem Bereich des Autorensystems durchgezogen, wo die entsprechenden Elemente im Programm auftauchen. So wurde bei den Bewertungskomponenten darauf acht gelegt das der Falsch- und Richtig-Fall die entsprechenden Farben wie die dazugehörigen Verzweigungen haben. Der Einsatz der Farben wurde dabei nicht nur auf die Verzweigungen beschränkt, sondern auch in der Anzeigenleiste der Lektionen eingesetzt. Lektionen, die bisher nicht in die Verzweigungsstruktur eingebunden sind, werden durch ein schwarzes Symbol gekennzeichnet, während eingebundene Lektionen sich für den Autor dadurch kennzeichnen, dass sie grau sind. Als weiteres sind die Kurs-Startseite und die Kurs-Bewertung blau gekennzeichnet um ihre Bedeutung als Start und Ende des Kurses hervorzuheben und um dem Autor zu signalisieren, dass er diese beim Aufbau der Verzweigungsstruktur nicht vergisst. Die Hervorhebung der Kurs-Startseite und Kurs-Bewertung wird dabei noch verstärkt, dass sie nicht in die anderen Lektionen einsortiert sind, sondern gleich am Anfang der Anzeigenleiste stehen. Auf eine derartige Positionswahl wurde auch bei der Anordnung der Bewertungsbereiche in den Bewertungskomponenten geachtet. So liegt der Bereich ab null Prozent ganz

oben und der Richtig-Fall mit bis zu hundert Prozent der Punkte befindet sich ganz unten. Dazwischen liegen aufsteigend die anderen Bewertungsbereiche. Dies entspricht der aufsteigenden Anordnung der Verzweigungen der Nachfolger-Lektionen in der Kursübersicht bzw. der Nachfolger-Lerneinheiten in der Lektionübersicht von links nach rechts, von null bis hundert Prozent. Mit Hilfe dieser Farbauswahl und der entsprechenden Positionierung der Elemente sollte dem Autor geholfen werden, dass er weniger Bedienfehler macht. Hierzu dient auch noch dieser Punkt, dass die Ebene der Kursübersicht und der Lektionübersicht denselben Aufbau besitzen und sich der Autor nicht an eine andere Bedienung bei der Zusammenstellung der Autorensteuerung gewöhnen muss.

2.7 Backend

Bisher wurde die Autorensteuerung unter dem Aspekt betrachtet, wie sie der Autor im Autorensystem zusammenstellen und festlegen kann. In diesem Kapitel wird nun kurz darauf eingegangen, wie diese Autorensteuerung im Backend zur Anwendung kommt. Dazu sei kurz der Begriff des Backends erklärt. Unter Backend werden vor allem das Servlet und die Auswertungsklassen der einzelnen Komponenten verstanden, die für den korrekten Ablauf und die richtige Anzeige der Lektionen bzw. Lerneinheiten im Browser zuständig sind. Dabei bezieht das Servlet und die verschiedenen Klassen die notwendigen Informationen aus den xml-Dateien (den einzelnen Lerneinheiten), die der Autor im Frontend, also im Autorensystem, angefertigt hat. Die Startseiten der Ebenen nehmen dabei eine besondere Rolle ein, da sie alle Informationen darüber enthalten, welche Lektion nach welcher Lektion bzw. welche Lerneinheit nach welcher Lerneinheit im Browser angezeigt werden soll. Aus diesem Grund müssen die Startseiten auch immer am Anfang der Ebene stehen und angezeigt werden, da ansonsten die notwendigen Informationen fehlen würden. Für die Informationsübergabe besitzen die Startseiten hierzu spezielle Steuerungskomponenten auf die im Kapitel 3 eingegangen wird und in denen die Informationen der Autorensteuerung gekapselt sind. Daneben beinhaltet die Startseite auch die Bewertungskomponente der entsprechende Ebene, die wie schon oft erwähnt den Endpunkt einer jeden Lektion bzw. eines jeden Kurses darstellt. Dies entspricht im Groben der Struktur einer Lerneinheit, bei der am Ende ebenfalls eine Bewertungskomponente steht, sobald diese Lerneinheit mindestens eine

Fragekomponente beinhaltet. Hierzu sei auf die Studienarbeit Nr. 1852 [Co/Mü] verwiesen. Das Backend ist vor allem von technischer Natur, worauf im Kapitel 3 eingegangen wird. In diesem Kapitel sei nur erwähnt wie die Zyklen behandelt werden und was es für Besonderheiten bei den Bewertungskomponenten gibt. Sobald der Lernende auf eine Lerneinheit bzw. Lektion trifft, die er schon einmal besucht hat, werden die erreichten Punkte der Lerneinheiten bzw. Lektionen innerhalb des Zyklus gelöscht. Dies beruht auf dem Prinzip, dass der Lernende eine weitere Chance bekommt, sobald er einen Zyklus beendet hat. Aus diesem Grund werden alle Punkte innerhalb des Zyklus gestrichen, so dass es den Anschein hat, als hätte der Lernende diesen Zyklus nicht durchschritten. Bei einem Zyklusdurchbruch jedoch gelangt der Lernende nicht mehr auf die Anfangs-Lerneinheit, wenn er die maximale Besuchsanzahl überschritten hat, und demnach werden die Punkte des letzten Durchgangs des Zyklus behalten und später in der Bewertungskomponente bewertet. Die Bewertungskomponenten arbeiten nun so, dass sie die erreichten Punkte des Lernenden ins Verhältnis zu den maximal erreichbaren Punkten pro Ebene setzen. Entsprechend des Leistungsbereiches, indem sich der Lernende nun befindet, wird auf einer eigenen Seite die vom Autor gegebene Rückmeldung angezeigt. Bei der Bewertungskomponente der Lektionebene wird zusätzlich noch eine Liste über die erreichten Punkte der einzelnen Lerneinheiten ausgegeben. Entsprechendes gilt für die Kurs-Bewertung.

2.8 Zusammenfassung der Konzeption der Autoren-steuerung

Zusammenfassend seien in diesem Kapitel kurz die wichtigsten Bestandteile der Autorensteuerung dargestellt. Die Autorensteuerung wird auf zwei unterschiedlichen Ebenen zusammengestellt. Auf der Ebene der Kursübersicht werden die Lernwege der Lektionen erstellt und in den Lektionenübersichten der einzelnen Lektionen die Lernwege der Lerneinheiten. Dies geschieht mit Hilfe des einfach zu bedienenden Konzept des Drag & Drop. Dabei wurde Wert darauf gelegt, dass die einzelnen Ebenen unabhängig voneinander sind, damit eine Wiederverwendung von Lektionen, Lerneinheiten, bzw. Komponenten durch Exportieren und Importieren ermöglicht ist. Mit anderen Worten bedeutet dies, dass eine Lerneinheit nichts von der nachfolgenden Lerneinheit wissen darf, was sinngemäß für die Lektionen

untereinander gilt. Die Informationen zur Verknüpfung der einzelnen Einheiten ist immer eine Ebene höher gespeichert. Ganz unabhängig sind die Ebenen aber nicht, was bei den Bewertungskomponenten durchbrochen wird, da diese die Anzahl der Bewertungsbereiche festlegen und dies dementsprechend Auswirkungen auf die Ebene höher besitzt. Neben dem Festlegen der Bewertungsbereiche ist es in den Bewertungskomponenten auch möglich eine spezielle Rückmeldung für jeden einzelnen Bewertungsbereich zu geben. Für die Eingabe der Rückmeldung wurde dabei die Textkomponente verwendet, die in der Diplomarbeit Nr. 2076 [Conradt] entwickelt wurde. Die Bewertungskomponenten berechnen dabei den Bereich, in den der Lernende kommt anhand der erreichten und der maximal erreichbaren Punkten der einzelnen Fragekomponenten auf der Ebene der Lerneinheit, bzw. der einzelnen Lerneinheiten auf der Ebene der Lektion und der einzelnen Lektionen auf der Ebene des Kurses. Für die Zyklen besitzt der Autor zusätzlich noch ein Hilfsmittel, um diese zu durchbrechen, wenn der Lernende eine Lerneinheit bzw. Lektion öfters als maximal erlaubt, besucht hat. Zusätzlich wurde bei der Entwicklung der Autorensteuerung darauf geachtet, dass der Autor durch eine geeignete Farbauswahl sich leicht im Programm zurecht findet. Die vom Autor zusammengestellte Autorensteuerung wird schließlich im Backend vom Servlet benutzt, um den richtigen Ablauf der Lerneinheiten zu ermöglichen. Im nächsten Kapitel wird nun die technische Umsetzung der Autorensteuerung sowohl im Frontend als auch im Backend erklärt.

3 Implementierung

Das Autorensystem teilt sich, wie im vorigen Kapitel schon erwähnt in zwei große Bereiche ein. Zum einen gibt es das eigentliche Autorensystem, welches auf der Festplatte des Autors liegt und mit Hilfe dessen der Autor die verschiedenen Lerneinheiten des Kurses erstellen und sie zu einem Kurs zusammenfügen kann. Als Ergebnis liefert das Programm die Lerneinheiten codiert in xml-Dateien und speichert die einzelnen Dateien in den richtigen Verzeichnissen ab. So werden alle Lerneinheiten einer Lektion in einem Unterverzeichnis mit dem Namen der Lektion gespeichert und die Lektionen liegen wiederum in einem Verzeichnis dessen Namen dem Namen des Kurses entspricht. Alle Kurse liegen dabei in einem Verzeichnis, auf welches der Apache Tomcat-Server verweist (siehe weiter unten).

Der zweite Bereich, das Backend, besteht aus dem Apache Tomcat Server, dem Servlet und den Auswertungsklassen, die für den korrekten Ablauf des Kurses und die korrekte Anzeige der Lerneinheiten zuständig sind. Der Quellcode besitzt hierbei insgesamt einen Umfang von mehr als 25000 Zeilen, der in den nächsten Seiten beschrieben wird. Dabei werden nur die wichtigsten Funktionsweisen erklärt, um nicht den Umfang des Dokuments zu sprengen.

3.1 Frontend

Das Autorensystem benötigt für die Realisierung, der im Kapitel 2 vorgestellten Funktionalität eine Vielzahl von Klassen, die nun systematisch vorgestellt werden. Dabei wird die gleiche Reihenfolge eingehalten, wie sie der Autor bei der Benutzung des Autorensystems nacheinander aufrufen würde. Für die Implementierung wurde die Programmiersprache Java verwendet, deren Hauptvorteil es ist, dass sie plattformunabhängig ist.

3.1.1 Klasse Autorensystem

Die erste Klasse, die aktiviert wird, wenn der Autor das Autorensystem startet, ist die Klasse Autorensystem. Es handelt sich hierbei um eine kleine Klasse, die lediglich

zwei Aufgaben besitzt. Als erstes ruft sie verschiedene Methoden der Klasse Init auf, die dafür sorgen, dass verschiedene Basisinformationen aus der Datei ini.xml in die Klasse Init geladen werden. Bei der Klasse Init handelt es sich dabei um eine statische Klasse, was bedeutet, dass von ihr keine Instanzen gebildet werden können, sondern dass sie nur einmal im ganzen Programm existiert. Jedoch handelt es sich um eine wichtige Klasse, da sie die erwähnten Basisinformationen besitzt, die an den verschiedensten Stellen des Programms benötigt werden und weswegen aus verschiedenen Teilen des Programms ein Zugriff auf die Klasse Init erfolgt. Zu diesen Informationen gehören die verschiedenen Verzeichnispfade wie Basisverzeichnis, Kursverzeichnis etc. und die Informationen über die dem Programm zur Verfügung stehenden Komponenten. Die Klasse Init wurde bereits in der Studienarbeit Nr. 1852 [Co/Mü] programmiert und im Rahmen dieser Diplomarbeit wurden nur ein paar Anpassungen bezüglich der Speicherstruktur der Informationen vorgenommen.
Nachdem die Informationen geladen wurden, erzeugt die Klasse Autorensystem nun eine Instanz der Klasse Hauptfenster und zeigt sie durch den Aufruf der Methode show() an. Die letzte Aufgabe der Klasse Autorensystem ist es schließlich beim Beenden des Autorensystems die Basisinformationen aus der Klasse Init in die Datei ini.xml zurückzuspeichern.

3.1.2 Klasse Hauptfenster

Die Klasse Hauptfenster erweitert die Java-Klasse JFrame und erzeugt das Fenster, welches in der Abbildung 3 zu sehen ist. Die grafischen Elemente, wie die Schaltflächen oder die Auswahl-Box in der Mitte des Fenster werden hierbei im Konstruktor erzeugt. Zusätzlich werden die einzelnen Menüpunkte und Schaltflächen mit den verschiedenen inneren Klassen verbunden, die die Funktionalität der Schaltflächen bereitstellen und weiter unten in diesem Kapitel erläutert werden. Neben diesen inneren Klassen besitzt die Klasse Hauptfenster weitere Methoden, die für den korrekten Ablauf des Programms benötigt werden.

Methode vorhandeneKurse()

Die erste dieser Methoden ist die Methode vorhandeneKurse. Die Aufgabe dieser Methode liegt darin, dass sie feststellt, welche Kurse im Kursverzeichnis liegen.

Hierzu bedient sich die Methode eines FileFilters, der nur diejenigen Verzeichnisse herausfiltert, die eine Datei mit dem Namen KursStartseite.xml beinhaltet. Sollte es diese Datei nicht geben, so handelt es sich bei dem Verzeichnis auch um keinen Kurs. Als Ergebnis wird ein File-Array anlegt, der die Pfade zu den gefundenen Kursen aufnimmt. Dieser Array wird unter anderem dazu benötigt, um die Auswahl-Box in der Mitte des Bildschirms zu füllen.

Methode grafikAktualisieren()

Da der Autor bei seiner Arbeit neue Kurse anlegen, bzw. alte Kurse löschen kann, kann sich die Anzahl der Kurse im Kursverzeichnis verändern. Für den visuellen Update der Anzeige dieser Information in der Auswahl-Box ist nun die Methode grafikAktualisieren() zuständig, indem sie diese Auswahl-Box erneuert.

Methode ungueltigerName(String p)

Die nächste Methode der Klasse Hauptfenster ist eine Methode für die Überprüfung der Eingabe eines Autors, sobald dieser einen Kurs benennt. Diese Methode fällt hierbei unter die im vorigen Kapitel erwähnte zweite Strategie zur Vermeidung von Bedienfehlern. Für die Überprüfung bekommt die Methode einen String übermittelt, der die Eingabe des Autors beinhaltet. Dieser String darf nun keine ungültige Buchstaben (\, /, :, *, ?, <, >, |) besitzen, noch darf er einen reservierten Namen (KursStartseite, KursBewertung, LKStartseite, LKBewertung, ?OFFEN?) haben. Als Ergebnis liefert die Methode einen booleschen Wert zurück, der den Wert false hat, wenn der Autor einen korrekten Namen eingegeben hat.

Methode loeschenMitUV(File l) und Methode kopierenMitUV(File q, File z)

Ebenso wie die obige Methode tauchen auch diese beiden Methoden in allen Klassen der verschiedenen Ebenen auf. Ihre Aufgaben sind es zum einen, ein Verzeichnis mitsamt seinen Unterverzeichnissen zu löschen bzw. zum anderen ein Verzeichnis mit Unterverzeichnissen zu kopieren. Hierbei rufen sich beide Methoden solange rekursiv auf, bis sie auf die unterste Ebene der Verzeichnisstruktur gelangen.

Methode neuenKursErstellen(File neuerKurs, String kursname)

Durch den Namen der Methode, dürfte es schon klar sein, dass diese Methode für die Erstellen eines neuen Kurses zuständig ist. Dies bewerkstelligt sie dadurch, dass mit Hilfe von DOM eine einfache KursStartseite erzeugt wird und in dem neu erzeugten Kursverzeichnis abgespeichert wird. Diese neue KursStartseite beinhaltet dabei nur eine Steuerungskomponente und eine Kurs-Bewertung mit Standardwerten. Diese Methode wird von der inneren Klasse KursNeuAction aufgerufen, was uns direkt zu den inneren Klassen der Klasse Hauptfenster führt.

Die inneren Klassen

Die Klasse Hauptfenster besitzt insgesamt zwölf innere Klasse, von denen hier die sieben wichtigsten beschrieben werden. Bei den anderen Klassen handelt es sich um kleine Hilfsklasse, deren Bedeutung aus dem Quellcode einfach zu bestimmen sein dürfte. Die sieben Klassen, die hier kurz beschrieben werden, erweitern allesamt die Adapterklasse AbstractAction und realisieren die Funktionalität der sieben Schaltflächen bzw. der entsprechenden Menübefehle. In der letzten Methode wurde hiervon schon die Klasse KursNeuAction erwähnt, die die Methode neuenKursErstellen() aufruft, um einen neuen Kurs anzulegen. Neben diesem Aufruf dieser Methode hat die innere Klasse lediglich die Aufgabe, ob es sich bei der Eingabe des Kursnamens um einen gültigen Kursnamen (siehe Methode ungueltigerName()) handelt und ob, es einen Kurs mit dem entsprechenden Namen schon gibt. Sollte der Autor hierbei einen Fehler gemacht haben, wird eine Fehlermeldung ausgegeben. Nachdem der Kurs angelegt und gespeichert wurde, muss letztlich noch ein Aktualisierung der Grafik erfolgen.
Für die inneren Klassen KursUmbenennenAction, KursKopierenAction, KursExportierenAction und KursImportierenAction gilt ähnliches, wie für die innere Klasse KursNeuAction. Es wird ebenfalls der eingegebene Name überprüft und beim erfolgreichem Ausführen wird bei den inneren Klassen KursUmbenennenAction und KursKopierenAction eine Aktualisierung der Auswahl-Box mit den verschiedenen Kursnamen durchgeführt. Die Aufgaben der beiden anderen Klassen dürfte anhand des Namens ersichtlich sein. Sie sind für das Exportieren und Importieren eines

ganzen Kurses zuständig und ermöglichen es dem Autor dadurch eine Sicherungskopie seines Kurse anzulegen.
Bleiben noch die inneren Klassen KursLoeschenAction und KursLadenAction. Die erste Klasse ist dabei schnell erklärt. Sie sorgt für das Löschen des Kurses, welchen der Autor in der Auswahl-Box ausgewählt hat, nachdem er eine Sicherheitsabfrage mit „Ja“ bestätigt hat.
Schließlich gibt es noch die innere Klasse KursLadenAction, die aktiviert wird, wenn der Autor die Schaltfläche „Kurs laden“ drückt bzw. den entsprechenden Menüpunkt auswählt. Diese Klasse führt den Autor eine Ebene tiefer auf die Kursübersicht. Hierzu wird eine Instanz der Klasse KursUebersicht angelegt und angezeigt. Zuvor jedoch wird eine Sicherung des Kurses angelegt, die je nachdem, ob der Autor die Änderungen des Kurses abspeichert oder verwirft, wieder gelöscht wird bzw. den veränderten Kurs ersetzt.

3.1.3 Klasse KursUebersicht und Klasse Kurs

Der Autor befindet sich nach dem Erstellen eines neuen Kurses bzw. beim Laden eines bereits vorhandenen Kurses nun auf der Ebene der Kursübersicht. Da es auf dieser Ebene schon möglich ist, die Ablaufsteuerung der Lektionen einzugeben, besitzt dieses Ebene weitaus mehr Funktionalität als die vorherige Ebene. Aus diesem Grund wurden bei der Entwicklung deswegen zwei Klassen angelegt. Hierbei hat die Klasse KursUebersicht die Aufgabe der grafischen Verwaltung des Fenster, welches in Abbildung 4 zu sehen ist, und die Klasse Kurs sorgt für die Kapselung der Informationen der Autorensteuerung. Wenn man dabei genauer hinschaut handelt es sich bei der Klasse Kurs um die im vorigen Kapitel mehrfach erwähnte Steuerungskomponente der KursStartseite, die der Informationsträger der Autorensteuerung innerhalb der Startseite ist. Wie jede Komponente erweitert auch diese Komponente die Schnittstelle der Klasse InformationsKomponente, die in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt und für diese Diplomarbeit um einige Methoden erweitert wurde.
Wie jede Komponente besitzt auch die Komponente Kurs einen Auswertungsklasse im Backend, die für die Auswertung der Informationen für das Backend zuständig ist. Zur weiteren Funktionsweise dieser Klasse sei aber auf das Kapitel 3.2 „Backend“ verwiesen.

Die Besonderheit der Komponente Kurs im Frontend liegt darin, dass sie im Gegensatz zu den anderen Komponenten, beim Aufruf der Lerneinheit KursStartseite.xml in der Lerneinheiten-Übersicht nicht angezeigt wird, sondern dass man die Informationen der Steuerungskomponente im Anzeigefenster des Ablaufs der einzelnen Lektionen anhand der verschiedenen Verzweigungslinien sieht.
Die Kursebene und die Lektionebene unterscheiden sich dabei nur in Details, weswegen in diesem Kapitel die einzelnen Klassen und Methoden der Kursübersicht ausführlicher beschrieben werden und für die Klassen der Lektionebene auf den Quellcode verwiesen ist.

3.1.3.1 Klasse KursUebersicht

Die Klasse KursUebersicht erweitert die Java-Klasse JDialog und bekommt im Konstruktor eine File übergeben, die auf die Datei KursStartseite.xml des ausgewählten Kurses zeigt, sowie den Namen des Kurses. Diese beiden Angaben werden in globalen Klassenfelder gespeichert, da diese Informationen an verschiedenen Stellen der Klasse benötigt werden. Daneben hat der Konstruktor festzustellen, wie viele Lektionen in dem Kurs vorhanden sind, was durch den Aufruf der Methode neueLKDateien() geschieht. Diese Methode funktioniert dabei auf eine ähnliche Weise wie die Methode vorhandeneKurse() aus der vorherigen Klasse Hauptfenster. Der Unterschied besteht lediglich darin, dass der Filter nun nach Verzeichnissen sucht, die eine Datei mit dem Namen LKStartseite.xml beinhalten. Nachdem dies festgestellt wurde, werden die verschiedenen grafischen Elemente erzeugt, die für das Erscheinungsbild der Kursübersicht nötig sind (siehe Abbildung 4). Der wichtigste Teil des Konstruktor besteht anschließend darin, die Informationen des Kurses zu laden, was durch den Aufruf der Methode kursLaden() gemacht wird und weiter unten näher erklärt wird. Da ein Kurs immer eine Kursstartseite besitzt wird nun diese zur aktuellen Lektion gemacht und im linken, unteren Anzeigefenster der Kursübersicht angezeigt. Dabei wird die Grafikanzeige wiederum durch eine Methode mit dem Namen grafikAktualisieren() auf den neuesten Stand gebracht. Im weiteren werden nun die verschiedenen Methoden und inneren Klassen der Klasse KomponentenUebersicht erläutert, sowie das Konzept des Drag & Drop, welches für das Setzen der Autorensteuerung zuständig ist, und das Zusammenspiel der Klassen KomponentUebersicht mit der Klasse Kurs. An dieser Stelle wird auf die Methoden

ungueltigerName, loeschenMitUV, sowie kopierenMitUV nicht weiter eingegangen, da sie die gleiche Funktionalität besitzen, wie die gleichnamigen Methoden in der Klasse Hauptfenster.

Methode neueLKDateien()

Wie bereits erwähnt besitzt diese Methode eine ähnliche Aufgabe wie die Methode vorhandeneKurse() der Klasse Hauptfenster. Zur Filterung der Verzeichnisse benutzt sie ebenso eine innere Klasse mit dem Namen VorhandeneLektionFilter. Der kleine Unterschied bezüglich der anderen Methode besteht darin, dass die Methode neueLKDateien() einen String-Array erzeugt, indem die Kursstartseite und die Kurs-Bewertung an den ersten zwei Stellen eingebaut werden, obwohl es sich bei diesen nicht um Lektionen handelt. Dies wird gemacht, damit sie bei der Aktualisierung der Grafik als erste zwei Symbole in der Anzeigeleiste der Lektionen erscheinen.

Methode kursLaden()

Die Methode kursLaden besitzt lediglich zwei Zeilen, die aber von zentraler Bedeutung sind. In diesen zwei Zeilen wird nämlich die Datei KursStartseite.xml geladen und die Informationen der Steuerungskomponente werden in einer Instanz der Klasse Kurs gekapselt. Genauer genommen handelt es sich hierbei um eine Instanz der Schnittstellen-Klasse InformationsKomponente, die aber über das Prinzip der Polymorphie dynamisch mit der Klasse Kurs verbunden ist. Der Klasse KursUebersicht steht nun ein globales Klassenfeld KursSteuerung zur Verfügung, auf welches immer dann zugegriffen wird, wenn der Autor eine Änderung an der Autorensteuerung vornimmt. In dem Kapitel 3.1.3.2 „Klasse Kurs“ wird dann näher auf die Manipulationsbefehle der Autorensteuerung eingegangen.

Methode aktuellerZustand(String lkname)

Als nächste Methode nach der Methode kursLaden() wird vom Konstruktor die Methode aktuellerZustand aufgerufen, die dafür sorgt, dass die Lektion mit dem übergebenen Name zur aktuellen Lektion wird. Dabei wird das ebenfalls globale Klassenfeld aktuelleLK gesetzt. Dieses Klassenfeld ist eine Instanz der Klasse

LektionenInformationen, welches alle Informationen einer einzelnen Lektion kapselt. Hierzu gehören vor allem die Informationen über die Vorgänger und Nachfolger der ausgewählten Lektion. Der Zugriff auf die richtige Lektion erfolgt dabei über die Methode getLK() der Komponente KursSteuerung, die im weiteren als Steuerungskomponente bezeichnet wird. Nachdem die Verbindung zur richtigen Instanz der Klasse LektionenInformationen erzeugt wurde, kann nun auch die Anzahl der Vorgänger- und Nachfolger-Lektionen festgestellt werden.

Methode grafikAktualisieren()

Mit Hilfe dieser Informationen kann nun vom Konstruktor die Methode grafikAktualisieren() aufgerufen werden, um den Anfangszustand des Kurses im Fenster Kursübersicht anzuzeigen. Dabei erzeugt die Methode grafikAktualisieren() bei jedem Aufruf die Anzeigenleiste mit den zur Verfügung stehenden Lektionen neu und den Bereich links unten, indem die aktuelle Lektion mit ihren Vorgänger und Nachfolgern abgebildet sind. Die beiden letzten Methoden werden immer dann aufgerufen, wenn sich die aktuelle Lektion ändert oder wenn sich Veränderungen in der Anzeigeleiste der Lektionen ergibt. Dadurch ist immer gewährleistet, dass die gerade aktuelle Situation auf dem Bildschirm angezeigt ist.

Im folgenden werden nun die Methoden beschrieben, die der Autor durch ein Drücken eines Menüpunktes in einem der Popup-Menüs bzw. des Hauptmenüs aktiviert. Im allgemeinen handelt es sich um Manipulationsbefehle bezüglich der Autorensteuerung bzw. einer ausgewählten Lektion. Demzufolge rufen die Methoden meistens eine Methode der Steuerungskomponente auf, die für die eigentliche Manipulation an der Autorensteuerung zuständig ist.

Methode loescheteilbaum(String lkname) und Methode loeschelink(String lkname, String nachfolgerart)

Letzteres gilt vor allem für die Methoden loescheteilbaum() und loeschlink(), die die gleichnamigen Methoden der Klasse Kurs aufrufen. Zur Funktionalität sei deswegen auf das Kapitel 3.1.3.2 „Klasse Kurs“ verwiesen. Hier sei nur gesagt, das die Auswirkungen dieser Methoden darin liegen, dass bei der einen Methode ein ganzer

Teilbaum der Autorensteuerung gelöscht wird und bei der anderen Methode nur eine Verzweigung geöffnet wird.

Methode LKSetzen(String linkart, String ziel)

Neben dem Löschen von Verzweigungen bzw. eines Teilbaum muss es natürlich eine Methode geben, die für das Setzen einer Lektion in den Verzweigungsbaum der Autorensteuerung zuständig ist. Diese Aufgabe übernimmt die Methode LKSetzen(). Diese wird dann aufgerufen, wenn der Autor eine Lektion von der oberen Anzeigeleiste auf eine offene Verknüpfung gezogen hat. Zuerst wird dann festgestellt, ob die ausgewählte Lektion bereits in der Verzweigungsstruktur eingebunden ist, was durch einen Aufruf der Methode LektionVorhanden() auf der Steuerungskomponente geschieht. Die aufgerufene Methode schaut hierfür in der internen Array-Liste der Klasse Kurs nach, in der die bisher in die Autorensteuerung eingebundenen Lektionen stehen. Sollte es der Fall sein, dass es sich um keine neue Lektion in der Verzweigungsstruktur handelt, wird nur der entsprechende Nachfolger der aktuellen Lektion von „offen“ auf den Namen der heruntergezogenen Lektion gesetzt und die Vorgänger der Nachfolger-Lektion werden durch den Namen der aktuellen Lektion ergänzt, die als Vorgänger hinzugekommen ist. Beides geschieht über entsprechende Methodenaufrufe der Steuerungskomponente. Im anderen Fall, wenn die heruntergezogene Lektion bisher nicht in die Verzweigungsstruktur eingebaut ist, wird festgestellt, wie viele Verknüpfungen die neue Lektion besitzt. Hierzu wird die Lektion-Bewertung der entsprechenden Lektion geladen und ausgelesen. Anschließend übernimmt die Methode setzeNeueLK() der Steuerungskomponente den Rest der Aktualisierung der Autorensteuerung. In beiden Fällen muss nun geprüft werden, ob der Autor durch das Setzen der Lektion einen Zyklus geschlossen hat, was durch einen Aufruf der Methode zyklusFeldSetzen() auf der Steuerungskomponente geschieht. Anschließend wird die Grafik aktualisiert. Damit sind die Methoden der Manipulation der Autorensteuerung abgehandelt, bis auf das Setzen des Zyklusdurchbruchs.

Methode zyklusdurchbruchSetzen(String lkname)

Diese Aufgabe obliegt dieser Methode. Es handelt sich dabei um die einzigste Verzweigungsart, die direkt auf der Kursebene gesetzt werden kann. Die Gründe hierfür sind im Kapitel 2.5 „Zyklus und Zyklusdurchbruch“ nachzulesen. Der Aufruf dieser Methode kann nur dann geschehen, wenn sich die entsprechende Lektion auch in einem Zyklus befindet. Sollte dies nicht der Fall sein, so wird im Popup-Menü auch nicht der Befehl „Zyklusdurchbruch“ angezeigt.
Dem Autor wird nach Auslösen des Menüpunktes zuerst ein Dialog angezeigt, der ihn auffordert, die maximale Anzahl von Besuchen und optional eine Rückmeldung an den Lernenden für diese Lektion einzugeben. Hierzu wird eine Instanz der inneren Klasse zyklusEingabe erzeugt, deren Aufgabe die Anzeige des Dialogs ist. Nachdem der Autor den Dialog geschlossen hat, wird wie immer erst überprüft, ob er eine korrekte Eingabe gemacht hat. Sollte dies der Fall sein, werden die neuen Werte in der entsprechenden Instanz dieser Lektion gespeichert.
Auf die richtige Instanz der Klasse LektionenInformationen wird dabei wiederum über die Methode getLK() der Klasse Kurs, also der Steuerungskomponente, zugegriffen, die eine Array-Liste über alle eingebundenen Lektionen besitzt. Nachdem nun auch die Methode des Zyklusdurchbruches erklärt wurde, werden anschließend die Methoden zur Manipulation einer einzelnen Lektion vorgestellt.

Die Methoden zur Manipulation einer einzelnen Lektion (neueLKerstellen(), LKUmbenennen(), LKLoeschen(), LKExportieren(), LKKopieren()) haben hierbei dieselben Aufgaben wie die entsprechenden Methoden eine Ebene höher und ähneln diesen im Aufbau. Deshalb werden diese Methoden hier nicht nochmals behandelt.

Methode LKBearbeiten(String lkname)

Die Methode LKBearbeiten hat jedoch einige Besonderheiten, die hier geklärt werden müssen. Zuerst einmal stellt die Methode fest, um welche Lektion es sich handelt. Dies ist wichtig, da die Kurs-Startseite und die Kurs-Bewertung anders behandelt werden, als die Lektionen. So wird bei der Kurs-Startseite und bei der Kurs-Bewertung eine Instanz der Klasse Lerneinheit erzeugt. Dieser Instanz werden

bestimmte Parameter übergeben, die für die richtige Anzeige des Fenster der Klasse Lerneinheit sorgen. So wird beim Bearbeiten der Kurs-Startseite lediglich die Präsentationsschaltfläche angezeigt, da auf einer Startseite nur Präsentationskomponenten erlaubt sind und ähnliches gilt für die Kurs-Bewertung bei der nur die Schaltfläche „Bewertung“ zur Verfügung steht.
Bei den anderen Lektionen schließlich wird zuerst geprüft, ob sie schon in der Verzweigungsstruktur der Autorensteuerung eingebunden sind oder nicht. Sollte dies nicht der Fall sein wird eine Instanz der Klasse LektionUebersicht geöffnet und der Autor hat die Möglichkeit die ausgewählte Lektion zu bearbeiten.
Der andere Fall ist jedoch komplexer, da hier einige Punkte beachtet werden müssen. Vor dem Bearbeiten der Lektion wird zuerst überprüft, welche Nachfolger-Verzweigungen die Lektion besitzt. Diese werden in vier Array-Listen gespeichert. Dabei nimmt jede Array-Liste die Nachfolger eines der vier Verzweigungstypen (falsch, undefiniert, richtig und weiter) auf. Nachdem die Lektion zur Sicherheit in einer Kopie gespeichert wurde, wird nun eine Instanz der Klasse LektionUebersicht geöffnet. Dieser Instanz werden die vier Nachfolgerlisten übergeben. Dies geschieht aus diesem Grund, weil der Autor durch eine Veränderung der Lektion-Bewertung automatisch die Nachfolger-Verzweigungen mit verändert. Diese Veränderungen werden in diesen Listen abgespeichert. Nachdem der Autor die Lektion beendet hat, wird nun festgestellt, ob sich Veränderung in diesen Listen ergeben haben. Hierzu werden die gespeicherten Listen mit den neuen Listen verglichen und die Änderungen werden in der Steuerungskomponente des Kurses nachgezogen. Änderung können hierbei sein, dass z.B. eine Verzweigung eines undefiniert-Falls weggefallen ist. In diesem Fall muss die alte Verzweigung mittels des Aufrufs der Methode loeschelink geöffnet werden. Ein anderer Fall kann sein, dass eine Verzweigung hinzugekommen ist, was ebenfalls in der Autorensteuerung (in der Steuerungskomponente) festgehalten werden muss. Der letzte Fall ist schließlich jener, dass die Lektion-Bewertung komplett ausgetauscht wurde, was dazu führt, dass es nur noch offene Verknüpfungen von dieser Lektion aus gibt. Nachdem dies alles geprüft wurde, wird die Sicherungskopie der Lektion wieder gelöscht, da die Lektion mit „Speichern“ beendet wurde.
Im Falle des Abbrechens wird die Sicherungskopie zum Original, natürlich müssen in diesem Fall nicht die aufwendigen Prüfungen gemacht werden.

Abschließend wird in dieser Methode noch geprüft, ob sich durch eine eventuelle Manipulation der Lektion-Bewertung und damit der Verzweigungen dieser Lektion, ein Zyklus geöffnet hat, was durch einen Aufruf der Methode zyklusFeldSetzen() der Steuerungskomponente geschieht. Hat sich dabei ein Zyklus geöffnet, so werden alle Verknüpfungen der Art „Zyklusdurchbruch“ des entsprechenden Zyklus gelöscht, da diese nicht mehr nötig sind. Wie immer kommt dann zum Schluss der Aufruf der Methode grafikAktualisieren(), um den neuen Zustand auf dem Bildschirm anzuzeigen.

Methode getResult()

Die letzte Methode der Klasse KursUebersicht schließlich ist die Methode getResult(). Diese Methode hat nur die Aufgabe der Rückgabe des Klassenfelds result, in dem steht, ob der Dialog KursUebersicht über die „Speichern“-Schaltfläche oder die „Abbrechen“-Schaltfläche beendet wurde. Diese Information wird eine Ebene höher benötigt. Die Methode getResult() ist eine allgemeine Methode, die in jeder Dialog-Klasse vorkommt und hat dabei immer die Aufgabe, zurückzugeben, wie der Dialog geschlossen wurde.

Bisher wurde nur geklärt, wie die Schnittstellen der Klasse KursUebersicht zur Klasse Kurs aussehen. Die Hauptaufgabe der Klasse KursUebersicht ist aber eigentlich die Erzeugung des Fensters der KursUebersicht, was mit Hilfe der folgenden drei inneren Klassen geschieht.

Klasse LektionDarstellungOben und Klasse LektionDarstellungUnten

Die Ähnlichkeit der Namen lässt darauf schließen, dass die beiden Klassen ähnlich aufgebaut sind, was auch den Tatsachen entspricht. Dabei bezieht sich das „Oben“ auf die Lektionen in der Anzeigeleiste und das „Unten“ auf die Darstellung einer einzelnen Lektion im linken unteren Fenster der Kursübersicht. Beide Klassen erweitern für die Darstellung einer einzelnen Lektion die Java-Klasse JPanel. Der Aufbau der Grafik gestaltet sich bei beiden Klassen relativ einfach. So wird pro Lektion ein Panel erzeugt, welches das Symbol für eine Lektion in Abhängigkeit des Namens der Lektion aufnimmt, und ein Label, welches den Namen der Lektion

anzeigt. Diese beiden Elemente werden in das Hauptpanel der Klasse eingefügt. Der Unterschied der beiden Klassen besteht darin, dass auf dem inneren Panel verschiedene Mouse-Listener realisiert sind und das das Popup-Menü unterschiedlich aufgebaut ist, was dadurch zu erklären ist, das verschiedene Befehle in der Anzeigenleiste gegenüber dem Fenster links unten zur Verfügung stehen (siehe Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern“). Die Mouse-Listener hingegen sind verschieden, um das Drag & Drop zu realisieren.
Hierfür steht dem Programm eine weitere innere Klasse, die Klasse DragDropInformationen zur Verfügung.
Beide Klassen (LektionDarstellungOben und LetkionDarstellungUnten) können dabei auf eine Instanz der Klasse DragDropInformationen zugreifen und sich dadurch die notwendigen Informationen für das Drag & Drop zukommen lassen. Der Beginn des Drag & Drop ist dabei dieser, dass der Autor eine Lektion in der oberen Anzeigeleiste angeklickt hat und gedrückt hält. Diese Aktion wird in der entsprechenden Instanz der Klasse LektionDarstellungOben im Mouse-Listener mousePressed registriert und es werden die Daten in der Klasse DragDropInformationen zurückgesetzt und der Mauszeiger wird in ein Lektion-Symbol umgewandelt. Zusätzlich wird der Name der Quelle in der Instanz der Klasse DragDropInformationen gespeichert. Im weiteren Verlauf kann der Autor nun den Mauszeiger über den ganzen Bildschirm bewegen. Sollte er dabei in den Bereich einer offenen Verknüpfung geraten, so wird dies vom Mouse-Listener mouseEntered der entsprechenden Instanz der Klasse LektionDarstellungUnten registriert. Diese Instanz schreibt nun die Verknüpfungsart und den Namen des Ziels, was in diesem Fall immer den Namen „offen“ hat, in die globale Instanz der Klasse DragDropInformationen. Sollte der Autor den Bereich der Verknüpfung wieder verlassen, so werden die Informationen wieder gelöscht. Wenn der Autor nun aber die Maustaste loslässt, so handelt es sich hierbei um einen mouseReleased-Event, welches von der Instanz der Klasse LektionDarstellungOben, die das Drag & Drop eingeleitet hat, abgefangen wird. Hier wird nun, nachdem der Mauszeiger wieder seine ursprüngliche Form hat, geprüft, ob das Setzen der Lektion möglich ist und es wird die Methode LKSetzen(), die weiter oben schon beschrieben wurde, aufgerufen. Die anderen Mouse-Listener der beiden Klassen haben schließlich noch die Aufgaben, das Popup-Menü anzuzeigen bzw. bei einem Doppelklick die Methode LKBearbeiten() aufzurufen, um die entsprechende Lektion zu manipulieren.

Klasse AblaufAnzeigePanel

Die dritte Klasse der Anzeige ist schließlich für die Darstellung der aktuellen Lektion mitsamt ihrer Vorgänger- und Nachfolger-Lektionen im Fenster links unten zuständig. Dabei verzichtet sie auf ein vorgefertigtes Layout von Java und berechnet die Positionen der einzelnen Lektionen selbst. Hierzu benötigt sie die Angaben, wie viele Vorgänger- bzw. Nachfolger-Lektionen die aktuelle Lektion besitzt. Diese Angaben stehen der Klasse durch die Methode aktuellerZustand(), die weiter oben beschrieben wurde, zur Verfügung. Neben den einzelnen Lektionen-Darstellungen, bei denen es sich um Instanzen der Klasse LektionDarstellungUnten handelt, werden in dieser Klasse auch die verschiedenen Verknüpfungslinien berechnet und gezeichnet.

Die Klasse KursUebersicht wäre damit zum größten Teil besprochen bis auf ein paar weitere innere Klassen. Die Klasse LektionImportierenAction übernimmt dabei eine ähnliche Aufgabe wie die innere Klasse KursImportieren der Klasse Hauptfenster und die innere Klasse LektionNeuAction ruft lediglich die Methode neueLKerstellen() auf, die weiter oben schon besprochen wurde. Interessant sind hier noch die Klassen KursSpeichernAction und GUAction.

Klasse GUAction

Die Klasse GUAction wird aktiviert, wenn der Autor die Schaltfläche Gesamtübersicht drückt. Wenn dies der Fall ist, holt sich die Klasse GUAction aus der Steuerungskomponente die Positionen der einzelnen Lektionen. Dabei werden die Positionen von der Methode GUPositionen() der Klasse Kurs immer neu berechnet, da sich die Anzahl der Lektionen innerhalb der Verknüpfungsstruktur durch die Arbeit des Autors an dieser ständig verändert. Die Positionen werden dann an eine Instanz der Klasse GesamtUebersicht übergeben, die für die weitere Anzeige sorgt. Abschließend stellt die Klasse GUAction fest, ob der Autor eine bestimmte Lektion in der Gesamtübersichtsdarstellung angeklickt hat und machte diese durch die Methode aktuellerZustand() zur aktuellen Lektion, weswegen auch die Grafik auf den neuesten Stand gebracht werden muss.

Klasse KursSpeichernAction

Nachdem der Autor seinen Kurs zusammengestellt hat, muss er diesen abschließend speichern und ruft durch das Drücken der entsprechenden Schaltfläche bzw. des entsprechenden Menüpunktes diese Klasse auf. In dieser Klasse werden nun die verschiedenen Prüfungen durchgeführt, die im Kapitel 2.6 „Strategie zur Vermeidung von Bedienfehlern“ beschrieben wurden. Dabei führt die Klasse KursSpeichernAction nicht direkt die Prüfungen durch, sondern übergibt diese Aufgabe an die Klasse Kurs über den Aufruf der Methoden keineoffeneLinksPruefung(), isEingebunden() und mitKursBewertungVerbunden(). Wenn alles richtig ist, werden die Steuerungsinformationen des Klassenfelds KursSteuerung in der Startseite des Kurses gespeichert. Damit ist nun die Klasse KursUebersicht besprochen und im nächsten Kapitel werden nun die verschiedenen Algorithmen der Klasse Kurs erläutert, die erst die richtige Bedienung der Oberfläche ermöglichen.

3.1.3.2 Klasse Kurs

Die Klasse Kurs erweitert die Schnittstelle InformationsKomponente und ist im Prinzip eine Ansammlung von Methoden, die von der Klasse KursUebersicht benötigt werden. Die Methoden dienen dabei alle dem Zweck die Array-Liste lektionen zu manipulieren. Diese Array-Liste beinhaltet alle Lektionen, die bereits in die Autorensteuerung eingebaut sind. Da für jede dieser Lektionen eine Vielzahl von Informationen gespeichert werden müssen, sind die einzelnen Elemente der Array-Liste vom Typ der Klasse LektionenInformationen, die die Informationen einer einzelnen Lektion kapselt.

Klasse LektionenInformationen

Diese Klasse ist dabei ziemlich einfach aufgebaut. Sie besitzt verschiedene Klassenfelder, in denen die Namen der Vorgänger und Nachfolger in Abhängigkeit, davon, an welcher Verknüpfungsart sie hängen, abgespeichert sind. Zusätzlich besitzt die Klasse auch Informationen über die Lektion selbst, wozu u.a. der Name gehört oder auch, ob sich die Lektion innerhalb eines Zyklus befindet. Für diese

Klassenfelder steht nun eine Vielzahl von Methoden zur Verfügung, die den Zweck haben, diese zu verändern.
An dieser Stelle soll aber nun nicht jede einzelne davon aufgezählt werden, da dies ziemlich schnell ziemlich langweilig werden würde. Hier sei auf den Quellcode verwiesen und die wichtigsten Methoden werden im Zusammenhang mit der Besprechung der Klasse Kurs vorgestellt.

Da die Klasse Kurs die Schnittstelle für Komponenten erweitert, überschreibt die Klasse Kurs verschiedene Methoden um einen korrekten Ablauf mit dem Autorensystem zu gewährleisten. Hierzu gehören die Methoden laden und speichern, die dafür sorgen, dass die Informationen aus der entsprechenden xml-Struktur der Kursstartseite geladen und in die Array-Liste lektionen gespeichert werden. Entsprechend gilt bei der Methode speichern der umgekehrte Weg. Des weiteren gehören hierzu die Methoden getKlasse() und getKategorie(), die für die richtige Identifizierung der Komponente gegenüber der aufrufenden Stelle sorgen. Die übrigen Methoden sind ausschließlich Methoden, die es nur in der Komponente Kurs gibt und werden nun der Reihe nach besprochen. Hierfür sollte man sich kurz klar werden, dass es sich bei der Autorensteuerung im Prinzip um einen Graphen handelt, der manipuliert wird, weswegen einige der Methoden im Prinzip Graphen-Algorithmen sind.

Die Nachfolger-Methoden

In der Komponente Kurs gibt es insgesamt vier Nachfolger-Methoden, deren Aufgaben es ist, die Namen der Nachfolger einer bestimmten Lektion in einer Array-Liste zurückzugeben. Dabei unterscheiden sich die Methoden nur darin, welche Arten von Nachfolger zurückgegeben werden. Der Aufbau der Methoden ist prinzipiell aber derselbe. Zuerst wird ein Verweis auf die Instanz der Lektion in der Array-Liste erzeugt, deren Name der Methode übergeben wurde. Anschließend werden alle Nachfolgerlisten der Lektion durchgegangen und die Rückgabe-Liste wird erzeugt. Nun zu den einzelnen Methoden. Die Methode getDirekteNachfolger() liefert alle Nachfolger einer Lektion zurück, wobei jeder Nachfolgername nur einmal in der Rückgabeliste vorkommt und die offenen Verknüpfungen nicht als Nachfolger zählen. Bei der Methode getAlleDirekteNachfolger() gilt ähnliches wie bei der vorigen

Methode, jedoch werden hier doppelt oder mehrfach vorkommende Nachfolgernamen auch doppelt bzw. mehrfach in die Rückgabeliste eingefügt. Schließlich werden bei der Methode getAlleDirekteNachfolgerMitOffen() auch noch die offenen Verzweigungen zurückgegeben. Die letzte der Nachfolgermethoden, die Methode getAlleDirekteNachfolgerLinktypen, unterscheidet sich von den anderen Methoden, da sie nicht die Namen der Nachfolger-Lektionen zurückgibt, sondern in ihrer Rückgabeliste die Namen der Verzweigungstypen beinhaltet. Bei allen Methoden wird hierbei die interne Reihenfolge der Nachfolger beachtet. So werden zuerst die Nachfolgernamen der Verzweigungsart „falsch“, dann die der Verzweigungsart „undefiniert“, „richtig“, „weiter“ und „zyklusdurchbruch“ in die Rückgabeliste geschoben.

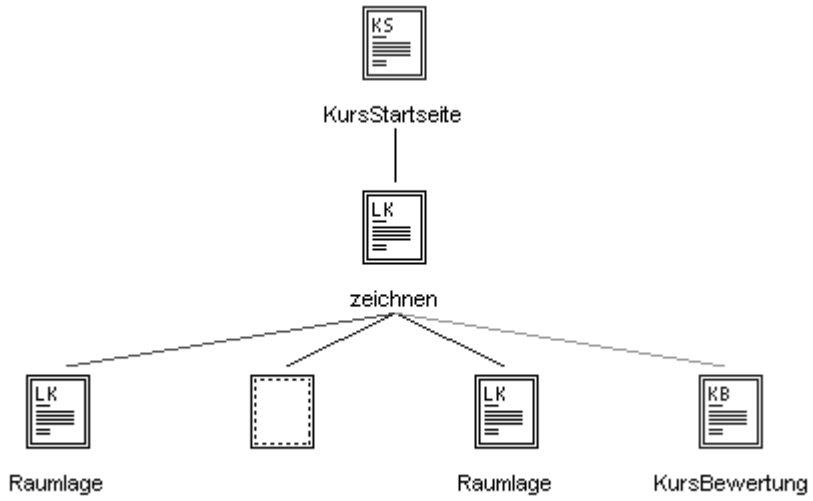

Abbildung 11: Nachfolger-Methoden

Beispielhaft sei auf die Abbildung 11 verwiesen. Die Methode getDirekteNachfolger() liefert für die Lektion „zeichnen“ die Rückgabeliste (Raumlage, Kursbewertung), die Methode getAlleDirekteNachfolger() liefert (Raumlage, Raumlage, KursBewertung), während die Methode getAlleDirekteNachfolgerMitOffen() die Liste (Raumlage, ?OFFEN?, Raumlage, KursBewertung) liefert. Die Zeichenfolge ?OFFEN? ist hierbei der interne Name für eine offene Lektion. Die Methode getAlleDirekteNachfolgerLinktypen() gibt schließlich die Liste (falsch, undefiniert0, undefiniert1, richtig) zurück.

Methode getDirekteVorgaenger(String lkname)

Das Pendant zu den Nachfolgermethoden bildet die Methode getDirekteVorgaenger(), die eine ähnliche Aufgabe wie die anderen Methoden einnimmt. Hier stehen aber in der Rückgabeliste nun die Namen der Vorgängerlektionen. Außerdem werden hier nicht die vielen Unterscheidungen wie bei den Nachfolgern gebraucht, da es zu den Vorgängern keine verschiedenen Verzweigungsarten gibt und ebenso keine offene Verbindungen. Des weiteren kommt in der Vorgängerliste einer Lektion der Name einer einzelnen Lektion von vorneherein nur einmal vor.

Die Nachfolger- und Vorgänger-Methoden sind dabei allesamt Hilfsfunktionen, die von den komplexeren Methoden der Klasse Kurs benötigt werden.

Methode alledirekteWege(String start, String ziel) und Methode direkteWege(String start, String ziel, ArrayList einzelnerWeg)

Die ersten dieser komplexeren Methoden sind hierbei die Methoden alledirekteWege() und direkteWege(). Diese beiden Methoden arbeiten eng miteinander zusammen und haben die Aufgabe, alle Wege festzustellen, die es von einer übergebenen Start-Lektion zur Ziel-Lektion gibt, und diese in einer Array-Liste zurückzugeben. Die Aufgabe der Methode alledirekteWege() besteht dabei nur darin, die globale Array-Liste alleDirekteWege zu initialisieren und die Methode direkteWege mit den Parameter start, ziel, sowie einer Array-Liste mit dem Anfang des Weges, aufzurufen. In dem letzten Parameter steht dabei anfangs der Name der Anfangs-Lektion, da diese den Start jedes einzelnen Weges darstellt. Die Methode direkteWege() arbeitet nun auf diesen Übergabeparameter. Zuerst stellt die Methode fest, über welche Nachfolger die Lektion, deren Name im Parameter start übergeben wird, verfügt. Anschließend wird für jeden Nachfolger geprüft, ob es sich bei dem Nachfolger um das gesuchte Ziel handelt. Wenn ja, dann wird die Array-Liste einzelnerWeg in die Array-Liste alledirekteWege geschoben, da jetzt ein Weg durch den Graphen bzw. durch die Autorensteuerung gefunden wurde. Im anderen Fall wird geprüft, ob der Nachfolger sich schon in der Array-Liste einzelnerWeg befindet. Sollte dies der Fall sein, muss dieser Weg nicht weiter verfolgt werden, da der

Algorithmus einen Zyklus durchschritten hat, die Methode aber nur die direkten Wege finden soll. Wenn der Nachfolgername aber nicht in der Array-Liste einzelnerWeg vorkommt, so besteht weiterhin die Möglichkeit, dass der Weg zur Ziel-Lektion führt. Aus diesem Grund wird die Methode direkteWege() rekursiv aufgerufen. Dabei wird der Nachfolgername als start-Parameter übergeben und die Array-Liste einzelnerWeg ist um den Nachfolgernamen angewachsen. Durch den rekursiven Aufruf auf jedem Nachfolger findet die Methode direkteWege schließlich jeden einzelnen Weg vom Start zum Ziel, wenn es einen gibt.

Methode mitKursBewertungVerbunden()

Eine wichtige Anwendung der beiden oberen Methode ist dabei in der Methode mitKursBewertungVerbunden() gegeben. Diese Methode realisiert eine der Prüfungen, die beim „Kurs speichern“ durchgeführt werden. Wie der Name schon sagt soll diese Methode prüfen, ob es von jeder Lektion ein „Lernweg“ zur Kurs-Bewertung gibt. Dazu ruft sie für jede Lektion, die in der Array-Liste lektionen gespeichert ist die Methode alledirekteWege() auf. Sollte nach dem Aufruf das globale Klassenfeld alleDirekteWege die Größe null haben, so wurde kein Weg gefunden und der Autor hat in diesem Fall einen Deadlock eingeben. Dabei wird die Funktionsweise der Methode direkteWege() durch ein boolesches Klassenfeld dahingehend modifiziert, dass diese auch dynamische Zyklen erkennen kann. So wird nämlich in der Methode direkteWege() zuerst geprüft, ob die maximale Besuchsanzahl der aktuellen Start-Lektion ungleich null ist. Ist dies der Fall, so dürfen nicht die normalen Nachfolger der Lektion auf einen Weg zur Kursbewertung geprüft werden, sondern es muss der Nachfolger, der in der Verzweigungsart „zyklusdurchbruch“ steht, benutzt werden. Dabei geht der Algorithmus von der Situation aus, das bei allen Lektionen mit einen Zyklus-Durchbruch die maximale Besuchsanzahl überschritten ist. Auf diese Weise kann festgestellt werden, ob es einen dynamischen Zyklus in der Autorensteuerung gibt. Diese spezielle Funktionsweise der Methode direkteWege() ist aber nur im Falle dieser Prüfung notwendig. Ansonsten funktioniert die Methode so wie sie oben besprochen wurde.

Die weiteren Prüf-Methoden auf der Verzweigungsstruktur der Autorensteuerung sind weitaus weniger komplex. So prüft die Methode isEingebunden() die Array-Liste

lektionen darauf ab, ob der übergebene Lektionennamen vorhanden ist und die Methode keineOffeneLinksPruefung() schaut in den Nachfolgerlisten jeder einzelnen Lektion nach, ob eine offene Verzweigung vorhanden ist.
Als nächstes werden nun die Lösch-Methoden betrachtet, die für das korrekte Löschen eines Teilbaums bzw. einer Verknüpfung sorgen.

Methode loeschteilbaum(String lkname) und Methode loeschen(String lkname)

Die Methode loescheteilbaum hat die Aufgabe einen ganzen Teilbaum aus der Autorensteuerung herauszulöschen. Hierzu initialisiert sie zuerst das globale Klassenfeld loeschListe und ruft die Methode loeschen() auf. Diese Methode sorgt für das korrekte Füllen der Löschliste. Nachdem der Aufruf zur Methode loescheteilbaum() zurückgekehrt ist, wird anschließend die Array-Liste lektionen aktualisiert, indem die Lektionen gelöscht werden, deren Namen in der loeschListe vorkommen.
Die Methode loeschen() bedient sich für das Füllen der loeschListe dabei ebenfalls des Prinzips der Rekursivität. Als erstes schiebt die Methode den als Parameter übergebenen Namen der Lektion in die loeschListe. Danach müssen die verschiedenen Verzweigungen, die auf diese Lektion gezeigt haben bzw. von ihr aus gingen gekappt werden, damit die Autorensteuerung auf keine Lektion zeigt, die nicht mehr in der Verzweigungsstruktur der Autorensteuerung vorhanden ist.
Um dies zu bewerkstelligen, holt sich die Methode loeschen() alle Vorgänger und alle Nachfolger der zu löschenden Lektion. Für jeden einzelnen Vorgänger muss nun jede Verzweigung, die auf diese Lektion zeigt, mit dem Wert „offen“ belegt werden, was durch einen Aufruf der Methode ersetzeNachfolger() auf der Vorgänger-Lektion mit den entsprechenden Parameter geschieht. Die Methode ersetzeNachfolger() ist hierbei ein Beispiel einer Methode, die in der Klasse LektionenInformationen realisiert sind. Entsprechend wird in jeder nachfolgenden Lektion die Vorgänger aktualisiert, indem der Name der zu löschenden Lektion aus der Vorgängerliste entfernt wird. Anschließend werden auf der zu löschenden Lektion selbst alle Nachfolger-Verknüpfungen auf „offen“ gesetzt, damit es keinerlei Verbindungen mehr zu dieser Lektion gibt. Dies ist notwendig, da nun auf jedem bisherigen Nachfolger der Lektion geprüft wird, ob es einen weiteren Weg von der Kursstartseite aus gibt, der nicht über die zu löschende Lektion führte. Wenn dieser Weg nicht vorhanden ist, so ist diese

Nachfolger-Lektion nicht mehr erreichbar und wird ebenfalls aus der Verzweigungsstruktur der Autorensteuerung herausgelöscht, was durch einen rekursiven Aufruf der Methode loeschen() geschieht.
Schließlich hat die Methode loeschen() die korrekte loeschListe zusammengestellt und alle Verknüpfungen in der Autorensteuerung richtig gesetzt.

Methode loeschelink(String lkname, String nachfolgerart)

Auf den ersten Blick scheint die Methode loeschelink gegenüber der vorigen Methode einfacher zu sein, da ja hier nur eine einzelne Verknüpfung gelöscht werden muss und nicht gleich ein ganzer Teilbaum. Doch auch diese Methode besitzt einige Schwierigkeiten, die beachtet werden müssen. Um die Aufgabe durchführen zu können, die dieser Methode bestimmt ist, benötigt die Methode als erstes den Name der aktuellen Lektion und die Verknüpfungsart, die geöffnet werden soll. Anschließend holt sich die Methode den Nachfolger, der an dieser Verknüpfung hängt und belegt erst dann den Wert dieser Verknüpfung mit „offen“. Der nächste Schritt besteht dann darin, dass überprüft wird, ob die nachfolgende Lektion noch an einem anderen Verknüpfungszweig der aktuellen Lektion hängt. Wenn dies der Fall ist, dann darf in der Vorgängerliste der nachfolgenden Lektion die aktuelle Lektion nicht gestrichen werden. Im anderen Fall geschieht dies.
Anschließend muss noch geprüft werden, ob die nachfolgende Lektion nur an der aktuellen Lektion gehangen ist. Wenn dieser Fall eintritt, so existiert kein Weg von der Kursstartseite zu dieser Lektion und die Lektion wird mit Hilfe der Methode loescheteilbaum() aus der Verzweigungsstruktur gelöscht. Damit müsste klar sein, wie die Klasse Kurs verschiedene Verknüpfungen öffnet bzw. wie ganze Teilbäume gelöscht werden können. Bevor man jedoch etwas löschen kann muss man es vorher erst einfügen.

Methode setzeNeueLK(String name, ArrayList nachfolgerlinks, String vorgaenger, String linkart) und Methode LKumbenennen (String alterName, String neuerName)

Und dies wird durch die Methode setzeNeueLK() ermöglicht. Für das korrekte Setzen einer neuen Lektion benötigt die Methode hierbei einige Parameter. Dies sind der

Name der Lektion und die Verknüpfungen, die die Lektion besitzt. Außerdem wird die aktuelle Lektion gebraucht (Parameter vorgaenger) und der Verknüpfungszweig an den die Lektion angehängt wird. Für die neue Lektion wird eine Instanz des Typs LektionenInformationen erzeugt und es werden die Nachfolgerlisten gesetzt, was mit Hilfe entsprechender Methoden dieser Klasse gemacht wird. Außerdem darf nicht vergessen werden, die Vorgängerliste der neuen Lektion mit dem Namen der aktuellen Lektion zu erweitern. Entsprechend muss auf der aktuellen Lektion der Nachfolgername des ausgewählten Verknüpfungszweiges durch den Namen der neuen Lektion ersetzt werden. Nachdem die Verknüpfungen alle richtig gesetzt wurden, wird nun die neue Instanz der Klasse LektionenInformationen in die Array-Liste lektionen geschoben und die neue Lektion ist in der Verzweigungsstruktur der Autorensteuerung eingefügt.
Sollte der Autor bei der Namensvergabe für die neue Lektion aus Versehen einen Fehler gemacht haben, so besitzt er, wie im Kapitel 2 schon gesagt, die Möglichkeit der Umbenennung einer Lektion. Hierzu muss er lediglich den entsprechenden Menübefehl aktivieren. Dabei ist es aber nicht getan, nur die einzelne Datei umzubenennen, wenn die Lektion in der Autorensteuerung eingebunden ist. In diesem Fall wird die Methode LKumbenennen() aufgerufen, die dafür sorgt, dass auch alle Verzweigungen der übrigen Lektionen auf diese Lektion umbenannt werden. Die Funktionsweise ist dabei recht einfach. Die Methode geht durch alle Lektionen durch und schaut in allen Listen und Klassenfelder nach dem alten Namen und ersetzt ihn durch den neuen Namen.

Methode zyklusFeldSetzen() und Methode zyklusLK(String lkname)

Als nächstes werden nun die Methoden vorgestellt, mit deren Hilfe die Klasse Kurs feststellt, ob sich in der Autorensteuerung ein Zyklus befindet. Die Methode zyklusFeldSetzen() wird dabei immer dann aufgerufen, wenn sich eine Änderung in der Verzweigungsstruktur der Autorensteuerung ergeben hat, da es hier möglich ist, dass sich ein Zyklus geschlossen bzw. geöffnet hat. Um nun festzustellen, ob die Autorensteuerung einen Zyklus besitzt, ruft die Methode zyklusFeldSetzen() für jede einzelne Lektion der Array-Liste lektionen die Methode alledirekteWege() auf. Dabei übergibt sie dieser als Start und Ziel des Weges den Namen der Lektion. Dadurch

wird festgestellt, ob ein Weg von der Lektion zur Lektion selbst vorhanden ist, d.h. ob ein Zyklus auf dieser Lektion existiert.
Anschließend wird entsprechend dem Ergebnis das Klassenfeld zyklus der Lektion auf den richtigen Wert gesetzt. Dieser Wert kann nun von der Klasse KursUebersicht über die Methode zyklusLK() abgerufen werden.

Bei den restlichen Methoden der Klasse Kurs handelt es sich um kleine Hilfsmethoden, die von der Klasse KursUerbesicht und der Klasse Kurs selbst benötigt werden, und deren Funktionsweise anhand des Quellcodes leicht zu verstehen sein dürfte. Als weiteres dürfte die Methode neueLKerstellen() nicht weiter erklärt werden müssen, da sie das Pendant zur Methode neuenKursErstellen() der Klasse Hauptfenster auf der Kursebene darstellt. Lediglich die Methoden, die für die Berechnung der Positionen der Lektionen in der Gesamtübersicht zuständig sind, sind noch von komplexerer Natur, worauf abschließend in diesem Kapitel eingegangen wird.

Methode GUPositionen(int mb, int mz, int fb) und Methode ebenenBerechnung(ArrayList quelle)

Im Kapitel 2 wurde die Gesamtübersicht vorgestellt, welche ein hilfreiches Werkzeug für den Autor darstellt, um die ganze Verzweigungsstruktur seines Kurses auf einen Blick zu haben. Da sich durch Manipulationen an der Autorensteuerung diese immer wieder verändert, muss vor jedem Neuaufruf der Gesamtübersicht die Positionen der einzelnen Lektionen neu berechnet werden.
Diese Aufgabe übernimmt die Methode GUPositionen(), während die Methode ebenenBerechnung() die Lektionen feststellt, die auf den einzelnen Anzeigeebenen angezeigt werden müssen. Für die oberste Ebene ist dies immer die Kursstartseite, da sie den Anfang eines jeden Kurses darstellt. Bei der Methode ebenenBerechnung() handelt es sich dabei wiederum um eine rekursive Methode, die sich ebenenweise nach unten arbeitet, bis sie schließlich auf der untersten Ebene ankommt. Der Algorithmus arbeitet dabei so, dass dieser für jede Lektion der übergebenen Array-Liste alle Nachfolger feststellt und diese mitsamt des Namens der Lektion in der Array-Liste aktuelleEbene abspeichert. Dadurch ist später eine Zuordnung der einzelnen Lektionen einer Ebene zu ihren Vorgängerlektionen

möglich. Für jede nachfolgende Lektion wird der Algorithmus dann rekursiv aufgerufen, um die nächste Ebene zu berechnen.
Damit die Lektionen in der Darstellung deutlich voneinander getrennt sind, wird an gewissen Stellen eine Lektion namens „Zwischenraum“ eingefügt. Diese Stellen befinden sich unterhalb der Lektion KursBewertung oder unterhalb einer Lektion, die bereits auf einer höheren Ebene angezeigt wird. (siehe Kapitel 2.3 „Bedienung des Autorensystems und Festlegung des Ablauf einer Lernsitzung“ - Darstellung von Zyklen). Als Abbruchskriterium dient dem Algorithmus dabei, diese Ebene, wenn die Array-Liste quelle nur noch Lektionen des Typs „Zwischenraum“ besitzt. Als Ergebnis hat die Methode nun auf jeder Anzeigeebene die Lektionen berechnet, die angezeigt werden müssen und sie den einzelnen Vorgängerlektionen zugeordnet. Das Ergebnis steht dabei in der Array-Liste alleEbenen der Methode GUPositionen() zur Verfügung, die nun dafür zuständig ist, anhand der Informationen die einzelnen Positionen der Lektionen zu berechnen. Dies geschieht ausgehend von der untersten Ebene, die immer die breiteste Ebene darstellt.
Für die Abstände zwischen den Lektionen, etc. wurde der Methode entsprechende Parameter übergeben. Nachdem die unterste Ebene berechnet wurde, wird anschließend jede einzelne Ebene berechnet, bis der Algorithmus auf der obersten Ebene angelangt ist. Dabei werden die Elternlektionen immer mittig über den Kindlektionen positioniert. Aus diesem Grund müssen bei der Berechnung immer die Werte von zwei Ebenen gleichzeitig herangezogen werden.
Nachdem nun alle Positionen der Lektionen auf allen Ebenen berechnet wurden, stellt die Methode GUPositionen() eine komplexe Rückgabeliste zusammen, in der für alle Ebene neben den Namen und der Position der jeweiligen Lektion auch der Verzweigungstyp mit angeben ist, der auf diese Lektion verweist. Mit Hilfe all dieser Informationen kann die Klasse GesamtUebersicht nun die gewünschte Darstellung der Verzweigungsstruktur erzeugen.

Klasse GesamtUebersicht

Die Informationen werden dabei durch die Klasse KursUebersicht durchgeschleust. Die Klasse GesamtUebersicht hat nun nur noch die Aufgabe, diese Informationen auszuwerten und die Lektionen anzuzeigen. Dabei wird ein Panel angelegt, welches die einzelnen Lektionensymbole aufnimmt. Die Darstellung einer einzelnen Lektion

obliegt hierbei einer ähnlichen Klasse wie der inneren Klasse LektionDarstellungUnten der Klasse KursUebersicht.
Das Panel besitzt wiederum keinen vordefinierten Layout-Manager und legt die einzelnen Lektionen an die durch die obigen Methoden berechnete Positionen. Da die Klasse GesamtUebersicht sowohl für die Kursebene als auch für die Lektionebene benutzt wird, benötigt die Gesamtübersicht für die richtige Anzeige der Symbole einen Parameter, der die Ebenenart übergibt. Neben der Anzeige hat die Klasse GesamtUebersicht schließlich nur noch die Aufgabe den Namen der Lektion zu bestimmen, auf den der Autor einen Doppelklick durchgeführt hat. Dieser wird von der Klasse KursUebersicht abgerufen und weiter verarbeitet (siehe auch weiter oben).

Damit ist eine ausführliche Beschreibung der für die Kursebene benötigten Klassen zu Ende. Die Klassen LKUebersicht, Lektion und LerneinheitenInformationen übernehmen dieselben Aufgaben wie die hier beschriebenen Klassen auf der Lektionebene. Da sie sich nur in Details unterscheiden und prinzipiell dieselben Methoden besitzen, sei auf den Quellcode verwiesen, um den Rahmen dieses Dokuments nicht zu sprengen.

3.1.4 Bewertungskomponenten

Bisher wurde besprochen, wie der Autor die Verknüpfungen, die von einer Lektion bzw. einer Lerneinheit ausgehen, schließen kann. Außerdem wurde gesagt, dass die Anzahl der Verknüpfungen von der Bewertungskomponente einer Ebene tiefer bestimmt wird. In diesem Abschnitt werden nun diese Bewertungskomponenten der drei Ebenen vorgestellt. Hierbei wird die Bewertungskomponente der Ebene der Lerneinheit als erstes beschrieben, da diese die meiste Funktionalität besitzt und auch als erstes programmiert wurde. So stammt diese Komponente aus der Studienarbeit Nr. 1852 [Co/Mü] und wurde für die Diplomarbeit an ein paar Stellen erweitert. Außerdem stellt sie eine Klasse dar, die auch von der Diplomarbeit Nr. 2076 [Conradt] in Hinsicht der nachfragenden Lerneinheiten benutzt wird. Auf das Prinzip der nachfragenden Lerneinheiten sei daher auf diese Diplomarbeit verwiesen.

Klasse LEBewertung

Die Klasse LEBewertung erweitert, wie jede andere Komponente auch, die Schnittstelle für Komponenten, die Klasse InformationsKomponente. Da in der Bewertungskomponente der typische Aufbau einer Komponente benutzt wird, wird an dieser Stelle dieser Aufbau gleich mitbeschrieben. Im Konstruktor der Komponente werden die für die Komponente nötigen Klassenfelder angelegt. Dabei werden diese Klassenfelder immer paarweise angelegt, wobei die eine Art von Klassenfelder, die Werte des letzten konsistenten Zustand der Klassenfelder beinhalten und die andere Art die Werte eines vorübergehenden inkonsistenten Zustand besitzen kann. Dabei ist immer ein konsistenter Zustand erreicht, wenn die Methode datenVerarbeitung (siehe weiter unten) keinen Fehler bei den Eingaben des Autors findet und dieser Zustand abgespeichert wird. Ein möglicher inkonsistenter Zustand tritt hingegen immer dann auf, wenn der Autor die Bewertungskomponente gerade bearbeitet.
Neben diesen Klassenfelder werden auch die Array-Listen angelegt, in denen die Namen der nachfolgenden Lerneinheiten gespeichert sind. Hier sei kurz daran erinnert, dass durch die Manipulation der Bewertungsbereiche dieser Komponente sich die Verzweigungen der Lerneinheit verändern kann, weswegen die Namen der nachfolgenden Lerneinheiten über die Klasse Lerneinheit von der aufrufenden Stelle in der Klasse LektionUebersicht der Bewertungskomponente übergeben wurden. Dabei verändert die Bewertungskomponente nicht die Namen der Lerneinheit, was dem Konzept der Wiederverwendung widersprechen würde, sondern es werden bei den Veränderungen der Bewertungsbereiche entweder neue Elemente hinzugefügt oder vorhandene Elemente gelöscht.

Methode neu(JDialog d) und Methode edit(JDialog d)

Nachdem die Komponente von der Klasse Lerneinheit (siehe Kapitel 3.1.5 „Klasse Lerneinheit“) erzeugt wurde, gibt es für die Bearbeitung der Komponente eine Anzahl von Methoden, die im folgenden aufgezählt werden.
Die Methode neu() wird dabei aufgerufen, wenn der Autor eine neue Bewertungskomponente anlegt. Die Aufgabe der Methode besteht neben dem Initialisieren der Klassenfelder darin, den Editor aufzurufen und ihm ein Liste mit

Eingabefenster zu übergeben, die von der Methode erzeugeEingabePanelListe() erzeugt wurden.
Der Editor wurde in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt und hat die Aufgabe die Eingabefenster der Reihe nach anzuzeigen, damit der Autor die Werte für die Komponente eingeben kann. Daneben ruft der Editor immer die Methode datenVerarbeitung() auf, wenn er zum nächsten Eingabefenster wechselt. Die Aufgabe der Methode datenVerarbeitung() wird weiter unten beschrieben.
Auf dieselbe Weise wie die Methode neu() arbeitet auch die Methode edit(), die ebenfalls den Editor mit den richtigen Eingabefenster aufruft. Der Unterschied besteht darin, dass die Klassenfelder nicht neu initialisiert werden müssen.

Methode datenVerarbeitung(int wahl)

Um Falscheingaben des Autors zu verhindern, sollte es in jeder Komponente die Methode datenVerarbeitung() geben, die die Eingaben des Autors überprüft. Der Editor wurde hierbei auf diese Weise programmiert, dass er die Methode datenVerarbeitung() dann aufruft, wenn er zum nächsten Eingabefenster wechselt und bevor der Editor geschlossen wird. Die Überprüfung der Eingaben des Autors hängt davon ab, wie die Klassenfelder beschaffen sind und welche Werte für die einzelnen Komponenten zwingend erforderlich sind. So wird in der Bewertungskomponente überprüft, ob der Autor für die einzelnen Bewertungsbereiche eine korrekte Prozentzahl in aufsteigender Reihenfolge eingeben hat und ob er vergessen hat, bei der Rückmeldung einen Text anzugeben. Die Eingabe des Textes wird hierbei mit Hilfe der Textkomponente erreicht, die in der Diplomarbeit Nr. 2076 [Conradt] entwickelt wurde.

Methode fertigstellen() und Methode synchronisieren()

Hat der Autor alles richtig eingegeben, so werden beim Verlassen des Editors durch die Methode fertigstellen() die Werte der „inkonsistenten“ Klassenfelder auf die Klassenfelder des konsistenten Zustands übertragen, da nun ein neuer konsistenter Zustand erreicht ist. Das Pendant hierzu ist die Methode synchronisieren, die die Werte des konsistenten Zustands auf die Klassenfelder des inkonsistenten Zustands

überträgt, was z.B. beim Abbrechen des Editors bzw. beim Laden der Komponente nötig ist.

Methode laden(NodeList inhalt) und Methode speichern(int position)

Beim Laden und Speichern wird wie im ganzen Programm DOM benutzt, da sich die Daten der Bewertungskomponente in einer xml-Struktur befinden. Die Methode laden bekommt hierfür von der Klasse Lerneinheit eine NodeList übergeben, in der die nötigen Informationen enthalten sind. Dies entspricht dem offenen Konzept des Autorensystems der Studienarbeit Nr. 1852 [Co/Mü], da nur die Komponente selbst wissen kann, wie sie ihre xml-Struktur verarbeitet muss. Entsprechend liefert die Methode speichern() einen Node zurück, der von der Klasse Lerneinheit an die entsprechende Stelle der Lerneinheit eingebaut wird. Jedoch werden die Methoden laden() und speichern() nicht nur von der Klasse Lerneinheit benutzt, sondern auch von den Methoden importieren und exportieren, die es ermöglichen, dass der Autor eine Bewertungskomponente in einer anderen Lerneinheit wieder verwenden kann.

Methode erzeugeBewertungsBlock(int nummer), Methode erzeugeEingabePanel(int wahl) und erzeugeEingabePanelListe()

Bevor jedoch die Werte der Bewertungskomponente abgespeichert werden können, muss der Autor diese zuerst eingeben, was er, wie schon oben erwähnt, mit Hilfe der Eingabefenster tun kann. Die Erzeugung dieser Eingabefenster obliegt nun diesen drei Methoden. Dabei wird in der Methode erzeugeBewertungsBlock() der graphische Aufbau eines einzelnen Bewertungsbereiches angelegt, der in das Haupteingabefenster eingebaut werden. Schließlich erzeugt die Methode erzeugeEingabePanelListe() eine Liste von Eingabefenster, die für die Komponente notwendig ist. Im Falle der Bewertungskomponente handelt es sich hierbei um eine kleine Liste, da nur ein Eingabefenster benötigt wird.

Methode getDummyLinks()

Im folgenden wird nun auf diese Methoden eingegangen, die speziell für die Autorensteuerung benötigt werden. Die ist zum einen die Methode getDummyLinks(),

die die Anzahl der Verknüpfungen der Bewertungskomponente zurückgibt und damit erst die Möglichkeit auf der Lektionebene bietet, Lerneinheiten miteinander zu verbinden. Damit stellt diese Methode im Prinzip den Ausgangpunkt der Autorensteuerung dar.

Methode setAlteNachfolger(ArrayList f, ArrayList u, ArrayList r) und Methode getNeueNachfolger()

Zum anderen sind diese beiden Methoden dafür zuständig, die Werte der Array-Listen mit den Namen der nachfolgenden Lerneinheiten zu füllen und diese nach dem Bearbeiten zurückzugeben.

Innere Klassen WeitereBewertungAction und EntferneBewertungAction

Bleibt nur noch die Frage, wie die Bewertungsbereiche der Bewertungskomponente gelöscht werden bzw. wie neue Bewertungsbereiche hinzugekommen. Diese beiden Aufgaben übernehmen die inneren Klassen WeitereBewertungAction und EntferneBewertungAction. Die Funktionsweise ist dabei schnell erklärt. Die innere Klasse WeitereBewertungAction erhöht die Anzahl der Bewertungsbereiche und fügt in die entsprechende übergebene Liste der nachfolgenden Lerneinheiten eine weitere Verknüpfung ein und kennzeichnet diese als „offen“.
Im Gegensatz hierzu entfernt die innere Klasse EntferneBewertungAction das entsprechende Element aus der Array-Liste und löscht dabei indirekt die dazugehörige Verknüpfung, wenn gerade in diesem Element der Name einer nachfolgenden Lerneinheit stand. Der Vergleich der alten und neuen Listen der Bewertungsbereiche obliegt im weiteren der Klasse LektionUebersicht, die dann für die richtige Anzeige und die Anpassung der Autorensteuerung sorgt.

Bei den weiteren Methoden bzw. inneren Klassen der Bewertungskomponente handelt es sich schließlich um Methoden, die für die Anzeige der wichtigsten Werte in der Übersicht der Lerneinheit zuständig sind. Hierbei wird von der inneren Klasse LEBewertungPanel ein Panel mit den wichtigsten Werten der Bewertungskomponente angelegt, welches über die Methode getPanel() von der Klasse Lerneinheit abgerufen werden kann.

Daneben werden für das Konzept der nachfragenden Lerneinheit einige Methoden benötigt, die in der Diplomarbeit Nr. 2076 [Conradt] näher beschrieben werden.

Klasse LKBewertung und Klasse KursBewertung

Der Unterschied der Bewertungskomponente der Ebene der Lerneinheit zu den Bewertungskomponenten LKBewertung und KursBewertung fällt schließlich gering aus. So fallen in der Klasse LKBewertung die ganzen Methoden der nachfragenden Lerneinheit weg, da diese Bewertungskomponente für die Ebene einer ganzen Lektion zuständig ist und es keine nachfragenden Lektionen gibt. Ansonsten besteht zwischen diesen Komponenten keinerlei Unterschied, da sich die Funktionsweise nur dahingehend geändert hat, dass die Klasse LKBewertung für die Bewertungskomponente der Ebene der Lektion zuständig ist. Gleiches gilt für die letzte Klasse der Bewertungskomponenten, die Klasse KursBewertung. Diese Klasse ist sogar noch einfacher gestrickt, da hier die Methoden getDummyLinks(), setAlteNachfolger() und getNeueNachfolger() wegfallen, die für die Auswirkungen der Verknüpfungen eine Ebene höher zuständig waren. Da sich die Kurs-Bewertung aber bereits auf der höchsten Ebene eines Kurses befindet, braucht dies in der Klasse KursBewertung nicht mehr beachtet werden. Damit wären fast alle Klassen des Frontends beschrieben, die im Zusammenhang mit der Autorensteuerung stehen. Bis auf die Klasse Lerneinheit, die im folgenden Kapitel beschrieben wird.

3.1.5 Klasse Lerneinheit

Die Klasse Lerneinheit besitzt die Aufgabe, die einzelnen Komponenten einer Lerneinheit anzuzeigen und eine einzelne Lerneinheit zu verwalten. Hierzu lädt sie die nötigen Informationen aus der xml-Datei der Lerneinheit und übergibt die einzelnen Knoten den zuständigen Komponenten. Dieses Prinzip wurde in der Studienarbeit Nr. 1852 [Co/Mü] entwickelt und es wurden an der Klasse Lerneinheit lediglich ein paar Anpassungen für die Besonderheiten der Autorensteuerung durchgeführt.
Zu diesen Anpassungen zählt unter anderem, dass die Klasse Lerneinheit nun auch neben „normalen“ Lerneinheiten spezielle Lerneinheiten, wie die Startseiten der Ebenen, behandeln muss. Um dies bewerkstelligen zu können, werden der Klasse

Lerneinheit über den Konstruktor bestimmte Werte für das Klassenfeld methode übermittelt. Die Besonderheiten der speziellen Lerneinheiten liegen darin, dass unter anderem das Erscheinungsbild des Fenster verändert ist, indem bei der Startseite einer Ebene nur die Präsentationsschaltfläche angezeigt wird und bei den Bewertungskomponenten Kurs-Bewertung bzw. Lektion-Bewertung nur die Schaltfläche „Bewertung“. Außerdem müssen die speziellen Lerneinheiten beim Abspeichern gesondert behandelt werden, da sie geringfügig andere Werte abspeichern als die „normalen“ Lerneinheiten.
Des weiteren besitzt die Klasse Lerneinheit zwei Methoden, damit die Kursübersicht bzw. die Lektionübersicht die Steuerungskomponente der Startseiten abrufen und beim Speichern des Kurses bzw. der Lektion überschreiben kann. Für die Autorensteuerung werden zusätzlich noch die Methoden getDummyLinks(), setAlteNachfolger() und getNeueNachfolger() benötigt, die eine ähnliche Aufgabe wie die gleichnamigen Methoden in den Bewertungskomponenten besitzen. Die Methoden der Lerneinheit haben hierbei nur die Aufgabe, die entsprechenden Informationen von den einzelnen Bewertungskomponenten zu den Klassen KursUebersicht bzw. LKUebersicht und umgekehrt durchzuschleusen, da diese Klassen keinen direkten Zugriff auf die Bewertungskomponenten haben und den Umweg über die Klasse Lerneinheit gehen müssen.
Die letzte Anpassung, die schließlich an der Klasse Lerneinheit gemacht wurde, ist diese, dass die Klasse Lerneinheit nun auch Verweiskomponenten verwalten kann. Dazu sei der Begriff Verweiskomponente kurz erklärt. Bei einer Verweiskomponente handelt es sich um eine Datei, deren Inhalt nicht im Autorensystem erzeugt werden kann. Dies sind z.B. Bilder verschiedener Formate, wie jpg, gif, etc. Hierbei sei auf das Kapitel 4 verwiesen, in dessen Rahmen ein paar Verweiskomponenten erläutert werden, die für den Beispiel-Kurs entwickelt wurden. Demnach kann bei diesen Komponenten die Informationen nicht direkt in die xml-Struktur gespeichert werden, sondern es wird ein Verweis auf eine externe Datei gespeichert. Der Speicherort einer externen Datei befindet sich dabei immer in einem Unterverzeichnis, welches den Namen der Lerneinheit-Datei besitzt. Dadurch ist gewährleistet, dass die externe Datei nur dieser Lerneinheit zur Verfügung steht. Die Klasse Lerneinheit hat nun bei dieser Komponentenart zusätzlich die Aufgabe, die externe Dateien mit zu verwalten. Hierzu geht die Klasse Lerneinheit beim Abspeichern durch die einzelnen Komponenten durch und überprüft, ob und welche externe Dateien die Komponente

benötigt. Die übrigen externen Dateien, die von keiner der Komponente benötigt werden, werden dann aus dem Unterverzeichnis gelöscht. Ähnliches gilt im Falle des Abbrechens der Lerneinheit bei dem ebenfalls überflüssige, externe Dateien gelöscht werden.
Die restlichen Methoden und innere Klassen der Klasse Lerneinheit, wurden schon in der Studienarbeit Nr. 1852 [Co/Mü] benutzt und haben im Rahmen der Diplomarbeit nur kleine Anpassungen erfahren, wobei die Funktionalität erhalten geblieben ist. Da die Klasse ebenfalls eine Schnittstelle für nachfragende Lerneinheiten darstellt, wurden schließlich in der Diplomarbeit Nr. 2076 [Conradt] weitere Methoden für das Konzept der nachfragenden Lerneinheiten hinzugefügt, die in dieser Diplomarbeit nachzulesen sind. Damit sind alle Klassen beschrieben, die das Erstellen der Autorensteuerung im Frontend ermöglichen.
Für den Beispielkurs (siehe Kapitel 4) wurden hierbei im Frontend noch weitere Komponenten erzeugt, deren Beschreibung im Kapitel 4 nachzulesen ist. Im folgenden wird nun die Autorensteuerung daraufhin beschrieben, wie sie im Rahmen des Backends funktioniert.

3.2 Backend

Das Backend unterteilt sich wiederum in drei Bereiche, die jeweils unterschiedliche Aufgaben besitzen. Der erste Bereich stellen dabei die xsl-Dateien dar, die für die Umwandlung der xml-Dateien (der Lerneinheiten) in HTML-Code zuständig sind. Dadurch ist es erst möglich, dass die Informationen im Browser angezeigt werden. Es gibt für jede einzelne Komponente eine eigene xsl-Datei, da die Informationen der verschiedenen Komponenten unterschiedlich behandelt werden. Der Lernende bekommt davon natürlich nichts mit, sondern sieht das Ergebnis im Browser-Fenster, Wenn er dann die angezeigte „Weiter“-Schaltfläche drückt, so wird der zweite Bereich des Backends aktiviert. Es handelt sich hierbei um das Servlet „AutorensystemServlet“. Dieses Servlet hat die Aufgabe die Daten, die ihm von der angezeigten Browserseite geschickt werden, zu sortieren und sie dem dritten Bereich, den Auswertungsklassen, zukommen zu lassen.
Dabei besitzt wiederum jede einzelne Komponente eine eigene Auswertungsklasse, um das Prinzip der Offenheit des Autorensystems zu ermöglichen. Der ganze Backend-Bereich benötigt dabei einen Server, der die Funktionalität der Bereiche

dem Internet zur Verfügung stellt. Die Wahl fiel dabei auf den Apache Tomcat Server, da dieser frei zugänglich ist und die Servlet-Technologie unterstützt. In diesem Zusammenhang wird gleich darauf hingewiesen, dass der Apache Tomcat Server dahingehend modifiziert wurde, dass das Kursverzeichnis des Autorensystems in einer internen Variable gespeichert wurde. Dadurch ist erst der Zugriff auf die Kurse über das Internet möglich. Kurse, die sich in einem anderen Verzeichnis auf dem Computer befinden, werden nicht berücksichtigt. In den nächsten Abschnitten werden nun die einzelnen Bereiche mitsamt ihrer Bedeutung näher beschrieben.

3.2.1 XSL-Dateien

Wie bereits erwähnt sorgen die xsl-Dateien für die Umwandlung der xml-Dateien in HTML-Code. Dabei stellt die Datei Komponenten.xsl die Hauptdatei dar und es gibt in jeder Lerneinheit einen Verweis auf diese Datei. Die Hauptdatei erzeugt dabei den Kopf einer HTML-Seite und fügt für die Kurse notwendige Javascripte ein. So benötigen Kurse mit SVG-Dateien für die Cookie-Übergabe eine spezielle Funktionalität, die im Kapitel 4 näher behandelt wird. Für das Abspielen von Sound-Dateien wird das Tag <bgsound id="play"> benötigt, welches direkt nach dem <body>-Tag eingefügt wird. Als weiteres werden Informationen angelegt, die es bei jeder Lerneinheit gibt und die vom Servlet AutorensystemServlet verarbeitet werden. Dies sind der Name der Lerneinheit und die Angabe der Methode, die das Servlet verwenden soll.

Da es sich um ein offenes System handelt, kann die Datei Komponenten.xsl nichts über die Verarbeitung der einzelnen Komponenten wissen. Aus diesem Grund gibt es für jede einzelne Komponente eine eigene xsl-Datei, die für die Umwandlung der xml-Datei verantwortlich ist. Über den Aufruf `<xsl:apply-templates/>` werden diese Dateien in die Hauptdatei eingefügt.

Zu unterscheiden gibt es hierbei zwei Arten von xsl-Dateien. Auf der einen Seite gibt es die Dateien, die für die Steuerungskomponenten zuständig sind. Dies sind: Kurs.xsl und Lektion.xsl. Auf der anderen Seite, die Dateien für die Präsentationskomponenten, Fragekomponenten und Bewertungskomponenten.

3.2.1.1 XSL-Dateien der Steuerungskomponenten

Auch hier unterscheidet sich der Aufbau zwischen den beiden Ebenen nur geringfügig und es wird deswegen nur die Datei Kurs.xsl beschrieben. Die Kurs-Steuerungskomponente befindet sich, wie in den vorigen Kapitel schon öfters erwähnt, immer an der ersten Position der Kursstartseite, die immer als erstes bei einem Kurs aufgerufen wird. Dadurch ist gewährleistet, dass die wichtigen Informationen der Autorensteuerung rechtzeitig zur Verfügung stehen. Beispielhaft sei an dieser Datei nun gezeigt, welche Informationen weitergeleitet werden:

- klasse: Wird auf „Kurs“ gesetzt und wird im Servlet dafür verwendet, um mit Hilfe des Reflection-Mechanismus die richtige Komponente anzulegen (in diesem Fall eine Komponente des Typs Kurs)
- kategorie: Gibt an, zu welcher Kategorie die Komponente gehört. Hier ist es die Kategorie „Steuerung“.
- position: Position gibt an, an welcher Stelle sich die Komponente befindet. Bei der Komponente „Kurs“ steht hierbei immer die Position 0.

Bei diesen Informationen handelt es sich um Informationen, die es bei jeder Komponente geben muss, um einen korrekten Ablauf im Servlet zu ermöglichen. Dabei erfolgt die Übertragung der Daten dadurch, dass die xsl-Datei die Werte der Informationen aus den xml-Dateien in sogenannte HIDDEN-Konstrukte umwandelt, um diese Informationen versteckt in den HTML-Seiten einzufügen. Diese HIDDEN-Konstrukte werden dann vom Servlet AutorensystemServlet ausgelesen. Bei den weiteren Informationen handelt es sich schließlich um Informationen, die es speziell nur bei der Kurskomponente gibt:

- name: Beinhaltet den Namen des Kurses.
- lektionenanzahl: Gibt an, wie viel Lektionen im Kurs vorhanden sind.

Nun werden die Informationen für jede einzelne Lektion verarbeitet, bei denen es sich vor allem um die Namen der Vorgänger- und Nachfolgerlektionen, sowie die Anzahl der maximalen Besuche auf dieser Lektion handelt. Die einzelnen Informationen im Überblick:

- name: Gibt den Namen der jeweiligen Lektion an.
- nuanzahl: Gibt die Anzahl der Verweise an, die weder den richtig- noch dem falsch-Fall entsprechen.
- vorgaenger: Name einer Vorgänger-Lektion

- nachfolger: Name einer Nachfolger-Lektion, die anhand des Verzweigungstyps unterschieden wird.
- Maxbesuche: Anzahl der maximalen Besuche auf der Lektion, sowie die Rückmeldung beim Eintreten des Zyklusdurchbruch.

Die Namen der Informationen werden dabei noch zusätzlich mit Zahlen ausgestattet, die zu einem die Position der Komponente innerhalb der Lerneinheit angibt und zu anderem werden die Positionen der einzelnen Lektionen innerhalb der Komponente selbst mit übergeben, da das Servlet diese Hinweise benötigt, um die richtige Reihenfolge der Informationen herzustellen und sie den einzelnen Komponenten zuweisen zu können (siehe Kapitel 3.2.2 „Servlet AutorensystemServlet“). Entsprechend werden die Informationen der Autorensteuerung einer einzelnen Lektion, d.h. der Ablauf der Lerneinheiten einer Lektion, mit Hilfe der xsl-Datei Lektion.xsl an das Servlet übergeben. Dabei unterscheiden sich die Übergabe nur in den Namen der einzelnen Informationen.

3.2.1.2 XSL-Dateien der Präsentationskomponenten

In diesem Kapitel wird nun kurz auf das Kapitel 4 vorweggegriffen, in dem die Verweiskomponenten JPG-Bild-Komponente, GIF-Bild-Komponente, SVG-Bild-Komponente und Sound-Komponente beschrieben werden. Für jede dieser Komponente gibt es wiederum eine eigene xsl-Datei. Da es sich um reine Präsentationskomponenten handelt, brauchen die entsprechenden xsl-Dateien neben den allgemeinen Angaben klasse, kategorie und position keine weiteren Informationen an das Servlet weitergeben. Der Rest der Informationen wird gleich hier verarbeitet und dient im allgemeinen nur der korrekten Anzeige der Komponente. So wird bei der JPG-Bild-Komponente und der GIF-Bild-Komponente ein <img>-Tag erzeugt mit dem korrekten Verweis auf die externe Datei im Unterverzeichnis der Lerneinheit, was im src-Attribut des img-Tag gespeichert wird. Des weiteren werden die Attribute id, width und height mit den Angaben aus der xml-Datei gefüllt. Dasselbe gilt im Prinzip für die SVG-Bild-Komponente, wobei hier statt eines img-Tag ein embed-Tag erzeugt wird.

In der Sound-Komponente hingegen, wird nun kein Bild angezeigt, sondern die übergebenen Informationen regeln, wann und wie oft die entsprechende wav-Datei

abgespielt werden soll. Dabei kann optional der Mediaplayer angezeigt werden, um den Lernenden das wiederholte Abspielen der wav-Datei zu ermöglichen.
Für jede Präsentationskomponente gibt es zusätzlich eine xsl-Datei, die für den Fall zuständig ist, dass sich die Präsentationskomponente in einer nachfragenden Lerneinheit befindet. Warum dies so ist, ist in der Diplomarbeit 2076 [Conradt] nachzulesen.

3.2.1.3 XSL-Dateien der Fragekomponenten

Der Vollständigkeit halber wird hier auch gleich die xsl-Datei der SVG-Frage behandelt, die speziell für den Kurs in Kapitel 4 entwickelt wurde. Bei dieser Frage handelt es sich um eine Art Bildfrage, was bedeutet, dass der Lernende statt eines Fragetextes ein Bild zu sehen bekommt und er mit Hilfe der Maus eine Aufgabe auf diesem Bild lösen muss. Durch diese Aktionen setzt der Lernende verschiedene Cookies, die den Antworten bei anderen Fragearten entsprechen. Da es sich bei der SVG-Frage um eine Frageart handelt, müssen im Gegensatz zu den Präsentationskomponenten weitaus mehr Informationen an das Servlet übergeben werden.
Im folgenden sind die wichtigsten Informationen angegeben, die von den der Datei SVGFrage.xsl bearbeitet werden:

- bildquelle: An dieser Stelle steht der Verweis auf die externe Bilddatei der Frage.
- cookie: Beinhaltet den Namen eines Cookie, welches von der SVG-Datei gesetzt werden kann. Es handelt sich hierbei im Prinzip um eine Antwort, die der Lernende durch die Manipulationen der Bilddatei gibt.
- cookieanzahl: Gibt die Anzahl der Cookies an, die von der Bilddatei aus, gesetzt werden können.
- beschreibung: Enthält die Beschreibung der Frage.
- auswertungsanzahl: Unter Auswertung wird verstanden, was der Autor für Kombinationen von Cookies erwartet. Dabei kann der Autor die verschiedenen Cookies in beliebige logische Zusammenhänge zusammenstellen. Jede vom Autor vorgegebene Kombination gilt dabei als eine Auswertungsvorschrift. Auswertungsanzahl gibt nun an, wie viele derartige Auswertungsvorschriften es gibt.

- maximalpunkte: In diesem Tag wird gespeichert, wie viele Punkte der Lernende für diese Frage maximal erreichen kann. In der Regel werden die maximale Anzahl von Punkten im richtig-Fall vergeben, d.h. wenn der Lernende die Frage korrekt beantwortet.

Jeder Auswertungsbereich besitzt dabei mehrere Unter-Tags, die von der xsl-Datei zum Servlet durchgeschleust werden müssen:

- logik: Unter logik ist eine beliebige Kombination der Cookies gespeichert. Z.B. bedeutet ERF(0) AND ERF(1), dass diese Auswertung nur dann ausgeführt wird, wenn der Lernende durch die Manipulation an der Bilddatei das Cookie 0 und das Cookie 1 auf true gesetzt hat. Demnach steht in dem Tag logik die eigentliche Auswertungsvorschrift.
- fall: Fall kann drei unterschiedliche Werte annehmen. Entweder es handelt sich um den richtig-Fall, d.h. die Kombination der Cookies dieser Auswertung ist die richtige, oder um den falsch-Fall, oder es handelt sich um keinen dieser Fälle (undefiniert-Fall).
- punkte: Punkte gibt an, wie viel Punkte der Lernende bekommt, wenn er durch das Setzen der Cookies in diesen Auswertungsbereich kommt.
- feedback: feedback enthält die Rückmeldung, die für den Lernenden ausgeben wird, wenn er diese Auswertungsvorschrift eingegeben bzw. ausgelöst hat.

Des weiteren besitzt die SVG-Frage für die korrekte Anzeige Attribute zur Breite und Höhe, die aber in derselben Weise funktionieren wie die Attribute der SVG-Bild-Komponente (siehe oben). In einer SVG-Frage besteht als weiteres die Möglichkeit mit Hilfe von Javascript Sounddateien abzuspielen, die in der HTML-Seite eingebettet sind. Das Einbetten der Sounddateien wird ebenfalls von dieser xsl-Datei übernommen. Dies geschieht genauso wie bei der Sound-Komponente auch, indem ein embed-Tag des Typs audio/x-wav angelegt wird. Im Gegensatz zur Sound-Komponente werden diese Sound-Komponenten immer versteckt angezeigt, da das Abspielen von der SVG-Frage kontrolliert wird. Die Namen der Sounddateien und die Anzahl der Sounddateien werden ebenfalls an das Servlet weitergegeben.

Schließlich werden noch die Informationen einer möglichen nachfragenden Lerneinheit, die die Frage enthalten kann, in HTML-Code umgesetzt. Außerdem gibt es, wie bei jeder Frage, eine xsl-Datei, die den Fall abdeckt, wenn die SVG-Frage in einer nachfragenden Lerneinheit vorkommt (siehe hierzu Diplomarbeit Nr. 2076 [Conradt]).

3.2.1.4 XSL-Dateien der Bewertungskomponenten

Abschließend werden die xsl-Dateien der drei Bewertungskomponenten betrachtet. Die Informationen der Bewertungskomponenten sind deswegen so wichtig, da sie bestimmen, welche Lerneinheit bzw. Lektion als nächstes angezeigt wird. Der richtige physikalische Link wird dabei vom Servlet mit Hilfe der Dummy-Links der Bewertungskomponenten zusammengesetzt (siehe hierzu Kapitel 3.2.2 „Servlet AutorensystemServlet“). Neben den allgemeinen Informationen, die in jeder Komponente vorhanden sind, bearbeiten die xsl-Dateien noch die Informationen der einzelnen Bewertungsbereiche, die der Autor im Frontend eingegeben hat. Hierzu zählen die Anzahl der Bewertungsbereiche, die Untergrenze der Punkte, die benötigt werden um in diesen Bereich zu kommen, die Rückmeldung an den Lernenden für diesen Bereich und der Link des Bereiches. Link kann dabei einen der Werte falsch, undefiniert und richtig annehmen, der wie oben erwähnt im Servlet verarbeitet wird. Als Besonderheit dieser xsl-Dateien wird als sichtbarer Teil der Bewertungskomponente eine Schaltfläche „Weiter“ erzeugt, die einen Verweis auf das Servlet besitzt. Wenn die Schaltfläche also gedrückt wird, wird das Servlet aufgerufen, nachdem zuvor die Funktion ende des Javascript kurse.js abgearbeitet wurde (siehe hierzu Kapitel 4). Die Unterschiede der Bewertungskomponenten der einzelnen Ebenen besteht wiederum nur darin, dass auf der Ebene der Lerneinheit noch zusätzlich Informationen zur nachfragenden Lerneinheit verarbeitet werden und dass bei der Kurs-Bewertung keine Schaltfläche angezeigt wird, da hier das Ende des Kurses erreicht wurde.

3.2.2 Servlet „AutorensystemServlet“

Das Servlet AutorensystemServlet stellt das zentrale Element des Backends dar und sorgt für die korrekte Anzeige der Informationen der einzelnen Lerneinheiten sowie den korrekten Ablauf der Autorensteuerung. Hierzu landen alle Informationen, die von den xsl-Dateien in HTML-Code umgewandelt wurden, letztendlich hier. Damit das Servlet funktioniert, wird ein Serverprogramm benötigt. Dafür wurde zu Testzwecken der Apache Tomcat Server benutzt, da dieser frei erhältlich war. Außerdem eignen sich Servlets für das Autorensystem sehr gut, da sie ebenfalls die Programmiersprache Java benutzen. Die Informationsübergabe einer HTML-Seite an

ein Servlet kann dabei auf zwei Arten geschehen. Entweder über eine Get-Anfrage oder eine Post-Anfrage. Bei Get-Anfragen werden alle Informationen an die URL angehängt und sind damit sichtbar, was natürlich ein Nachteil ist, da der Lernende unter anderem auch die Antwort einer Frage sehen kann. Ein weiterer Nachteil von Get-Anfragen ist es, dass man nur eine begrenzte Menge von Informationen an das Servlet übergeben kann. Aus diesem Grund erfolgt die Informationsübergabe über Post-Anfragen. Zur Verarbeitung dieser beiden Anfragearten, besitzt das Servlet zwei Methoden „doGet“ und „doPost“. Da das Programm keine Unterscheidung zwischen beiden Anfragearten benötigt und da nur Post-Anfragen ankommen, ruft die Methode doPost die Methode doGet mit Übergabe aller Parameter auf. Dadurch werden beide Anfragearten gleich behandelt. Die beiden Parameter HttpServletRequest und HttpServletResponse der Methoden dienen hierbei zur Kommunikation zwischen Server und Client. So steht in HttpServletRequest der Anfrageheader mit allen Informationen der HTML-Seite. In HttpServletResponse wiederum sind die Informationen gespeichert, die mit der Antwort des Servers an den Client zusammenhängen. Diese beiden Parameter werden an diversen Stellen des Servlets benutzt.

Bevor jedoch die Informationen einer Anfrage bearbeitet werden können, wird der Anfrage-Header eines Users (eines Lernenden) zuerst zwischengespeichert und erst dann bearbeitet, wenn das Servlet wieder frei ist. Dies geschieht aus diesem Grund, da das Servlet nicht mehrere Anfragen, die gleichzeitig ankommen - was im Internet durchaus möglich ist - bearbeiten kann. Um keine dieser Anfragen zu verlieren bzw. zu überschreiben, werden diese nun zwischengespeichert und solange verzögert, bis der kritische Bereich des Servlets wieder frei ist. Wenn die Anfrage in den kritischen Bereich eintreten kann, wird dann zuerst festgestellt, welche Methode bei der Bearbeitung der Informationen benutzt werden muss. Der Wert des Klassenfelds „methode“ wurde dabei von der Datei Komponenten.xsl (siehe oben) gesetzt. Oder der Wert steht in einem temporären Klassenfelds des Sitzungs-Tracking, worauf weiter unten näher eingegangen wird.

Bevor jedoch das Klassenfeld methode zum Tragen kommt, werden zuerst alle Cookies eingelesen, die von der HTML-Seite mit übergeben wurden. Der CookieString wurde dabei vom Javascript kurse.js (siehe Kapitel 4) zusammengesetzt und als HIDDEN-Information mit auf der HTML-Seite gespeichert. Nachdem auch diese Informationen eingelesen wurden, wird der Name der

Lerneinheit festgestellt und im Klassenfeld dateiname gespeichert. Dieses Klassenfeld wird bei der Erzeugung des korrekten physikalischen Link der nachfolgenden Lerneinheit benötigt (siehe weiter unten).

Da das Servlet nur solange existiert, bis es die einzelne Lerneinheit abgearbeitet hat, benötigt es einen speziellen Mechanismus, um Informationen von einem Lebenszyklus des Servlets zu nächsten Lebenszyklus zu übermitteln. Hierzu bieten Servlets dem Programmierer ein leistungsstarkes Konzept an mit dem Namen Sitzungstracking. Dieses Sitzungstracking ermöglicht es Informationen über eine beliebige Anzahl von Aufrufen von HTML-Seiten (bzw. xml-Seiten) weiterzugeben. Erst wenn das Browserfenster geschlossen wird, verliert die Sitzung ihre Gültigkeit und die Informationen für diese Sitzung sind nicht mehr erreichbar. Von diesem Konzept wird vor allem bei der Kurs- und Lektionsteuerung (Autorensteuerung) Gebrauch gemacht, wenn es darum geht Informationen der Autorensteuerung, die sich in den Steuerungskomponenten der Startseiten befinden, den nächsten Lebenszyklen des Servlets zu übermitteln.

Aber auch Informationen hinsichtlich nachfragender Lerneinheiten werden im Sitzungstracking gespeichert (siehe hierzu Diplomarbeit Nr. 2076 [Conradt]). Der entsprechende Zugriffsschlüssel auf die Sitzung wird dabei als Cookie beim Client gespeichert.

Eine Instanz dieses Sitzungstracking wird am Anfang eines Kurses angelegt. Bei den folgenden Aufrufen des Servlets wird dann festgestellt, ob es schon ein Sitzungstracking gibt oder ob eins angelegt werden muss. Dadurch ist gewährleistet, dass es nur ein Sitzungstracking pro Kurs gibt.

Nachdem das Sitzungstracking nötigenfalls initialisiert wurde, wird nun im weiteren Verlauf des Servlets anhand des Wertes des Klassenfelds methode unterschieden, welcher Programmcode als nächstes ausgeführt werden soll. Dies ist notwendig, da die Anfragen verschiedener HTML-Seiten verschieden behandelt werden müssen. So wird bei einer Kursstartseite die Informationen der Komponenten Kurs und KursBewertung ausgelesen und im Sitzungstracking im kurs-Objekt abgespeichert. Gleiches gilt für die Lektionstartseite. Dagegen wird eine normale Lerneinheit lediglich von der Methode LEauswertung ausgewertet. Im folgenden sind die Werte angegeben, die das Klassenfeld methode annehmen kann:

- Kursstart: Bei jedem Kursanfang (Aufruf der Datei KursStartseite.xml eines Kurses) werden zuerst die Informationen der Komponenten Kurs und

KursBewertung im Sitzungstracking gespeichert. Danach wird die Methode feedbackKursstart aufgerufen, die die Feedbackseite für den Lernenden zusammenstellt.

- Lektionstart: Es werden im Prinzip dieselben Operation durchgeführt wie bei Kursstart, diesmal aber auf der Lektionebene. Anschließend wird wiederum eine Seite mit einer Rückmeldung für den Lernenden zusammengestellt. Diesmal durch die Methode feedbackLektionstart(). Zur genauen Funktionsweise siehe weiter unten.
- Weiter: Bei weiter wird die Methode weiter() aufgerufen.
- KursBewertung: Wenn das Klassenfeld methode auf diesem Wert steht, dann wird die Methode Kursauswertung() aufgerufen.
- LKBewertung, LEauswertung: Entsprechend werden hier die Methoden LKBewertung() bzw. LEauswertung() aufgerufen.
- Bei den übrigen Werten des Klassenfelds methode werden Methoden aufgerufen, die für die Bearbeitung einer nachfragenden Lerneinheit zuständig und für die auf die Diplomarbeit Nr. 2076 [Conradt] verwiesen sei.

Da es sich bei dem Autorensystem um ein offenes und erweiterbares System handelt, kann das Servlet selbst nicht wissen, wie die Informationen der einzelnen Komponenten einer Lerneinheit nun verarbeitet werden müssen. Aus diesem Grund müssen die entsprechenden Auswertungsklassen der Komponenten für diese Arbeit herangezogen werden. Dies geschieht ähnlich zum Frontend mit Hilfe der Polymorphie und die Aufgabe der Methode KomponentenErzeugen() ist es, für jede einzelne Komponente der Lerneinheit eine Instanz der entsprechenden Auswertungsklasse zu erzeugen.

Wie man hier sieht, benötigt also jede Komponente, die in das Autorensystem eingebaut wird, mehrere Dateien. Dies sind die Dateien für das Frontend, für das Servlet (Auswertungsklassen) und die xsl-Dateien, die die Informationen umwandeln.

Methode KomponentenErzeugen()

Da die Instanzen der Komponenten der Lerneinheiten für die weitere Verarbeitung der Informationen in den meisten Methoden des Servlets benötigt werden, wird die

Methode KomponentenErzeugen vor diesen Methoden aufgerufen, um die Instanzen der Komponenten rechtzeitig anzulegen. Beim Anlegen der einzelnen Komponenten sind dabei ein paar Punkte zu beachten, die im folgenden beschrieben werden. Zuerst holt sich die Methode KomponentenErzeugen() alle Parameternamen aus dem Anfrageheader und die dazugehörigen Werte. All dies wird in einem Enumeration-Klassenfeld gespeichert. Der Nachteil an einem Enumeration-Feld ist, dass die Parameter in einer willkürlichen Reihenfolge gespeichert werden. Die erste Aufgabe der Methode ist es also zuerst die Informationen zu sortieren, bevor im zweiten Schritt die entsprechende Komponente mit Hilfe des Reflection-Mechanismus angelegt wird. Hierzu wird ein kleiner Trick angewendet.
Bei jeder Komponente gibt es 2 Parameter, die immer gleich heißen. Dies sind die Parameter klasse und position. Über den Aufruf *request.getParameterValues("klasse")* wird die Reihenfolge nicht verändert und es werden alle Werte dieses Parameters hintereinander gespeichert. Dabei handelt es sich aber um die Reihenfolge, wie die einzelnen Komponenten aufeinander folgen, was genau benötigt wird. Entsprechendes gilt für den Parameter position. So kann jeder Komponente ihre Position zugeordnet werden und genau diese Position steht vor jedem Parameter außer klasse und position selbst. Durch eine String-Analyse kann nun jede Information zu ihrer Komponente zugeordnet werden, wobei die Information an sich in einer beliebigen Reihenfolge vorliegen. Die weitere Sortierung erfolgt in den Instanzen der Komponenten selbst (siehe Kapitel 3.2.3 „Auswertungsklassen“). Die Aufgabe der Methode KomponentenErzeugen() ist es lediglich, die einzelnen Parameter den einzelnen Komponenten zuzuordnen und bei ihrer Erzeugung zu übergeben.
Nachdem die Vorsortierung erfolgt ist und die Parameternamen mit ihren Parameterwerten in einem zweidimensionalen Array gespeichert sind, wird die Komponente mit Hilfe des Reflection-Mechanismus und dem Namen aus dem Klassenfeld klasse erzeugt. Als Ergebnis hat die Methode KomponentenErzeugen einen Array mit den einzelnen Komponenten angelegt, der für die weiteren Methoden des Servlets zur Verfügung steht.

Im folgenden werden nun die einzelnen Methoden betrachtet, die das Servlet benötigt um die Informationen der Lerneinheiten zu verarbeiten und den korrekten Ablauf der Autorensteuerung zu gewährleisten. Dabei kann man diese Methoden,

neben den Methoden der nachfragenden Lerneinheit (Diplomarbeit Nr. 2076 [Conradt]), in drei große Bereiche unterteilen. Dies sind die Auswertungs-Methoden, die Feedback-Methoden und die Methode linkErzeugen().

Die Auswertungsmethoden

Die Auswertungsmethoden ermöglichen es hierbei erst, die Eingaben des Lernenden zu analysieren. Für jede einzelne Ebene des Kurses gibt es dazu eine eigene Methode. Auf der untersten Ebene, der Ebene der Lerneinheit ist dies die Methode LEauswertung().

Methode LEauswertung()

Nachdem über die Methode KomponentenErzeugen() der Komponenten-Array angelegt wurde, ruft die Methode LEauswertung nacheinander die Methode auswertung() der einzelnen Komponenten auf. Diese Komponenten sind zuständig für die Auswertung der Antworten, die der Lernende geschrieben oder auf andere Weise (siehe Bildfrage) eingegeben hat. Als Ergebnis wird an das Servlet die erreichte Punktzahl zurückgegeben. Dadurch kann ermittelt werden, wie viele Punkte ein Lernender bei einer Lerneinheit erreicht hat. Diese erreichte Punktzahl wird an die Bewertungskomponente übergeben, welche immer die letzte Komponente einer Lerneinheit darstellt. Die Bewertungskomponente stellt dann fest, in welchen Bereich sich der Schüler mit der erreichten Punktzahl befindet. Dabei wird überprüft, ob der Lernende eine nachfragende Lerneinheit ausgelöst hat, was zu einem gesonderten Ablauf der nächsten Lerneinheiten führt (siehe Diplomarbeit Nr. 2076 [Conradt]).
Im Normalfall wird anschließend die Methode feedbackLE() aufgerufen, die eine Rückmeldung der Ergebnisse an den Lernenden liefert und die nächste Lerneinheit aufruft, die in der Autorensteuerung angegeben ist. Außerdem werden beim Normalfall die erreichten Punkte der Lerneinheit, sowie die maximal erreichbaren Punkte der Lerneinheit im lektion-Objekt gespeichert, welches wiederum im Sitzungstracking gespeichert ist. Diese Informationen werden dann von der Methode LKauswertung genutzt, wenn es um die Bewertung einer ganzen Lektion geht.

Methode LKauswertung()

Eine Ebene höher, d.h. auf Lektionebene wird zur Bewertung einer ganzen Lektion diese Methode aufgerufen. Diese Methode bezieht die Informationen über die vom Lernenden erreichten Punktzahlen der einzelnen Lerneinheiten aus dem lektion-Objekt des Sitzungstracking und übergibt sie an die Komponente LKBewertung, welche ebenfalls im Sitzungstracking mit gespeichert ist. Danach wird eine Feedbackseite mit Hilfe der Methode feedbackLKBewertung() zusammengebaut und an den Lernenden geschickt. Zu guter letzt wird der Besuch dieser Lektion im kurs-Objekt des Sitzungstracking gespeichert mitsamt der erreichten Punktzahl der Lektion, sowie der maximal erreichbaren Punktzahl der Lektion. Diese Informationen werden von der Methode Kursauswertung ausgewertet, wenn der Schüler das Ende des Kurses erreicht hat. Die Methode Kursauswertung() unterscheidet sich dabei von der Methode Lkauswertung() nur darin, dass das Ganze eine Ebene weiter höher abläuft und für die Seite der Rückmeldung die Methode feedbackKursBewertung() aufgerufen wird.

Die Feedback-Methoden:

Nachdem die Auswertungsmethoden die Eingaben des Lernenden mit Hilfe der Auswertungsklassen (siehe Kapitel 3.2.3 „Auswertungsklassen“) analysiert haben, müssen nun die Ergebnisse dem Lernenden mitgeteilt werden, was durch die Feedback-Methoden geschieht. Hierbei gibt es wieder für jede Ebene eine eigene Feedback-Methode.

Methode feedbackLE()

Die Methode feedbackLE() ist dabei für die Rückmeldung der untersten Ebene zuständig und wird von der Methode LEauswertung() aufgerufen. Die Rückmeldung erfolgt dabei durch das Anzeigen einer HTML-Seite, die eine Rückmeldung jeder einzelnen Komponente aufnimmt. Hierbei ist die Bewertungskomponente der Lerneinheit für eine zusammenfassende Rückmeldung der ganzen Lerneinheit zuständig. Zusätzlich besitzt jede HTML-Seite mit einer Rückmeldung eine „Weiter“-Schaltfläche, die den Lernenden zur nächsten Lerneinheit führt. Der Link hierfür

wurde zuvor in der Methode linkErzeugen() erzeugt, die als Parameter den Namen der aktuellen Lerneinheit und den Dummy-Link aus der aktuellen Bewertungskomponente bekommt. Mit Hilfe dieser zwei Informationen liefert die Methode linkErzeugen() den richtigen physikalischen Wert zurück. Zur weiteren Erläuterung siehe weiter unten. Damit hat der Lernende einen Überblick über seine Leistungen einer einzelnen Lerneinheit bekommen.

Methode feedbackLKBewertung()

Für die Rückmeldung der Leistungen einer ganzen Lektion ist hingegen die Methode feedbackLKBewertung() zuständig. Die nötigen Informationen bezieht sich diese Methode dabei nicht von den einzelnen Komponenten einer Lerneinheit, sondern von dem lektion-Objekt aus dem Sitzungs-Tracking. Hier sind in einer Liste die erreichten Punkte jeder einzelnen besuchten Lerneinheit gespeichert. Diese Informationen werden an die Feedback-Methode der LKBewertungs-Komponente geschickt, die ebenfalls im Sitzungstracking gespeichert ist. Das Erscheinungsbild der HTML-Seite unterscheidet sich dabei zu der Feedback-Seite einer einzelnen Lerneinheit darin, dass die erreichten Punkte jeder einzelnen Lerneinheit neben der Gesamtrückmeldung für die ganze Lektion angezeigt wird. Die Gesamtrückmeldung hängt dabei davon ab, wie viele Punkte der Lernende in der Lektion zusammengetragen hat und in welchen Bewertungsbereich der Bewertungskomponente der Lektion er gekommen ist.
Mit Hilfe des Dummy-Links dieses Bereichs wurde zuvor in der Methode linkErzeugen() der Link zur nächsten Lektion erzeugt und dieser ebenfalls in der HTML-Seite mitgespeichert.

Methode feedbackKursBewertung()

Für die Rückmeldeseite der höchsten Ebene schließlich ist die Methode feedbackKursBewertung() zuständig. Der wesentliche Unterschied zu den beiden anderen Methoden liegt darin, dass diese Methode die letzte Seite des Kurses zusammenstellt und deswegen die Methode linkErzeugen() nicht mehr aufrufen muss, da keine weiteren Lerneinheiten bzw. Lektionen mehr folgen. Der Aufbau der Feedbackseite selbst ähnelt dabei der Feedbackseite der Lektionebene mit dem

Unterschied, dass nun die erreichten Punkte jeder besuchten Lektion angezeigt wird und dass es keine Schaltfläche mehr gibt, die auf eine nächste Lektion führen würde.

Methode feedbackKursstart(), feedbackLektionstart() und weiter()

Neben diesen Feedback-Methoden für die einzelnen Ebenen gibt es noch drei weitere Feedback-Methoden. Diese Methoden werden dann aufgerufen, wenn der Lernende gerade eine Startseite des Kurses oder einer Lektion über die Schaltfläche „Weiter“ verlassen hat.
Der wesentliche Unterschied zu den obigen Methoden liegt darin, dass bei diesen Methoden keine Rückmeldung gegeben wird, da der Autor keinerlei Eingaben auf den Startseiten machen konnte, da diese nur aus Präsentationskomponenten bestehen. Die einzigste Aufgabe dieser Methoden ist es lediglich den Lernenden auf die nächste Lektion bzw. Lerneinheit zu leiten. Der notwendige physikalische Link wurde dabei wie immer in der Methode linkErzeugen() erzeugt.
Ähnliches gilt für die Methode weiter(), die dann vom Servlet ausgeführt wird, wenn die Lerneinheit nur aus Präsentationskomponenten besteht. Auch in diesem Fall muss keine Rückmeldung an den Lernenden erfolgen, da dieser keine Eingaben machen konnte und der Lernende muss nur auf die nächste Lerneinheit geleitet werden.

Die übrigen Feedback-Methoden beziehen sich schließlich alle auf die nachfragenden Lerneinheiten, für die wie immer auf die Diplomarbeit Nr. 2076 [Conradt] verwiesen sei.

Methode linkErzeugen()

Abschließend wird jetzt die Methode linkErzeugen() betrachtet, die in den oberen Methoden schon mehrfach genannt wurde und deren Aufgabe es ist, aus den Informationen der Bewertungskomponenten und den Informationen in den Objekten lektion und kurs den Link für die nächste Seite zu erzeugen und damit die Autorensteuerung zu ermöglichen.
Um diesen Link zu erzeugen, benötigt die Methode zwei Informationen. Dies ist zu einem der Name der aktuell aktiven Lerneinheit bzw. Lektion und zum anderen der

Dummylink der Bewertungskomponente der entsprechenden Ebene. Die richtige Ebene wird dabei dadurch festgestellt, indem abgeprüft wird, auf welchen Wert das Klassenfeld methode gerade steht. Es sind hierbei die Werte LKBewertung auf der einen Seite oder LEauswertung bzw. OFauswertung auf der anderen Seite möglich. Die Informationen bezüglich des Dummylinks und der aktuell aktiven Lerneinheit bzw. Lektion werden dann dem lektion- bzw. kurs-Objekt übergeben. In diesen Objekten, welche im Sitzungstracking gespeichert sind, sind die Informationen der Steuerungskomponenten der einzelnen Ebenen und damit die Autorensteuerung selbst gespeichert. Diese Objekte liefern als Ergebnis den Namen der nächsten Lerneinheit bzw. Lektion zurück. Hierzu schauen sie in einer Liste für die aktive Lerneinheit bzw. Lektion nach, was für ein Name bei dem entsprechenden Dummylink steht. Mit Hilfe dieses Namens kann nun der Link zusammengebaut werden. Dabei muss beachtet werden, dass aufgrund der Verzeichnisstruktur der Link auf die nächste Lektion anders zusammengebaut ist, als der Link auf die nächste Lerneinheit. Der Hauptunterschied besteht nämlich darin, dass der zurückgegebene Name der nächsten Lektion das Verzeichnis angibt, indem die Startseite der Lektion gespeichert ist, während bei der nächsten Lerneinheit unter dem zurückgegebenen Namen der Name der Datei der nächsten Lerneinheit angegeben ist. Handelt es sich bei dem zurückgegeben Namen jedoch um einen Verweis auf eine Bewertungskomponente der Lektion- bzw. Kursebene so wird dieser nochmals gesondert behandelt. So wird in diesen Fällen auf das Servlet selbst verwiesen, da die Rückmeldung dieser beiden Ebenen durch das Servlet automatisch geregelt wird.

Neben dem physikalischen Link liefert die Methode linkErzeugen() als weiteren Parameter, den Wert der Methode zurück der beim nächsten Aufruf des Servlets für die HTML-Seite benutzt werden muss. Damit ist nun auch geklärt, wie die im Frontend eingegebene Autorensteuerung im Backend ausgeführt wird. Im nächsten Abschnitt wird darauf eingegangen, wie die Auswertungsklassen der einzelnen Komponenten funktionieren und wie sie die vom Servlet übergebenen Informationen verwalten.

3.2.3 Auswertungsklassen

Dabei sind für die Autorensteuerung vor allem die Klassen Kurs und Lektion wichtig, die das Pendant zu den gleichnamigen Klassen in Frontend bilden. Die Aufgaben dieser Klassen ist es hierbei die Informationen der Autorensteuerung aufzunehmen und im Zusammenspiel mit der Methode linkErzeugen() den Name der nächsten Lektion bzw. Lerneinheit zurückzuliefern. Aus diesem Grund hat das Servlet nach Auswertung der Startseite des Kurses mit Hilfe der Informationen der Steuerungskomponente eine Instanz der Klasse Kurs angelegt und diese im Sitzungstracking gespeichert. Ähnliches gilt hierbei für die Startseiten einer Lektion. Dadurch dass die Objekte lektion und kurs im Sitzungstracking gespeichert sind, hat die Methode linkErzeugen() erst die Möglichkeit auf die Autorensteuerung zuzugreifen. Die Objekte bleiben dabei solange erhalten, bis der Lernende zu einer anderen Lektion geleitet wird und demnach das lektion-Objekt durch die Autorensteuerungs-Informationen der neuen Lektion ausgetauscht wird, und im Falle des kurs-Objekts bleibt dieses solange erhalten, bis der Browser geschlossen wird. Im folgenden sei wiederum nur die Klasse Kurs betrachtet, da sich die Klasse Lektion wiederum nur darin unterscheidet, dass sie für eine Ebene tiefer zuständig ist.

3.2.3.1 Klasse Kurs

Der Konstruktor der Klasse Kurs bekommt vom Servlet drei Parameter übergeben. Dies sind der Parameter methode, der von einigen Komponenten bzw. Auswertungsklassen benötigt wird, die Informationen über die eventuell vorhandenen Cookies der HTML-Seite und die Informationen, die vom Servlet für diese Komponente zusammengestellt wurden. Wie oben schon erwähnt wurde, sind diese Informationen unsortiert, da das Servlet aufgrund des Prinzips des offenen Autorensystems nichts über die Reihenfolge der Informationen wissen kann. Demnach müssen diese Informationen erst einmal sortiert werden, was die Hauptaufgabe des Konstruktors ist. Nachdem dies geschehen ist, steht die Instanz der Komponente Kurs dem Servlet zur Verfügung. Diese wird bekanntlich im Sitzungstracking des Servlets gespeichert.
Dabei wird in dieser Instanz die Komponente Kursbewertung mit abgespeichert, die dann benötigt wird, wenn der Autor die letzte Seite des Kurses mit der Rückmeldung

über den ganzen Kurs erreicht. Bei den Informationen, die nun in der Komponente Kurs gespeichert sind, handelt es sich um all die Ablaufinformationen der Lektionen auf der Kursebene, also um die Informationen der Autorensteuerung auf dieser Ebene, die der Autor im Frontend zusammengestellt hat. Des weiteren sind die Informationen über die eventuell vorhandenen Zyklusdurchbrüche auf den einzelnen Lektionen gespeichert. Die Aufgabe der Komponente Kurs ist es nun, diese Informationen dem Servlet zur Verfügung zu stellen, wenn dieses sie braucht.

Methoden der Ablaufsteuerung

So wird in der Methode linkErzeugen() des Servlets (siehe auch oben) zuerst die Position der aktuellen Lektion in der entsprechende Liste der Komponente Kurs festgestellt. Nachdem dies festgestellt wurde, kann die Methode linkErzeugen() über die Methoden getNachfolgerfalsch(), getNachfolgerundefiniert(), getNachfolgerrichtig und getNachfolgerweiter() der Komponente Kurs den Namen der nachfolgenden Lektion bekommen, je nachdem um welche Verzweigungsart es sich handelt. Die Methoden der Komponente Kurs greifen dabei auf die entsprechenden Listen zu, in denen die richtige Information gespeichert sind.
Anschließend erhöht die Methode linkErzeugen() die Anzahl der Besuche auf der nachfolgenden Lektion um eins, was in der Komponente Kurs gespeichert wird (Methode setBesuche()). Hierzu führt die Komponente eine Besuchsliste, in der für jede Lektion die Anzahl der Besuche angegeben ist. Diese Liste wird im übrigen am Anfang des Kurses über die Methode loescheBesuche() auf null initialisiert, da am Anfang des Kurses ja noch keine Lektion besucht wurde.
Die nächste Aufgabe der Methode linkErzeugen() ist es anschließend zu überprüfen, ob auf der nächsten Lektion ein Zyklusdurchbruch eingetreten ist, oder anders ausgedrückt, ob die maximal erlaubte Besuchsanzahl auf der nächsten Lektion überschritten ist, was dadurch geschieht, dass die Methode linkErzeugen() die Methode zyklusdurchbruch() der Komponente Kurs aufruft. Diese vergleicht die aktuelle Besuchsanzahl der nächsten Lektion mit der maximal erlaubten Besuchsanzahl und gibt dementsprechend ein booleschen Wert zurück. Sollte ein Zyklusdurchbruch eingetreten sein, benötigt die Methode linkErzeugen() weitere Informationen, wie die vom Autor eingegebene Rückmeldung und den Namen der

nachfolgenden Lektion für diesen Fall (Methoden getMaxBesucheFeedback() und getNachfolgerzyklus()).

Methode für die Kursauswertung und für die Rückmeldung

Die übrigen Methoden der Klasse Kurs werden benötigt, wenn es darum geht den ganzen Kurs auszuwerten und eine Rückmeldung auf der letzten Seite des Kurses auszugeben. Da diese Rückmeldung (siehe oben) so aufgebaut ist, dass sie für jede einzelne Lektion die erreichten und die maximalen erreichbaren Punkte ausgibt, müssen diese Informationen während des Kursablaufs in der Komponente Kurs zwischengespeichert werden, was durch einen Aufruf der Methode setBesuchteSeite() nach jeder Auswertung einer ganzen Lektion geschieht. Dabei sorgt die Komponente Kurs intern dafür, dass die Lektionen, die sich innerhalb eines Zyklus befinden automatisch mitsamt ihren Punkten gelöscht werden. Dies ist die Aufgabe der Methode loescheAblaufListe(), die durch die Listen der besuchten Lektionen geht und prüft, ob die Lektion schon einmal vorkommt. Wenn ja wurde ein Zyklus durchschritten und die Liste der besuchten Lektionen wird ab dem ersten Auftreten dieser Lektion gelöscht. Bei der Auswertung eines Kurses schließlich werden all diese Informationen über die erreichten und die maximal erreichbaren Punkte aus der Komponente Kurs geholt (Methode getPunkte(), getMaxPunkte()) und der Kursbewertungs-Komponente, die ebenfalls in der Komponente Kurs zwischengespeichert wurde, übergeben. Für die abschließende Rückmeldung werden dann über die Methoden getNameListe(), getPunkteListe() und getMaxPunkteListe() die Informationen über die einzelnen Lektionen geholt und der Rest der Rückmeldung ist die Aufgabe der Kursbewertungs-Komponente, die im Kapitel 3.2.3.2 besprochen wird.
All dies gilt, wie immer sinngemäß auch für die Lektion-Ebene und die Komponente lektion, die für die Verwaltung der Informationen der Autorensteuerung auf dieser Ebene zuständig ist.

3.2.3.2 Bewertungskomponenten (Klasse LEBewertungAuswertung, LKBewertungAuswertung und KursBewertungAuswertung)

Der Konstruktor der drei Bewertungskomponenten hat wie bei den Klassen Kurs und Lektion dieselbe Aufgabe und sortiert alle Informationen, die ihm vom Servlet übergeben wurden. Dabei handelt es um die Informationen über die einzelnen Auswertungsbereiche, die der Autor im Frontend zusammengestellt hat. Die Hauptfunktionalität dieser Klasse besteht darin, die Leistungen der Lernenden auf der entsprechenden Ebene auszuwerten und den Verzweigungstyp (falsch, undefiniert, richtig, weiter) des erreichten Auswertungsbereich zurückzugeben. Hierzu bekommt die Methode auswertung() von den einzelnen Auswertungsmethoden des Servlets (Methode LEauswertung(), LKauswertung(), Kursauswertung()) die erreichten und maximal erreichbaren Punktzahlen der Lerneinheit, bzw. der Lektion oder des Kurses übergeben. Anhand der Prozentwerte der einzelnen Bereiche, stellt die Methode dann fest, in welchen Bereich der Lernende gekommen ist. Dieser Bereich wird intern in der Komponente gespeichert. Nachdem die Leistung des Lernenden bewertet wurde, ruft das Servlet anschließend die für die einzelnen Ebenen zuständige Feedback-Methode auf. Diese Methoden (siehe oben) stellen die einzelnen Feedbackseiten zusammen und holen sich hierfür aus den Bewertungskomponenten mit der Methode getLink() den Dummylink, den die Methode linkErzeugen() für die Zusammenstellung des physikalischen Links benötigt. Der Dummylink hängt dabei nur davon ab, in welchen Auswertungsbereich der Lernende anhand seiner Leistungen gekommen ist. Abschließend benötigt die Feedbackmethode die Rückmeldung der Bewertungskomponente, die die ganze Einheit (Lerneinheit, Lektion, Kurs) bewertet. Dies wird durch den Aufruf der Feedback-Methode der Bewertungskomponente erreicht und hängt wiederum nur vom erreichten Auswertungsbereich ab.

Die Unterschiede zwischen den einzelnen Bewertungskomponente bestehen dabei nur darin, dass die Bewertungskomponente einer einzelnen Lerneinheit weitere Funktionalität bezüglich des Konzepts der nachfragenden Lerneinheiten (siehe Diplomarbeit Nr. 2076 [Conradt]) besitzt und das in der Kursbewertungs-Komponente die Methode getLink() wegfällt, da sich diese Bewertungskomponente bekanntlich auf der höchsten Ebene befindet.

3.2.3.3 Präsentationskomponenten und Fragekomponenten

Was nun noch fehlt, sind die Auswertungsklassen der einzelnen Präsentations- bzw. Fragekomponenten, die im Kapitel 4 vorgestellt werden. Die Klassen des Backends seien aber wiederum vorgezogen, da ihre Funktionalität leicht zu verstehen ist und um das Backend in diesem Kapitel vollständig abzuhandeln. Die Auswertungsklassen der JPG-Bild-Komponente, GIF-Bild-Komponente, SVG-Bild-Komponente und Sound-Komponente sind dabei ähnlich zu deren xsl-Dateien recht einfach aufgebaut. Im Konstruktor dieser Klassen werden wie immer zuerst die Informationen sortiert, die vom Servlet übertragen wurden. Es handelt sich dabei lediglich um Informationen zur richtigen Anzeige der Komponente.
Als einzigste weitere Methode besitzen diese Klassen schließlich nur noch die Methode show(), die vom Servlet dann benötigt wird, wenn die entsprechende Präsentationskomponente Bestandteil einer nachfragenden Lerneinheit ist. So dürfen im Falle einer nachfragenden Lerneinheit die Präsentationskomponenten nur dann angezeigt werden, wenn auch die nachfragende Lerneinheit angezeigt wird. Zum Konzept der nachfragenden Lerneinheit sei wie immer auf die Diplomarbeit Nr. 2076 [Conradt] verwiesen.
Die Auswertungsklasse der SVG-Frage hingegen besitzt weitaus mehr Funktionalität, die nun im einzelnen beschrieben wird. Der Kontruktor hat nun neben der Sortierung der einzelnen Informationen auch die Aufgabe die übergebenen Cookies zu speichern. Bei den Cookies, die für die Bildfrage bestimmt sind, handelt es sich dabei um die Eingaben, die der Lernende durch die Manipulation des Bildes gegeben hat. Dabei wurden die Werte der Cookies entweder auf false oder true gesetzt. Sollten keine Cookies vorhanden sein, so werden automatisch alle Werte der Cookies bzw. der Eingaben auf false gesetzt. Die wichtigsten Methoden einer jeden Frage und damit auch der SVG-Frage sind dabei die Methoden auswertung() und feedback(), die vom Servlet in der Methode LEauswertung() bzw. in den einzelnen Feedback-Methoden aufgerufen werden. Die Methode auswertung() hat ähnlich zu der gleichnamigen Methode der Bewertungskomponente die Aufgabe, die Leistung des Lernenden zu beurteilen. Wobei hier nur die Leistung einer einzelnen Bildfrage ausgewertet wird. Um dies zu erreichen vergleicht die Methode die Antworten, sprich die gesetzten Cookies des Lernenden mit den einzelnen Auswertungsvorschriften der Bewertungsbereiche, die im Frontend erstellt wurden. Zum allgemeinen Konzept

der Bildfrage sei auf das Kapitel 4 verwiesen. Zum Vergleich benutzt die Methode auswertung() hierbei die Methoden logikparser(), die für die Aufschlüsselung der Auswertungsvorschrift zuständig ist, und die Methode vergleichsauswertung(), die einen einzelnen Bestandteil der Auswertungsvorschrift auswertet. Je nachdem welche Auswertungsvorschrift der Lernende ausgelöst hat, werden die entsprechende Punkte des dazugehörigen Bereichs an die Methode LEauswertung() zurückgegeben. Auf diese Weise kann diese Methode die Punkte der einzelnen Fragekomponenten zusammenrechnen und der oben beschriebenen Bewertungskomponente zukommen lassen. Neben der Methode auswertung() benötigt jede Fragekomponente auch eine Methode feedback(), die eine Rückmeldung über die Leistungen des Lernenden für die einzelne Frage erstellt. Diese Rückmeldung wird dann in die Feedbackseite der Lerneinheit, die vom Servlet zusammengestellt wird, eingebaut.

Die übrigen Methoden der SVG-Frage dienen schließlich dem Konzept der nachfragenden Lerneinheit. So kann jede Frage eine oder mehrere nachfragende Lerneinheiten besitzen, die dann ausgelöst werden, wenn der Lernende in einen entsprechenden Bewertungsbereich gekommen ist. Die Informationen zu diesen nachfragenden Lerneinheiten werden hierbei von der Klasse SVGFrageAuswertung mit verwaltet und bei Bedarf dem Servlet zur Verfügung gestellt (Methode getNLKomponente()). Da beim Konzept der nachfragenden Lerneinheiten, bei richtiger Beantwortung dieser Frage, die Originalfrage wieder gestellt wird, müssen die Werte dieser Originalfrage bis zum zweiten Anzeigen der Originalfrage gespeichert werden, was in einem entsprechenden Klassenfeld des Sitzungstracking vom Servlet aus gemacht wird. Die nötigen Informationen hierzu werden durch die Methode makeOF() der Klasse SVGFrageAuswertung zur Verfügung gestellt. Damit sind die wichtigsten Methoden des Konzepts der nachfragenden Lerneinheiten erklärt. Zur genaueren Darstellung der nachfragenden Lerneinheiten sei wiederum auf die Diplomarbeit Nr. 2076 [Conradt] verwiesen.

Im nächsten Kapitel wird nun der Kurs beschrieben, der mit Hilfe des Autorensystems zusammengestellt wurde. Des weiteren die Komponenten, die für diesen Kurs entwickelt wurden und die das Prinzip des offenen Autorensystems nochmals veranschaulichen.

4. Lernprogramm zur Steigerung der Wahrnehmungsleistung sehbehinderter Kinder

Mit Hilfe des entstandenen Autorensystems wurde nun ein Lernprogramm erstellt, welches der Steigerung der Wahrnehmungsleistung sehbehinderter Kinder dienen soll. Das Lernprogramm geht dabei auf das in der Studienarbeit Nr. 1655 [Mühlbradt] und Diplomarbeit Nr. 1637 [Mühlbradt2] von Frau Mühlbradt entwickelte Lernprogramm zurück. Frau Mühlbradt bezieht sich dabei selbst wiederum auf das von Marianne Frostig entwickelte Programm der visuellen Wahrnehmungsförderung. [Frostig]
Im Bereich der visuellen Wahrnehmungsförderung werden dabei fünf Bereiche unterschieden, die jetzt kurz vorgestellt werden. [Mühlbradt]

Visuomotorische Koordination

Hierunter versteht man die Fähigkeit, das Sehen mit der Bewegung eines Gegenstands zu koordinieren. D.h. das Auge lenkt die Hand, wenn sie etwas greifen oder fangen soll. Gleiches gilt für die Füße, mit Hilfe derer automatisch die Position verändert wird, wenn sich die Person in einer ungünstigen Lage befindet, um z.B. einen Ball zu fangen.

Im Rahmen des Lernprogramms konnten dabei mit Hilfe der Bildfragen nur Aufgaben abgedeckt werden, die unter den Bereich der Feinmotorik fallen, da die Grobmotorik, wie das Fangen eines Balles oder andere sportliche Betätigungen auf dem Computer nicht nachzubilden sind. Die Feinmotorik bezieht sich dagegen vor allem auf die Bewegungen der Hand und der Finger, wozu am Computer Aufgaben gestellt werden, die mit der Maus zu lösen sind.

Figur-Grund-Wahrnehmung

Der zweite Bereich der visuellen Wahrnehmungsförderung ist der Bereich der Figur-Grund-Wahrnehmung. Hierzu sei ein kurzer Einblick in die Funktionsweise des menschlichen Gehirns gegeben. Das menschliche Gehirn ist so aufgebaut, dass es

aus den verschiedenen einströmenden Reizen, die über die menschlichen Sinne ankommen, einen bestimmten bzw. mehrere bestimmte Reize auswählen kann. Diese Reize werden dann zum Zentrum der Aufmerksamkeit, was auch als Figur bezeichnet wird. Die restlichen Reize hingegen werden zum Grund. Beispielhaft kann man sich das vorstellen, das sich eine Person auf einen bestimmten Bereich eines Bildes konzentriert, der dann automatisch zur Figur wird, während der Rest des Bildes zum Grund wird. Dabei gilt, dass ein Gegenstand bzw. ein Bereich eines Bildes nur in Beziehung zu seinem dazugehörigen Grund genau wahrgenommen werden kann.

Ein Kind mit Problemen im Bereich der visuellen Wahrnehmungsförderung hat nun Schwierigkeiten, sich auf einen einströmenden Reiz zu konzentrieren und diesen zur Figur zu machen. Und andersherum kann dies bedeuten, das das Kind sich schwer von einem Reiz abwenden kann, wenn dieser Reiz die Konzentration des Kindes fesselt.

Bei den Aufgaben zur Figur-Grund-Wahrnehmung soll nun erreicht werden, dass sich das Kind auf den wichtigsten einströmenden Reiz konzentriert.

Wahrnehmungskonstanz

Mit der Wahrnehmungskonstanz ist gemeint, dass eine Person in der Lage ist, bestimmte Eigenschaften eines Gegenstandes wie seine Form, Lage und Größe korrekt wahrzunehmen. So erkennt eine Person mit einer normalen Wahrnehmungskonstanz, dass ein Gegenstand, welchen er aus unterschiedlichen Entfernungen sieht, trotzdem die gleiche Größe besitzt. Ähnliches gilt für die Helligkeitskonstanz, wobei hier eine Person z.B. bei einem weißen Papier auch bei dunklen Lichtverhältnissen die Farbe weiß erkennt. Entsprechend kann man sich bei der Formkonstanz vorstellen, dass eine Person mit einer normalen Wahrnehmungskonstanz in der Lage ist, gleichartige Formen zu erkennen, auch wenn diese in verschiedenen Größen dargestellt sind.

In dem erstellten Lernprogramm werden beispielhaft die Form- und Größenkonstanz anhand von drei Aufgaben getestet.

Wahrnehmung der Raumlage

Der nächste Bereich der visuellen Wahrnehmungsförderung ist der Bereich der Wahrnehmung der Raumlage. Dabei ist gemeint, dass eine Person die wahrnehmenden Gegenstände mit den Begriffen hinter, vor, über, rechts und links im Bezug zu sich selbst bezeichnen kann. Kinder, die Probleme in diesem Bereich haben, können die wahrnehmenden Gegenstände nicht richtig zueinander und zu sich selbst zuordnen, weswegen sie bei Bewegungen unsicher und ungeschickt wirken können. Außerdem fällt in diesen Bereich, ob das Kind die Raumlage eines Gegenstands erkennen und mit dem richtigen Begriff bezeichnen kann.

Wahrnehmung räumlicher Beziehungen

Bei der Wahrnehmung räumlicher Beziehungen geht es schließlich darum, ob das Kind den räumlichen Bezug zweier Gegenstände zueinander richtig einordnen kann. Mit anderen Worten muss das Kind feststellen können, ob beispielsweise ein Gegenstand hinter einem anderen Gegenstand ist oder in einer anderen räumlicher Beziehung zu ihm steht.

Im entwickelten Lernprogramm sollen nun anhand mehrerer Aufgaben festgestellt werden, ob der Lernende, in diesem Fall das Kind, Probleme in einem dieser Bereiche hat und es soll dahingehend gefördert werden, dass es weniger Probleme bei der visuellen Wahrnehmung besitzt. Im Gegensatz zu den Arbeiten von Frau Mühlbradt wurde bei der Entwicklung der Bildfrage SVG benutzt, da dieses Vektorgrafikformat einige Vorteile besitzt. Die Vorteile liegen darin, dass SVG auf XML basiert, welches als Dateiformat im ganzen Autorensystem verwendet wird. Des weiteren wurde SVG dahingehend mit entwickelt, dass es besonders für das Internet geeignet ist. Und der dritte Vorteil liegt darin, dass man mit Hilfe von Javascript, welches in die SVG-Datei eingebettet ist, die Bilder leicht manipulieren kann, was bedeutet das das Kind z.B. die einzelnen Grafikelemente verschieben, löschen oder neue hinzufügen kann. Durch dieses Manipulieren ist es erst möglich, das dem Kind Bildaufgaben gestellt werden, die es mit der Maus lösen kann.
Das Lernprogramm besteht dabei aus fünf Lektionen mit jeweils drei Lerneinheiten, wobei jede Lektion einen der fünf visuellen Wahrnehmungsbereiche abdeckt. Das

Lernprogramm wurde dabei für den Internet Explorer geschrieben, da dieser den Vorteil bietet, dass es leicht möglich ist, Sounddateien von Javascript aus abspielen zu können (siehe hierzu weiter unten). Dementsprechend wurden die Javascripte für den Internet Explorer angepasst, wobei es hier möglich ist, diese leicht für den Netscape-Browser abzuändern. Im weiteren werden nun die einzelnen Lektionen mit den Bildfragen vorgestellt und danach wird die technische Umsetzung der einzelnen Bildfragen erklärt, sowie das Konzept des Zusammenspiels der Bildfragen mit dem Autorensystem. Bei den einzelnen Bildfragen wurden hierbei teilweise andere Bilder benutzt, die aber auch aus dem visuellen Arbeitsheften des Marianne-Frostig-Programms stammen. [Frostig]

4.1 Aufbau des Lernprogramms

In der ersten Lektion werden die Fähigkeiten des Kindes im Bereich der visuomotorischen Koordination getestet.

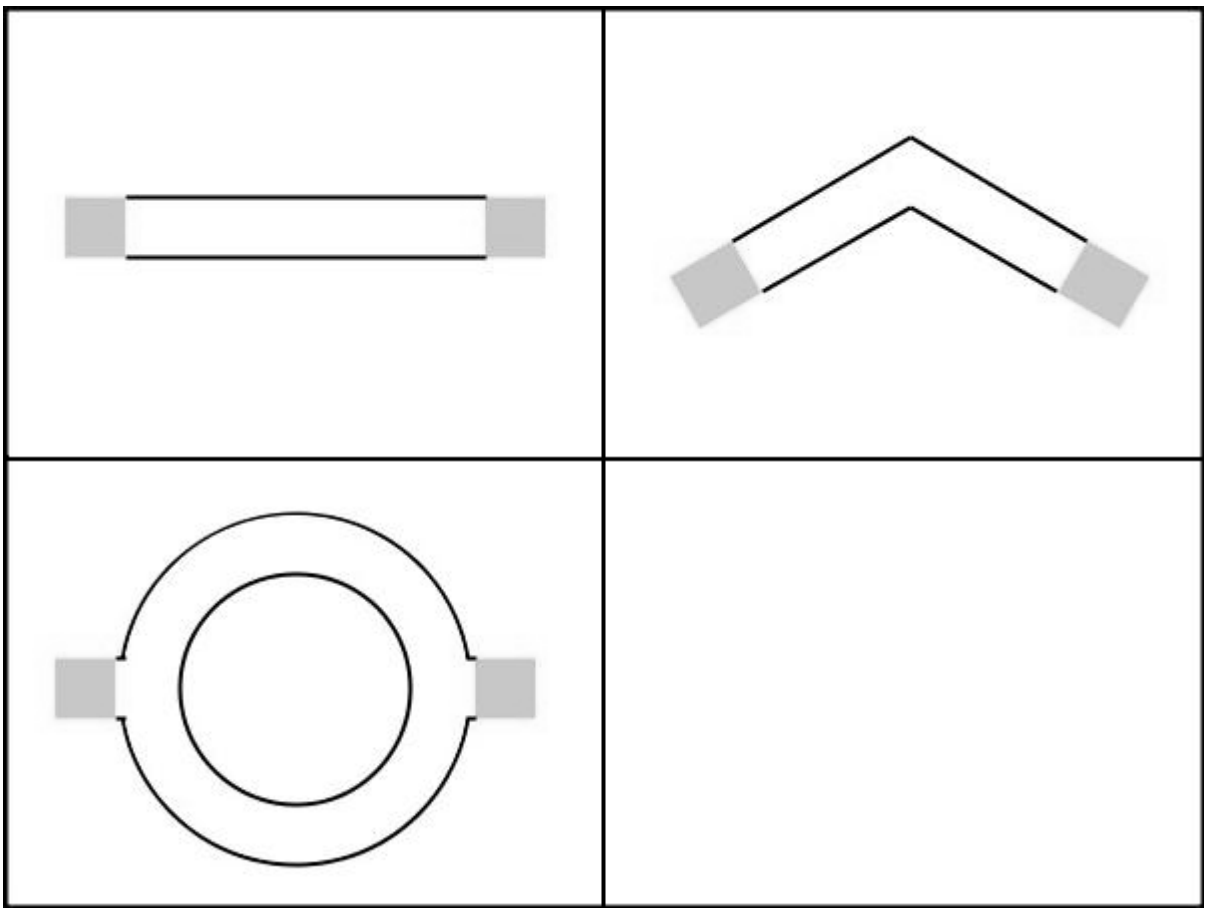

Abbildung 12: Aufgaben der visuomotorischen Koordination

Hierzu werden ihm die drei Bildfragen gestellt, die in der Abbildung 12 zu sehen sind. Die Aufgabe des Kindes liegt darin mit Hilfe der Maus eine Linie von einem grünen Bereich zum anderen zu ziehen, ohne dabei die Grenzlinien zu überschreiten. Es ist also eine gewisse Fingerfertigkeit notwendig, um die Maus mit der Hand korrekt zu führen, was in den Bereich der Feinmotorik fällt. Die Aufgaben werden dabei stufenweise schwerer, wie es im übrigen in jeder Lektion ist, um die Leistung des Kindes abstufen zu können.

In der nächsten Lektion werden anschließend Aufgaben der Figur-Grund-Wahrnehmung gestellt.

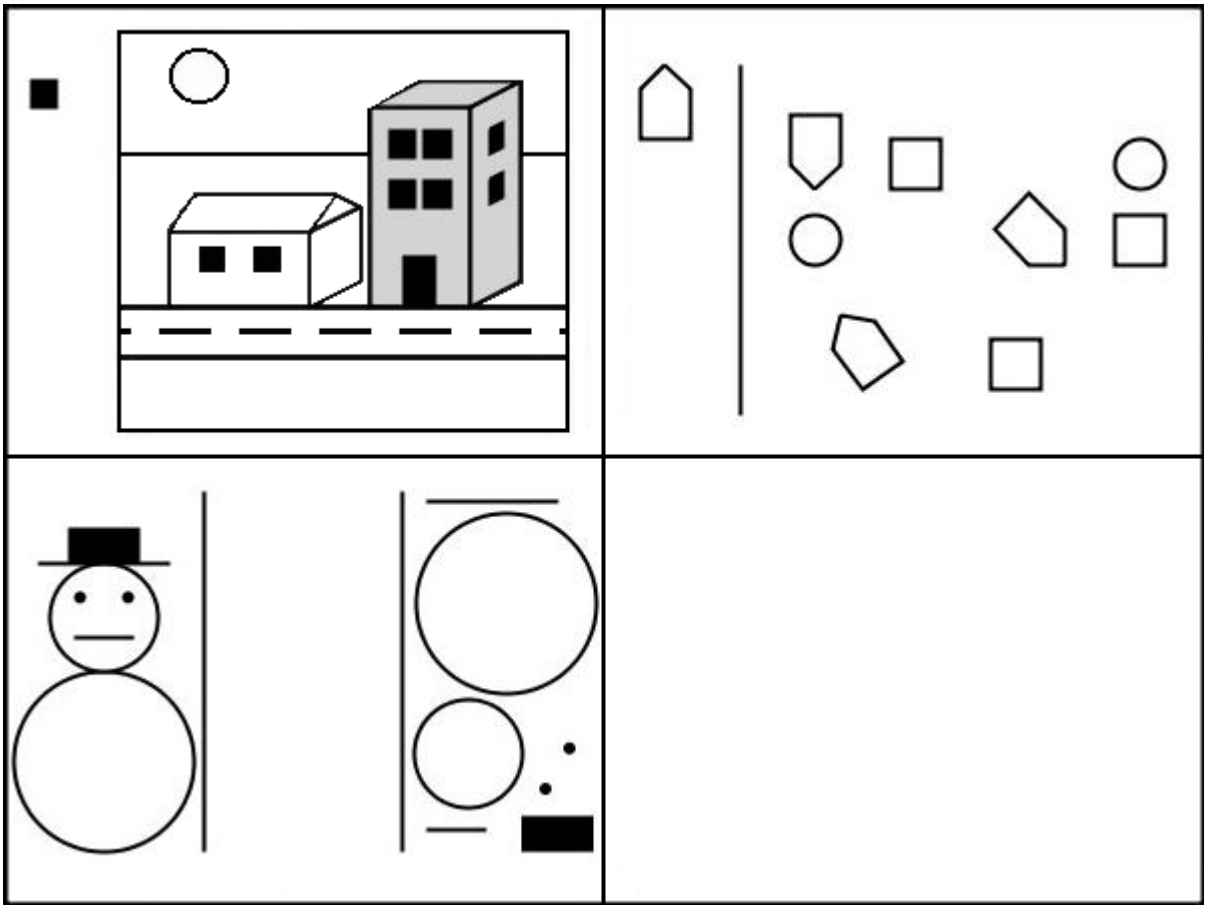

Abbildung 13: Aufgaben zur Figur-Grund-Wahrnehmung

Die erste Bildfrage zeigt hierbei ein Bild mit zwei Häuser in einer abendlichen Landschaft. Das Kind hat also viel zu sehen, es muss sich aber darauf konzentrieren, alle Quadrate im Bild zu finden, von denen eins links angezeigt wird. Bei der darauffolgenden Aufgabe, erhöht sich die Schwierigkeit dahingehend, dass die gesuchte Form nun auch gekippt dargestellt ist. Und in der letzten Bildfrage muss

das Kind den Schneemann zusammensetzen. Damit das Ganze für das Kind nicht zu langweilig wird, gibt es bei manchen Bilder eine Belohnung, wenn das Kind etwas richtig gemacht hat. Dies kann eine kleine Sounddatei sein, die dann abgespielt wird oder es verändert sich bei dem gefundenen Gegenstand die Farbe. So gehen z.B. in der ersten Bildfrage die Lichter in den Häuser an, wenn das Kind auf das richtige Fenster geklickt hat. Diese kleinen Belohnungen soll dazu dienen, dass das Kind während des Lernprogramms nicht die Lust an der Arbeit verliert.

Nachdem das Kind die Bildfragen der Figur-Grund-Wahrnehmung bearbeitet hat, kommt es nun in die dritte Lektion, in der dem Kind Fragen bezüglich der Wahrnehmungskonstanz gestellt werden.

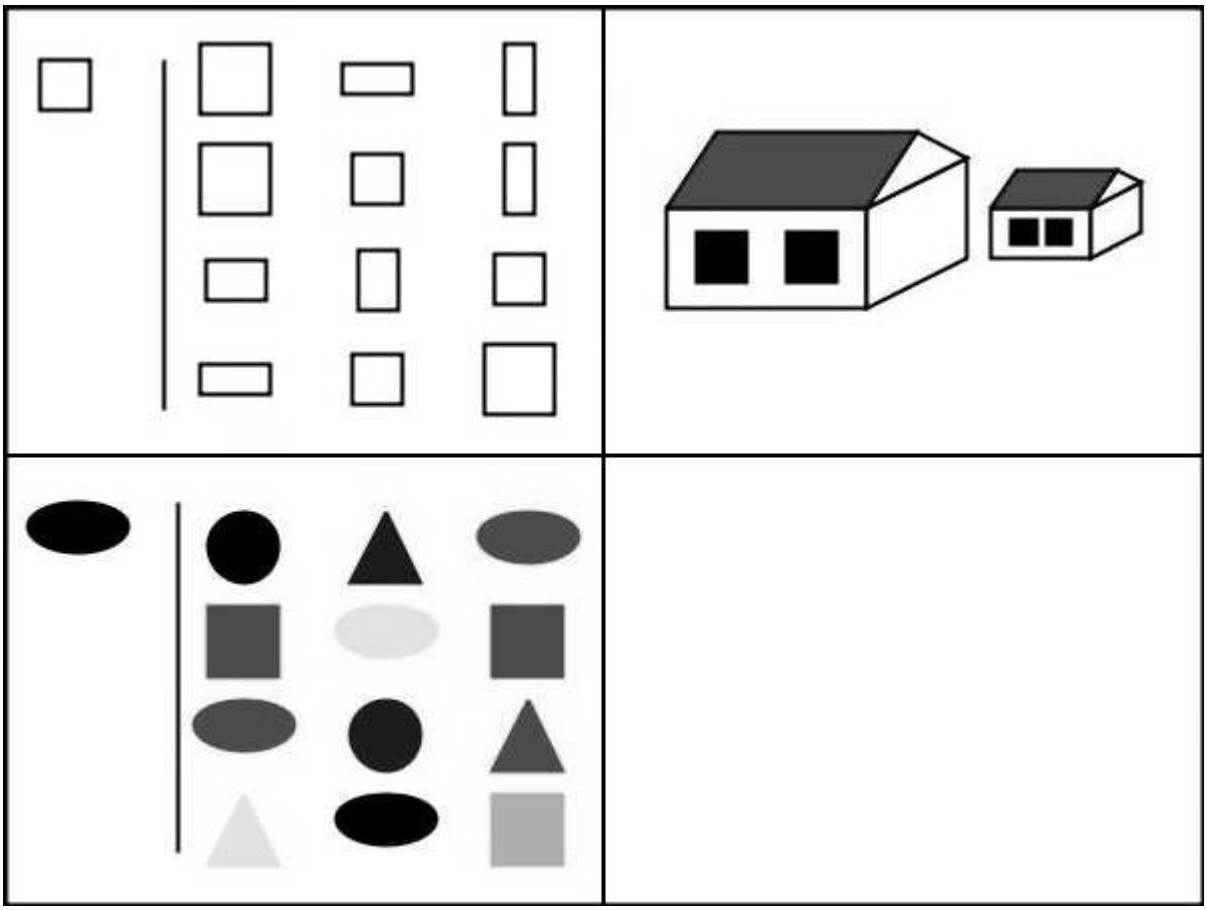

Abbildung 14: Aufgaben der Wahrnehmungskonstanz

Dabei deckt die Lektion die Bereiche Formkonstanz und Größenkonstanz ab. So muss das Kind in der ersten Bildfrage, die Quadrate finden, welche die genau gleiche Form wie das linke Quadrat besitzen. In der letzten Bildfrage hingegen muss es alle Ellipsen finden, wobei erschwerend hinzukommt, dass die Ellipsen unterschiedliche

Farben besitzen. Dazwischen wird dem Kind eine leichte Bildfrage zur Größenkonstanz gestellt, indem es nur auf das Haus klicken muss, welches größer ist.

Nun kommt das Kind in die vierte Lektion mit den Bildfragen der Wahrnehmung der Raumlage. Die einzelnen Bildfragen sind dabei in der Abbildung 15 zu sehen.

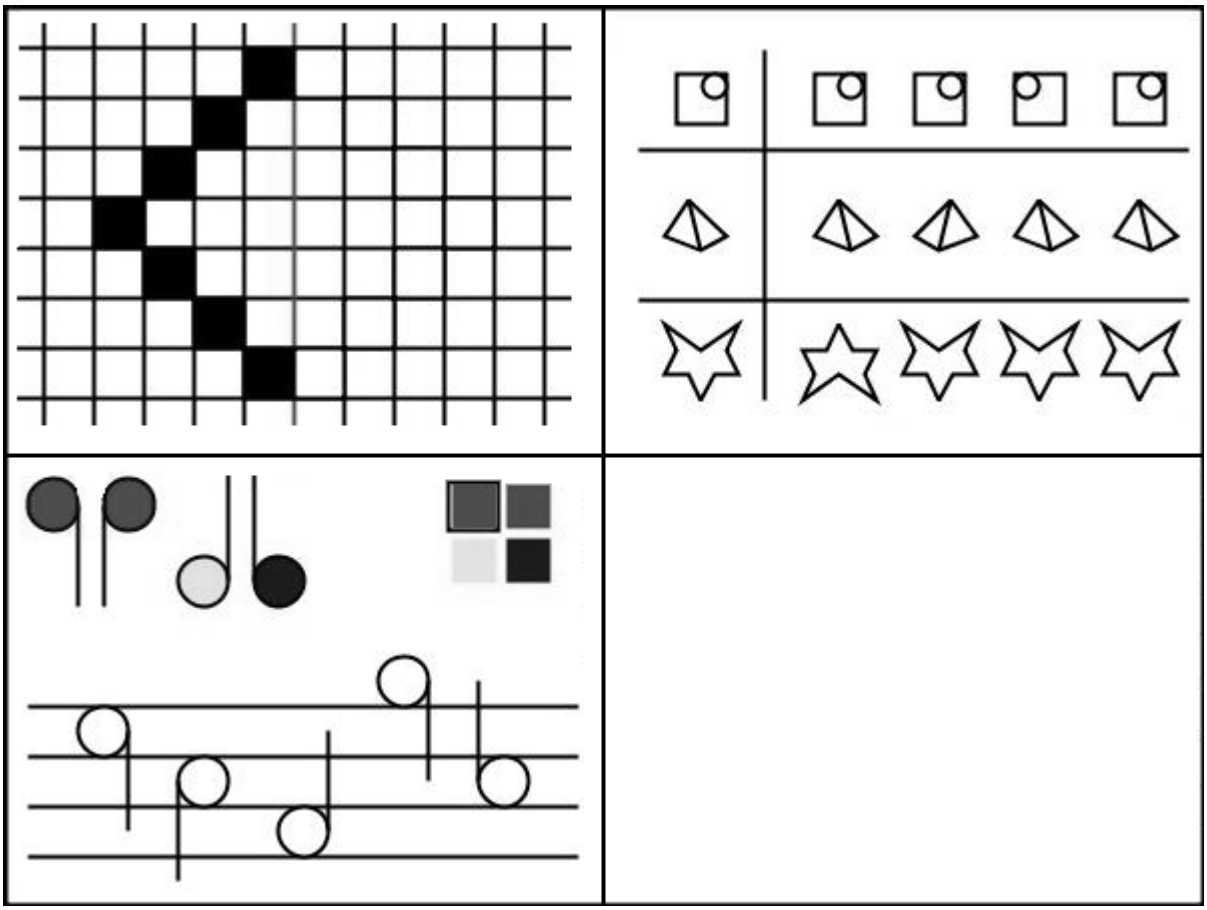

Abbildung 15: Aufgaben der Wahrnehmung der Raumlage

Das Kind muss in all den Bildfragen dieser Lektion erkennen können, wie die einzelnen Grafikelemente räumlich liegen. In der ersten Bildfrage erstellt dabei das Kind das Spiegelbild der schwarz angezeigten Quadrate. Wenn das Kind hierbei einen Fehler macht, so wird ein kleine Sound-Datei abgespielt, die dies dem Kind verdeutlichen soll. Dadurch wird dem Kind geholfen, die richtigen Quadrate auf der rechten Seite anzuklicken. Im Erfolgsfall schließlich klatscht die Bildfrage als Belohnung Beifall. In der anschließenden Aufgabe muss das Kind dann herausfinden, welches der rechts abgebildeten Symbole nicht dem linken Symbol entspricht. Dabei muss es besonders auf die Details achten. So ist der Unterschied

des aus der Reihe tanzenden Symbols der ersten Reihe nur dadurch gegeben, dass der Kreis sich nun links und nicht rechts im Quadrat befindet. Die abschließende Aufgabe schließlich verlangt, dass das Kind die Noten mit der richtigen Farbe ausmalt. Die Schwierigkeit dieser Bildfrage liegt dabei darin, dass das Kind genau auf die Lage der Notenlinie achten muss.

Abschließend werden dann in der letzten Lektion des Lernprogramms die Bildfragen aus dem Bereich der Wahrnehmung räumlicher Beziehungen gestellt.

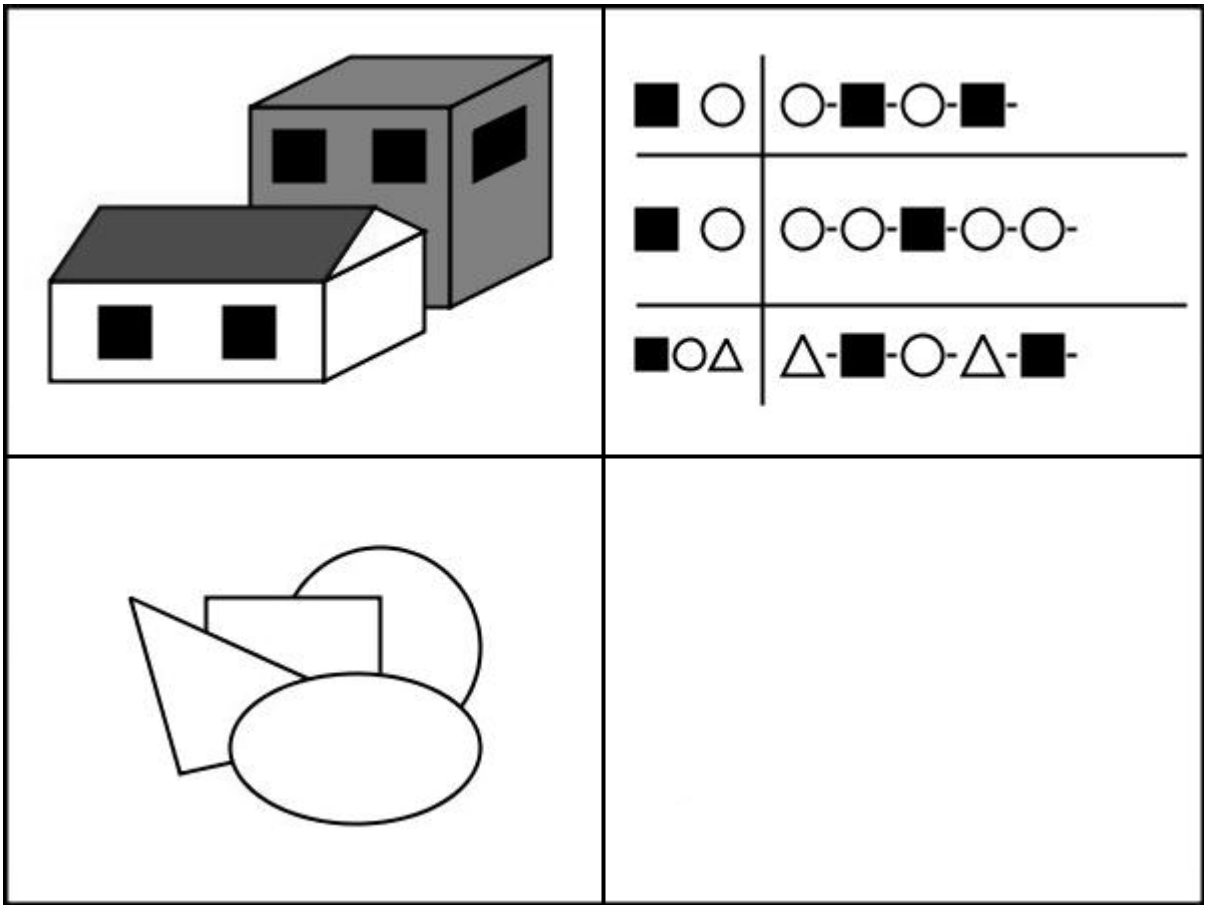

Abbildung 16: Aufgaben zur Wahrnehmung räumlicher Beziehungen

Wie oben schon besprochen geht es in diesem Bereich der visuellen Wahrnehmung darum, dass geprüft wird, ob das Kind die räumliche Beziehungen von Gegenständen untereinander erkennen kann. Die erste und letzte Bildfrage ist deswegen so gestaltet, das das Kind auf die jeweils hinterste Form klicken muss. Dies wäre in erstem Fall das graue Haus und in zweiten Fall der Kreis. Ein bisschen schwieriger hingegen ist die zweite Bildfrage, bei der das Kind zuerst feststellen

muss, welche Beziehung zwischen den einzelnen Elementen der Reihe besteht, bevor es die richtige Form setzt.

Der Kurs wurde auf diese fünfzehn Bildfragen beschränkt, damit das Kind nicht die Lust verliert, wenn das Lernprogramm zu lange dauert. Der Ausbau des Kurses mit mehr Bildern stellt aber kein Problem dar, wenn der Autor dies wünscht. Zusätzlich wurde zu jeder Bildfrage eine kleine Anweisungsdatei aufgenommen, die dem Zweck dienen soll, eine kurze Anweisung zur Bildfrage vorzuspielen, damit das Kind auch auditiv erfährt, was es bei der jeweiligen Bildfrage machen muss.
Für den Lehrer werden dann anschließend nach jeder Bildfrage die durch den Autor eingegebene Rückmeldung über die erbrachte Leistung des Kindes angezeigt. Auf den jeweiligen Feedbackseiten der einzelnen Lektionen erhält der Lehrer schließlich einen kleinen Gesamtüberblick über die Leistungen des Kindes in dem jeweiligen Wahrnehmungsbereich. In den folgenden Kapitel wird nun SVG vorgestellt und anhand ausgewählter Beispiele gezeigt, wie eine Bildfrage zu entwickeln ist und welche Feinheiten dabei zu beachten sind. Abschließend wird dann das Konzept erläutert, wie die Bildfragen in das Autorensystem eingefügt werden, damit sie zur Erstellung eines derartigen Lernprogramms, wie es oben beschrieben wurde, zur Verfügung stehen.

4.2 SVG

SVG steht für Scalable Vector Graphics und hat den Anspruch, das führende Grafikformat im Internet zu werden [Spona]. Der Vorteil gegenüber den bisherigen Grafikformaten wie jpg und gif, liegt darin, dass die einzelnen Grafikelemente nicht Pixel für Pixel abgespeichert werden, sondern dass die einzelnen Vektoren der Grafikelemente abgespeichert werden. So wird bei einer Linie nur der Anfangs- und Endpunkt, sowie einige Attribute bezüglich der Darstellung, abgespeichert. Dadurch verringert sich die benötigte Speicherkapazität für eine Grafikdatei erheblich, was im Internet stets von Vorteil ist. Im Bezug auf die Bildfragen, die für das Lernprogramm zusammengestellt wurden, eignet sich das SVG-Format deswegen so gut, da die einzelnen Grafikelemente der Bildfragen aus Grundformen der Geometrie, wie Kreise, Rechtecke, Ellipsen, etc., bestehen. Für jede dieser Formen stellt SVG einen Befehl zur Verfügung, mit dem man diese Grundform beschreiben kann. Demnach

besteht eine SVG-Datei aus normalen Text, welcher sozusagen das Bild beschreibt. Für die Umwandlung dieses Textes benötigt der Browser - im Fall des Lernprogramms handelt es sich dabei um den Internet Explorer - ein sogenanntes Plug-In, welches für die Anzeige einer SVG-Datei sorgt. Diese Plug-In's sind hierbei im Internet für die unterschiedlichen Browser erhältlich und vermutlich besitzen zukünftige Versionen der Browser ein eigenes Programm zur Anzeige von SVG-Dateien. [Spona]

Der weitere Vorteil von SVG besteht darin, dass die Grafikelemente nicht statisch sind, sondern dass man sie manipulieren kann. Dazu sei kurz erklärt, dass SVG von der Beschreibungssprache XML abstammt. In XML werden dabei die einzelnen Tags als Knoten bezeichnet, die man mit Hilfe von DOM verändern kann. Gleiches gilt auch für die einzelnen SVG-Tags, wobei die einzelnen Tags bei SVG in den meisten Fällen die Befehle für die Grundformen darstellen. Dies wird im in den einzelnen SVG-Datei dadurch ausgenutzt, dass sich das interne Javascript der Datei mit Hilfe von DOM die einzelnen Grafikelemente herholt. Diese Grafikelemente werden dann mit Mouse-Listenern versehen, die registrieren, wenn der Lernende z.B. das entsprechende Grafikelement anklickt oder die Maus über das Grafikelement bewegt. Sollte einer dieser Mouse-Listenern reagieren, wird dann eine Funktion des Javascripts aufgerufen, die für die entsprechende Manipulation des Grafikelements sorgt. Damit ist die Manipulationsmöglichkeit einer SVG-Datei nicht nur auf das Verändern der Farbe einzelner Grafikelemente beschränkt, sondern dem Entwickler sind mit Hilfe des Javascript komplexe Möglichkeiten der Manipulation gegeben, da ihm mit Javascript eine Programmiersprache für seine Wünsche zur Verfügung steht. Dabei ermöglicht erst Javascript das Setzen von Cookies, welche die Antworten des Lernenden bei den Bildfragen darstellen (siehe Kapitel 4.2.1 Aufbau einer Bildfrage).

Neben den bisher genannten Vorteilen von SVG, sind in SVG auch leicht Animationen möglich, d.h. das Bewegen von Grafikelementen, bzw. das Verändern von Farben oder anderen Attributen der Grafikelemente. Hierfür stehen dem Entwickler einer SVG-Datei einfache Animationsbefehle zur Verfügung. Damit eignet sich SVG noch in einem weiteren Punkt besonders gut für das Lernprogramm, da Animationen die Bearbeitung der Bildfragen auflockern und so das Kind mehr Spaß am Lernprogramm hat. Beispielhaft wurde die Möglichkeit der Animation anhand des Titelbilds des Kurses aufgezeigt, welches die Lösung einzelner Bildfragen zeigt. Im

kommenden Kapitel wird nun beschrieben, wie eine einzelne SVG-Datei und damit eine einzelne Bildfrage des Lernprogramms aufgebaut ist.

4.2.1 Aufbau einer Bildfrage

Jede Bildfrage des Lernprogramms besteht dabei mindestens aus zwei Dateien. Dies ist zum einen die SVG-Datei selbst und zum anderen eine ini-Datei, die die wichtigsten Informationen bezüglich der Bildfrage besitzt und die von der Komponente SVGFrage (siehe Kapitel 4.3) ausgelesen wird. Zuerst sei aber der Aufbau der SVG-Datei selbst geklärt.
Eine SVG-Datei besteht dabei wiederum aus zwei Bereichen. Dies sind zum einen die einzelnen Grafikelemente, die für die Anzeige des Bildes nötig sind und zum anderen das eingebettete Javascript, welches für die Manipulationsmöglichkeiten des Bildes sorgt. Beispielhaft für den Aufbau einer Bildfrage sei hierzu die Bildfrage „Spiegelbild“ aus der Lektion 4 betrachtet.

Spiegelbild.svg

Jede SVG-Datei besitzt ein Grundgerüst, welches die Datei als SVG-Datei identifiziert. In der ersten Zeile wird dabei die verwendete XML-Version definiert. Als Nächstes wird mit Hilfe der <DOCTYPE>-Angabe die zu verwendete DTD (DocType Definition) bestimmt. Die DTD legt im allgemeinen fest, welche Tags der Datei zur Verfügung stehen. Die Namensräume, die hierbei für eine SVG-Datei benötigt werden, müssen innerhalb des <DOCTYPE>-Tags angegeben werden. Nach diesem Kopf der SVG-Datei erfolgt dann der eigentliche Code der SVG-Datei, der immer mit dem svg-Tag eingeleitet wird. In diesem Tag wird gleichzeitig neben der Breite und Höhe des anzuzeigenden Bildes die Angabe geschrieben, dass beim Laden des Bildes die Funktion init() des eingebetteten Javascript ausgeführt werden soll.
Bevor jedoch das Javascript beschrieben, welches nur bei den Bildfragen benötigt wird, wird kurz der zweite Teil der SVG-Datei, der sich am Schluss der Datei befindet, vorgezogen. Es handelt sich hierbei um die einzelnen Tags der angezeigten Grafikelemente (siehe Abbildung 15 links oben). So besteht die Bildfrage „Spiegelbild“ aus mehreren Linien und Rechtecken. Für jede einzelne Linie ist dabei ein Befehl angegeben, der diese Linie bezüglich der Position und der Anzeige

beschreibt. Entsprechendes gilt für die Rechtecke. So erzeugt z.B. der folgende Befehl, eine Linie, die an der Position 50/0 beginnt und am Punkt 50/450 endet. Die Liniendicke beträgt dabei 4 und die Linie hat die Farbe schwarz:

<line id="Horlinie1" x1="50" y1="0" x2="50" y2="450" style="stroke-width:4; stroke:black;"/>

Außerdem hat die Linie einen eindeutigen Namen, der sie von den anderen Grafikelementen unterscheidet. Dieser Name wird in dem Attribut id angegeben.
Mit Hilfe dieses Namens kann die Funktion init() des Javascripts die einzelnen Grafikelemente eindeutig identifizieren und die einzelnen Grafikelemente in Klassenfeldern abspeichern. Für das Einlesen der Grafikelemente wurde dabei das DOM-Konzept verwendet. Die Elemente werden außerdem mit verschiedenen MouseListenern verbunden, die eine entsprechende Funktion aufrufen, die für die weitere Manipulation zuständig ist. Im Beispiel wird beim Drücken der linken Maustaste (mousedown) auf ein Grafikelement die Funktion quadratzaehler() aufgerufen. Außer es handelt sich bei dem Grafikelement um die Umrandung, für die die Funktion falschzähler() aufgerufen wird. Neben dem Setzen der MouseListener, ist die Aufgabe der Funktion init() diese, dass sie die im Javascript benötigten Klassenfelder auf Standardwerte setzt und alle Cookies mit dem Wert false initialisiert, was gleichbedeutend ist, dass der Lernende die Bildfrage noch nicht bearbeitet hat. Es liegt dabei im Ermessen des Entwicklers einer Bildfrage welche Cookies er in der Bildfrage setzen möchte. Dabei muss er nur beachten, dass in den Browser nur zwanzig Cookies pro HTML-Seite erlaubt sind. Mit Hilfe der Cookies kann der Entwickler eine detaillierte Möglichkeit anbieten, was der Lernende bei der Bildfrage gelöst hat. Er muss dabei nur dafür sorgen, dass das Cookie auf true gesetzt wird, wenn der Lernende die entsprechende Teilantwort, die das Cookie abdeckt, gegeben hat. Im Beispiel des Spiegelbilds wird das Cookie „zusammengebaut“ dann gesetzt, wenn der Lernende das Spiegelbild komplett gezeichnet hat. Die Cookies „1falsch“, „5falsch“ und „10falsch“ werden hingegen auf „true“ gesetzt, wenn der Lernende einmal, fünfmal bzw. zehnmal auf ein falsches Feld geklickt hat. Das Setzen der Cookies wurde dabei auf die Werte false und true beschränkt, da dies in den meisten Fällen ausreicht. Sollte noch andere Werte benötigt werden, so ist dies durchaus möglich, wobei hierfür die Komponente

SVGFrage mitsamt den dazugehörigen Dateien des Backends geringfügig angepasst werden müssten.

Die übrigen Funktionen des Javascripts werden schließlich dann aufgerufen, wenn der Lernende auf eines der Grafikelemente klickt. So wird in dieser Bildfrage ein kleiner Trick verwendet, indem das Spiegelbild auf der rechten Seite anfänglich mit weißen Quadraten angezeigt wird. Dadurch dass der Hintergrund auch weiß ist, kann der Lernende die richtigen Quadrate nicht erkennen. Sollte er nun auf eines dieser weißen Quadrate klicken, so geschieht folgendes. Da dieses Grafikelement mit einem MouseListener verbunden ist, wird geschaut, welche Funktion nun aufgerufen werden soll. Im Falle des Quadrates handelt es sich um die Funktion quadratzaehler(). Diese Funktion schaut zuerst nach auf welches Grafikelement geklickt wurde. Handelt es sich dabei um ein bisher weißes Quadrat, so verändert die Funktion die Farbe des Quadrates in schwarz und der Lernende sieht nun das Quadrat. Außerdem wird ein Zähler erhöht, der die Anzahl der gefundenen Quadrate speichert. Abschließend wird die Methode pruefung() aufgerufen, die in jeder Bildfrage des Lernprogramms dafür zuständig ist, die Cookies zu setzen. Wenn der Lernende nun nicht ein weißes Quadrat angeklickt hat, so geht die Ereignisverarbeitung an die Funktion falschzaehler(), die ebenfalls einen Zähler verwaltet, der nun aber die Anzahl der Klicks auf ein falsches Quadrat beinhaltet. Danach wird auch hier die Funktion pruefung() aufgerufen.

Die Funktion pruefung() überprüft nun den Wert der einzelnen Zähler und je nachdem welchen Wert sie besitzen, werden die oben genannten Cookies gesetzt. Um keine Namenskonflikte mit anderen Cookies anderer Bildfragen in derselben Lerneinheit zu bekommen, werden den Namen der gesetzten Cookies der Name der Bildfrage vorangestellt. Der Name der Billdatei ist dabei in dem id-Attribut des svg-Tags gespeichert.

Die letzte Funktion des Javascripts ist schließlich die Funktion play() mit deren Hilfe eine Sounddatei abgespielt werden kann, die zuvor über die xsl-Datei der SVG-Frage (siehe Kapitel 3.2.1 „XSL-Dateien“) in die HTML-Seite eingebettet wurde. Dabei wurde die Funktion so programmiert, dass die Sounddatei auch nur dann abgespielt wird, wenn sie auch wirklich in der HTML-Seite vorhanden ist, da ansonsten eine Fehlermeldung während des Programmablaufs angezeigt werden würde. Damit ist die Beschreibung der Datei spiegelbild.svg zu Ende und es dürfte klar sein, wie eine Bildfrage aufgebaut ist und wie der Lernende durch die Manipulation der Bilddatei die

Cookies und damit die Antworten setzt, die von der Komponente SVGFrage weiter verarbeitet werden (siehe Kapitel 4.3). Die übrigen Bildfragen halten sich dabei an denselben Aufbau wie die hier vorgestellte Datei spiegelbild.svg. Lediglich in der Manipulation der einzelnen Grafikelemente unterscheiden sich die Dateien, da in den verschiedenen Bildfragen die einzelnen Grafikelemente auf unterschiedliche Weise verändert werden. Hierfür sei auf den Quellcode verwiesen, um nicht den Umfang des Dokuments zu sprengen.

Spiegelbild.ini

Die zweite Datei einer Bildfrage ist weitaus weniger komplex als die eigentliche SVG-Datei. Wie oben schon kurz angedeutet, sind in dieser Datei die wichtigsten Informationen bezüglich der Bildfrage gespeichert. Dies sind u.a. die Cookies, die von der Bildfrage gesetzt werden können. Dabei handelt es in dem Beispiel um die Cookies mit den Namen „zusammengebaut“, „1falsch“, „5falsch“ und „10falsch“. Des weiteren wird eine Beschreibung der Cookies gespeichert, die dem Autor angeben soll, wann welches Cookie gesetzt wird. Außerdem stehen die Angaben zur Größe der Bildfrage in der Datei und optional die Namen der Sounddateien der Bildfrage, sollte diese welche verwenden. All diese Informationen werden von der Komponente SVGFrage, die im nächsten Abschnitt beschrieben wird, eingelesen, damit der Autor des Lernprogramms die Möglichkeit hat, die einzelnen Bildfragen in sein Lernprogramm einzubauen.

4.3 Komponente SVGFrage

Mit Hilfe von SVG ist es also möglich eine einzelne Bildfrage zu erstellen, deren Antworten in Form von Cookies gesetzt werden. Was nun noch fehlt ist eine Komponente des Autorensystems, die dafür sorgt, das diese Bildfrage in ein Lernprogramm eingebaut werden kann. Hierzu wurde die Komponente SVGFrage entwickelt, die wiederum zeigt, dass ein offenes Autorensystems den großen Vorteil bietet, dass der Entwickler einer Komponente, diese nach seinen eigenen Wünschen gestalten und anpassen kann, wie er sie benötigt. Die Komponente SVGFrage ermöglicht es dem Autor dabei erst, dass er für die gegebenen Antworten des

Lernenden eine Rückmeldung an diesen geben kann bzw. für die Antworten Punkte zu vergeben.

Der Aufbau ähnelt dabei dem Aufbau der Bewertungskomponenten, welche im Kapitel 3.1.4 beschrieben wurde. Dies dürfte auch nicht verwundern, da die Komponente SVGFrage wie alle anderen Komponenten auch die gemeinsame Schnittstelle für Komponenten, die Klasse Komponente, erweitert. So haben viele der Methoden der Klasse SVGFrage dieselbe Aufgaben, wie die gleichnamigen Methoden der Bewertungskomponenten.

Aus diesem Grund wird hier auf die Beschreibung der Methode neu, edit, fertigstellen und synchronisieren verzichtet (siehe Kapitel 3.1.4). Des weiteren haben die Methoden laden, speichern, importieren und exportieren die Aufgabe die Daten der SVG-Frage, die von der Klasse Lerneinheit geliefert werden in die entsprechenden Klassenfelder zu schreiben, bzw. die Daten als Knoten an die Klasse Lerneinheit zu liefern. Die beiden letzteren Methoden sorgen wiederum dafür, dass die Komponente an einem anderen Speicherort gesichert bzw. von dort hergeholt werden kann. Zu beachten ist bei diesen ganzen Methoden, dass die Komponente SVGFrage eine Verweiskomponente ist, d.h. es müssen bei den ganzen Lade- und Speicheroperationen die externen Dateien mit beachtet werden, die die Komponente benötigt. Bei der SVG-Frage sind dies die svg-Datei, die ini-Datei und die optional verwendeten Sounddateien. Als weiteres kommt noch eine jpg-Datei hinzu, die zur Anzeige der Kurzinformationen der Komponente in der Lerneinheit-Übersicht dient. Ein weiterer Punkt, der beim Laden bzw. Speichern beachtet werden muss, ist dieser, dass nicht alle Informationen über die Methode laden geholt werden bzw. über die Methode speichern geliefert werden. Die restlichen Informationen, dabei handelt es um die Attribute zur Größe des Bildes bzw. um den eindeutigen Namen der SVG-Frage, werden über die Methoden setAttributeListe von der Klasse Lerneinheit aus gesetzt bzw. mit der Methode getAttributeListe von dieser geholt.

Wie es bei den Komponenten allgemein üblich ist, werden die eingegebenen Daten des Autors vor dem Abspeichern aber zuerst geprüft. Diese Aufgabe wird wiederum von der Methode datenVerarbeitung übernommen, da diese von der Klasse Editor aus aufgerufen wird. Die Methode datenVerarbeitung hat dabei ein anderes Aussehen als die gleichnamige Methode der Bewertungskomponenten, da in der Komponente SVGFrage andere Klassenfelder geprüft werden müssen. Die aufwendigste Prüfung nimmt dabei vor allem die Prüfung der

Auswertungsvorschriften der einzelnen Auswertungsbereiche ein. So ist es bei den Fragekomponenten möglich, dass der Autor eine beliebige Anzahl von Auswertungsblöcken eingeben kann. Für jeden Auswertungsblock kann der Autor eine individuelle Anzahl von Punkten und ein individuelle Rückmeldung angeben. Außerdem muss der Autor für jeden Auswertungsblock eine Auswertungsvorschrift eingeben. Eine Auswertungsvorschrift ist dabei eine logische Kombination der Antworten der Frage. Im Falle der Komponente SVGFrage sind die Cookies, die die Bildfrage setzen kann, die Antworten, die der Autor mit den Operatoren NICHT, UND und ODER verknüpfen kann. Zur Eingabe einer einzelnen Auswertungsvorschrift siehe weiter unten. Der Sinn einer Auswertungsvorschrift ist dieser, dass der Autor eines Lernprogramms die Leistungen des Lernenden beliebig untergliedern kann, je nachdem welche Antworten der Lernende gegeben hat. Hierin spielt auch das Konzept der nachfragenden Lerneinheit mit herein, welche zumeist dann aufgerufen wird, wenn der Lernende eine teilweise richtige Antwort gegeben hat (siehe Diplomarbeit Nr. 2076 [Conradt]). Diese Auswertungsvorschrift wird nun in der Methode datenVerarbeitung daraufhin überprüft, dass getestet wird, ob die eingegebene Kombination immer wahr wird. In diesem Falle wird eine Fehlermeldung ausgegeben und der Autor muss diese Auswertungsvorschrift ändern. Für diese Prüfung werden die Methoden vergleichsauswertung und logikparser benötigt, bei denen es sich um leicht abgewandelten Code der gleichnamigen Methoden der Klasse SVGFrageAuswertung des Backends (siehe Kapitel 3.2.3.3) handelt. Dies wird dadurch verständlich, dass bei beiden Klassen den Methoden eine Eingabe des Lernenden zur Analyse gegeben wird. Im Frontend handelt es sich hierbei um eine von der Komponente konstruierte Eingabe, während es sich im Backend um die durch den Lernenden gesetzten Cookies handelt.

Bevor jedoch die Methode datenVerarbeitung die eingegebenen Daten des Autors prüfen kann, muss dieser sie zuvor erst eingegeben haben. Hierfür stehen dem Autor drei Eingabefenster zur Verfügung, die dem Autor der Reihenfolge nach angezeigt werden. Die drei Eingabepanels wurden dabei wiederum von den Methoden erzeugeAuswertungsBlock(), erzeugeEingabePanel() und erzeugePanelListe() erzeugt. Die Unterschiede zu den Bewertungskomponenten liegen darin, dass sich das Aussehen der Eingabefenster verändert hat, da nun andere Daten vom Autor erwartet werden und dass die Methode erzeugePanelListe() nun drei Eingabefenster für den Editor erzeugt, im Gegensatz zu dem einen Eingabefenster der

Bewertungskomponenten. In Abbildung 17 ist beispielhaft das erste Eingabefenster der Komponente SVGFrage angezeigt.

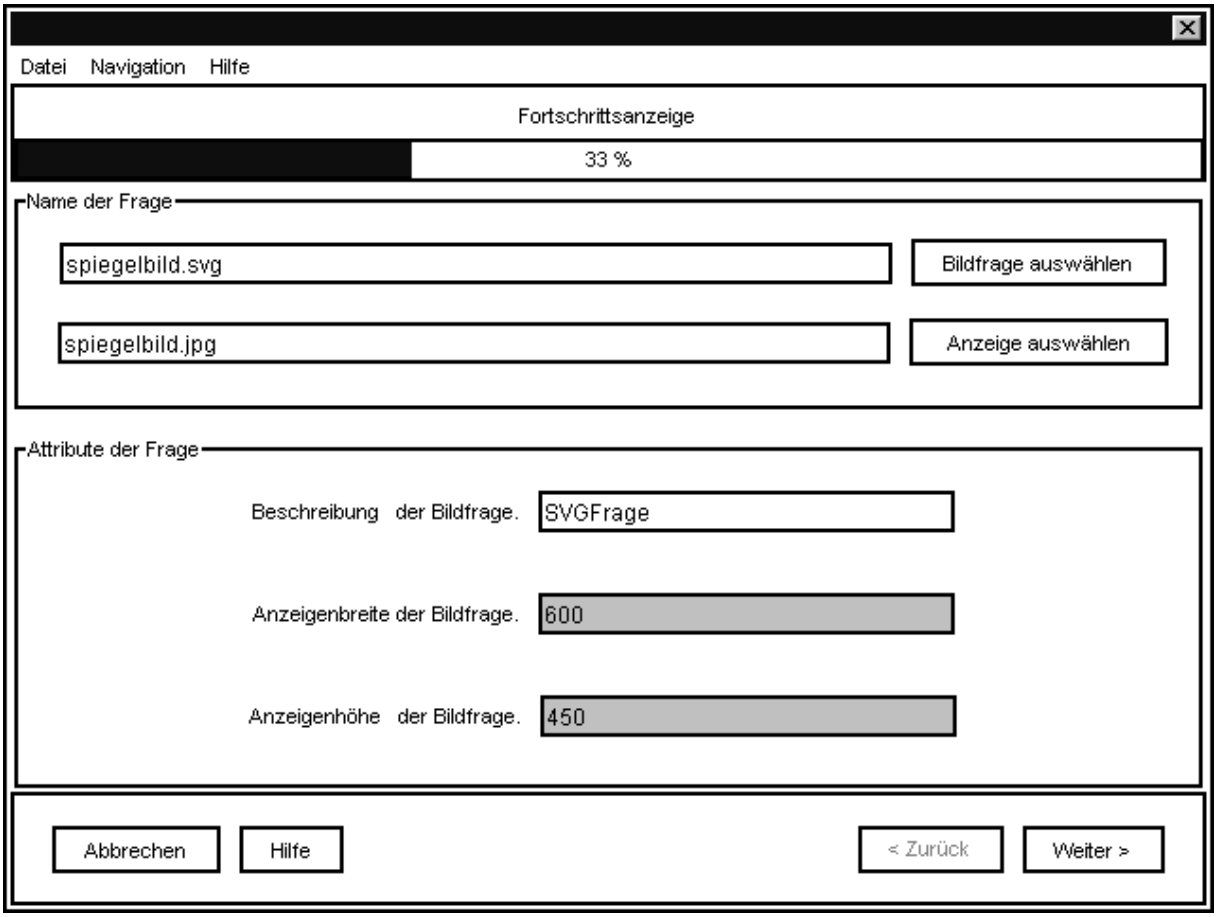

Abbildung 17: Erstes Eingabefenster der Komponente SVGFrage

Das Programm erwartet hier vom Autor, das er über die Schaltfläche „Bildfrage auswählen“ eine SVG-Datei einlädt, die einer Bildfrage entspricht. D.h. es muss neben der eigentlichen SVG-Datei auch die dazugehörige ini-Datei existieren. Bevor jedoch der Name der Bildfrage im nebenstehenden Anzeigefenster erscheint, werden verschiedene Prüfungen durchgeführt. So wird zuerst überprüft, ob die ini-Datei mit den wichtigen Informationen existiert. Wenn nicht handelt es sich um keine Bildfrage. Des weiteren wird überprüft, ob die externen Dateien der neu geladenen Bildfrage andere externe Dateien in derselben Lerneinheiten überschreiben würden, was nicht erlaubt ist. Die übrigen Prüfungen sind dann anschließend von der Art, dass geschaut wird, ob in der ini-Datei falsche Werte angegeben sind. All diese Überprüfung sind die Aufgabe der inneren Klassen DateiAuswahlAction, die aufgerufen wird, wenn der Autor die Schaltfläche „Bildfrage auswählen“ gedrückt hat. Nachdem anschließend die Werte der ini-Datei gespeichert sind, werden diese auf

den verschiedenen Eingabefenster angezeigt. Hierfür werden die Methoden seite0PanelBildquelleDynamisch(), seite0PanelBreiteHoeheDynamisch() und seite1PanelBeschreibungDynamisch() aufgerufen, die den entsprechenden Teil der Grafik aktualisieren.

Bei den Werten der ini-Datei handelt sich dabei um die Angaben zur Größe der Bildfrage und um eine Beschreibung, wann die Cookies gesetzt werden. Diese Beschreibung wird dem Autor auf dem zweiten Eingabefenster angezeigt, welches bei anderen Fragearten (siehe Diplomarbeit Nr. 2076 [Conradt]) das Eingabefenster für die Antworten ist, welche der Autor erwartet. Im Gegensatz zu den anderen Fragearten sind die Antworten bei der SVG-Frage jedoch schon durch den Entwickler der Bildfrage vorgegeben. Damit die Komponente diese Antworten bzw. Cookies kennt, werden diese ebenfalls aus der ini-Datei ausgelesen und stehen dem Autor beim Setzen der Auswertungsvorschrift zur Verfügung. Die letzte Information schließlich, die aus der ini-Datei geholt wird, ist die Angabe der von der Bildfrage verwendeten Sounddateien.

Nachdem der Autor die Bildfrage geladen hat, muss er auf dem ersten Eingabefenster noch die Beschreibung zu dieser Bildfrage eingeben, welche standardmäßig den Wert „SVGFrage“ hat. Diese Beschreibung wird u.a. bei den Feedbackseiten des Backends benutzt. Die letzte Eingabe, die schließlich vom Autor erwartet wird, ist die Angabe eines Bildes, welches in der Komponentenübersicht einer Lerneinheit angezeigt werden soll. Dabei überprüft die zuständige innere Klasse AnzeigeDateiauswahlAction wiederum darauf ab, ob eine bereits vorhandene externe Datei einer anderen Komponente überschreiben würde.

Anschließend kommt der Autor durch das Drücken der „Weiter“-Schaltfläche auf das bereits beschriebene zweite Eingabefenster. Nachdem der Autor nun die Informationen zu den Cookies gelesen hat, kommt er auf das letzte Eingabefenster, dessen Layout und Funktionsweise sich an die Fragekomponenten der Diplomarbeit Nr. 2076 [Conradt] anlehnt.

Auf diesem letzten Eingabefenster stehen dem Autor eine Vielzahl von Eingabemöglichkeiten zur Verfügung. Zum einen kann der Autor die Punkte und die Rückmeldung für jeden Auswertungsbereiches angeben. Der Autor kann dabei über die Schaltfläche „weitere Auswertung“ und „markierte Auswertung(en) löschen“ die Anzahl der benötigten Auswertungsbereiche bestimmen. Je nachdem, welche der beiden Schaltflächen der Autor gedrückt hat, werden dabei die inneren Klassen

WeitereAuswertungAction bzw. EntferneAuswertungAction aufgerufen, die für die korrekte Verwaltung der Klassenfelder zuständig sind und anschließend die Grafik aktualisieren.

Wann der Lernende in einen dieser Auswertungsbereiche kommt, hängt dabei nur von der Auswertungsvorschrift ab. Zur Eingabe dieser Auswertungsvorschrift muss der Autor auf die Schaltfläche „bearbeiten“, welche sich links oben im jeweiligen Auswertungsblock befindet, klicken. Es öffnet sich nun ein weiteres Eingabefenster, welches in der Diplomarbeit Nr. 2076 [Conradt] entwickelt und für die SVG-Frage angepasst wurde.

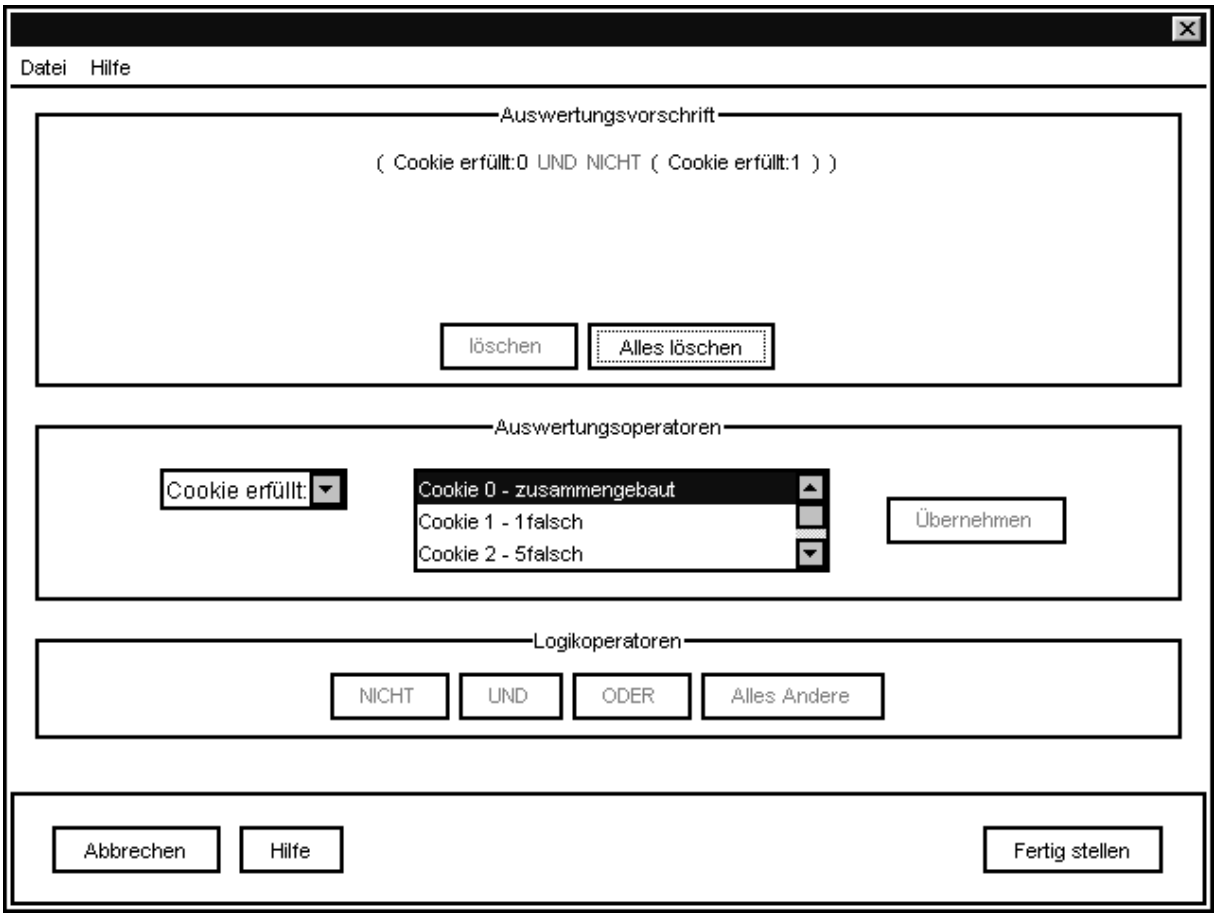

Abbildung 18: Eingabefenster der Auswertungsvorschrift

Das Fenster, welches von den Methoden erzeugeLogikPanel() und erzeugeLogikPanelListe() erzeugt wird und in der Abbildung 18 angezeigt ist, ist dabei in drei Bereiche unterteilt. In der Mitte sind die Cookies angezeigt, die aus der ini-Datei eingelesen wurden. Im unteren Bereich sind die Logikoperatoren NICHT, UND und ODER angegeben, mit Hilfe derer der Autor die gewünschte Auswertungsvorschrift zusammenstellen kann, die dann im oberen Bereich angezeigt

wird. Für das Zusammenstellen einer Auswertungsvorschrift markiert der Autor im oberen Bereich des Eingabefensters den gewünschten Teil der Auswertungsvorschrift. Danach besitzt er die Möglichkeit mit Hilfe der Logikoperatoren die entsprechende Logik in die Auswertungsvorschrift einzubauen oder über die „Übernehmen“-Schaltfläche ein Cookie einzubauen. Dabei bedeutet das Setzen eines Cookies, das das Cookie erfüllt sein muss, d.h. das der Wert des Cookies auf true stehen muss.

Beispielhaft würde das im Falle der Abbildung 18 bedeuten, dass diese Auswertungsvorschrift dann wahr wird, wenn der Lernende durch seine Manipulation an der angezeigten Bildfrage das Cookie 1 („zusammengebaut“) auf true und das Cookie 2 („1falsch“) nicht auf true gesetzt hat, was genau die Auswertungsvorschrift *(Cookie erfüllt: 0 UND NICHT (Cookie erfüllt: 1))* aussagt. Für die Funktionsweise dieses Eingabefensters werden bei den verschiedenen Schaltflächen die inneren Klassen NOTAction, AndOrAction, AllaAction, ComboBoxAction, UebernehmenAction, LogikLoeschenAction und AllesLoeschenAction benötigt, die für das korrekte Setzen bzw. Löschen der Auswertungsvorschriften zuständig sind. Hierfür benutzen diese Klasse die Hilfsmethode logikGrenzen(), die für die Berechnung der Grenzen beim Markieren eines Teils der Auswertungsvorschrift zuständig ist. Des weiteren werden die Methoden auswertungsvorschriftCodieren() und auswertungsvorschriftDecodieren() benötigt, die die Auswertungsvorschrift in eine Form bringen, damit sie besser für den Autor zu lesen ist, während der Computer für das Backend eine andere kompaktere Form benötigt. All diese Methoden und inneren Klassen wurden in der Diplomarbeit Nr. 2076 [Conradt] bei der Entwicklung der Fragekomponenten mit entwickelt und sind dort näher beschrieben. Für die SVG-Frage wurden nur kleine Anpassungen durchgeführt, die sich vor allem darauf beschränken, das die SVG-Frage andere Auswertungsoperatoren benutzt, wie die übrigen Fragearten.

Nachdem der Autor nun die Auswertungsvorschrift eingegeben hat, hat er im Prinzip alle Informationen für eine SVG-Frage eingegeben, die für den korrekten Ablauf im Backend benötigt werden. Dabei muss er bei den Auswertungsvorschriften beachten, dass er bei einem der Auswertungsbereiche den Fall „Alles andere“ eingegeben hat, welcher dann eintritt, wenn keine der anderen Auswertungsvorschriften zutrifft. Dies ist notwendig, damit der Lernende nicht die Möglichkeit hat, im Backend eine Eingabe zu machen, die in keinen Auswertungsbereich fällt.

Da es sich bei der Komponente SVGFrage um eine Frageart handelt, hat der Autor schließlich noch die Möglichkeit für jeden der Auswertungsbereiche eine nachfragende Lerneinheit vorzusehen, die dann ausgelöst wird, wenn der Lernende in diesen Auswertungsbereich kommt und die Bewertungskomponente der Ebene der Lerneinheit in einem Bereich ist, der vorsieht, dass die nachfragenden Lerneinheiten der einzelnen Fragen angezeigt werden sollen. Zur genauen Funktionsweise siehe Diplomarbeit Nr. 2076 [Conradt]. Hier sei nur kurz gesagt, dass der Autor auf dem letzten Eingabefenster die Möglichkeit hat, eine nachfragende Lerneinheit genauso wie eine „normale“ Lerneinheit zusammenzubauen. Er muss dabei nur die Schaltfläche „bearbeiten“, welche sich im jeweiligen Auswertungsblock rechts oben befindet, drücken und es öffnet sich eine weitere Instanz der Klasse Lerneinheit, auf der der Autor nun die einzelnen Komponenten der nachfragenden Lerneinheit zusammenstellen kann.
Schließlich muss der Autor nur noch auf die Schaltfläche „Fertig stellen“ des Editors drücken und die SVG-Frage ist als Komponente in der Lerneinheit eingebaut.
In dieser Komponentenübersicht wird für die SVG-Frage eine Kurzinformation angezeigt, die dem Autor die wichtigsten Daten zu dieser Komponente präsentiert. Für die Zusammenstellung dieser Kurzinformation ist hierbei die innere Klasse SVGFrageDarstellung zuständig, die ein Panel mit Angaben der Beschreibung, der Anzahl der Cookies und der Anzahl der Auswertungsbereiche erzeugt. Zusätzlich wird in das Panel das Anzeigebild eingefügt, welches der Autor auf dem ersten Eingabefenster eingegeben hat. Der Zugriff auf das von der inneren Klasse SVGFrageDarstellung erzeugte Panel erfolgt hierbei über die Methode getPanel(). In diesen Zusammenhang fallen auch die Methode visuellerUpdate(), die immer dann aufgerufen, wenn die Klasse Lerneinheit die Komponentenübersicht erneuern muss, und die Methode ressourceFreigeben(), die dafür sorgt dass der Bildspeicher für das Anzeigebild freigegeben wird, wenn dieser nicht mehr nötig ist.

Bei den übrigen Methoden der Klasse SVGFrage handelt es sich schließlich alles um Hilfsmethoden, die an diversen Stelle der Klasse Lerneinheit aufgerufen werden. Es handelt sich dabei in den meisten Fällen um Übergaben kleiner Informationen der Komponente an die Klasse Lerneinheit. Dabei sind die zwei wichtigsten Methoden, die Methoden getMaxPunkte() und die Methode getExterneDateien().

Die erste Methode gibt hierbei die maximale Punktzahl zurück, die der Lernende für diese SVG-Frage bekommen kann. Während die zweite Methode die Namen aller externen Dateien zurückgibt, die von der SVG-Frage benutzt werden. Dabei sind auch die externe Dateien mit angegeben, dies sich in den einzelnen nachfragenden Lerneinheiten befinden. Die letzte Methode wird hierbei von der Klasse Lerneinheit dann aufgerufen, wenn der Autor die Ebene der Lerneinheit verlässt und die Klasse Lerneinheit dafür sorgen muss, dass überflüssige externe Dateien aus dem zur Lerneinheit zugehörigen Verzeichnis gelöscht werden müssen.

Zusammenfassend sei hier noch einmal ein kurzer Überblick gegeben, wie die einzelnen Teile einer Bildfrage im Lernprogramm zusammenspielen. Zuerst muss ein Entwickler eine Bildfrage mit Hilfe von SVG erstellen. Diese Bildfrage ist für das spätere Setzen der Cookies im Backend zuständig.
Für das Frontend wird zusätzlich eine ini-Datei benötigt, in der die wichtigsten Informationen zur Bildfrage stehen. Als nächstes wird die Komponente SVGFrage benötigt mit Hilfe derer der Autor die Bildfrage als Komponente in eine Lerneinheit einladen kann. Dabei besitzt der Autor die Möglichkeit die Antworten bzw. in diesem Fall die Cookies in einer von ihm gewünschten Auswertungsvorschrift zusammenzustellen und den einzelnen Auswertungsbereichen zuzuordnen. Zusätzlich orientiert sich die SVG-Frage an andere Fragearten (siehe Diplomarbeit Nr. 2076 [Conradt]) damit es möglich ist auch nachfragende Lerneinheit einzugeben.
Im Backend schließlich sorgen die einzelnen xsl-Dateien (siehe Kapitel 3.2.1) für die korrekte Anzeige der SVG-Frage. Die einzelnen Grafikelemente der Bildfrage kann der Lernende nun mit Hilfe der Maus manipulieren, was vom in die SVG-Datei eingebetteten Javascript registriert und bearbeitet wird. Hat der Lernende dabei was richtig gemacht, wird vom selben Javascript ein entsprechendes Cookie gesetzt. Alle anderen Cookies, die nicht vom Javascript gesetzt werden, werden in der Auswertungsklasse auf false gesetzt. Wenn der Lernende mit allen Fragen auf der Lerneinheit fertig ist, drückt er die Schaltfläche „Weiter“, um zum Servlet zu kommen. Während des Drückens der Schaltfläche wird dabei ein weiteres Javascript (kurse.js) ausgeführt, welches dafür sorgt dass alle Cookies der HTML-Seite als HIDDEN-Informationen in die Seite mit einfügt werden. Des weiteren werden alle Cookies, die nicht dem Schlüssel für das Sitzungstracking entsprechen, gelöscht, damit es zu keinen unvorhergesehenen Konflikten kommt, wenn der Lernende auf dieselbe

Lerneinheit zurückkommen sollte und die Cookies noch gesetzt sind. Das Servlet verarbeitet anschließend alle ankommenden Informationen und lässt die entsprechenden Informationen den zuständigen Auswertungsklassen zu kommen. Im Falle der SVG-Frage ist dies die Klasse SVGFrageAuswertung (siehe Kapitel 3.2.3.3), die nun die Aufgabe hat, die gesetzten Cookies zu analysieren und die Punkte des Bereichs zurückzugeben, dessen Auswertungsvorschrift wahr geworden ist. Für den restlichen Ablauf des Backends sei dann auf das Kapitel 3.2 verwiesen.

4.4 Präsentationskomponenten

Neben der Komponente SVG-Frage werden dabei für das Lernprogramm noch weitere Komponenten benötigt, die in diesem abschließenden Kapitel beschrieben werden. Hierbei handelt es um die Präsentationskomponenten JPG-Bild-Komponente, GIF-Bild-Komponente, SVG-Bild-Komponente und Sound-Komponente. All diese Komponenten dienen dazu den Lernstoff dem Lernenden darzubieten. Dabei sorgen die ersten drei Komponenten dafür, dass der Autor ein Bild mit dem entsprechenden Grafikformat in sein Lernprogramm einbauen kann. Bei der letzten Komponente hingegen besitzt der Autor die Möglichkeit eine Sound-Datei des wav-Formats einzubauen.

Klasse JPGBildKomponente

Der Aufbau dieser Komponente unterscheidet sich dabei nur geringfügig vom Aufbau der anderen bereits beschriebenen Komponenten. So übernehmen die einzelnen Methoden der Klasse wiederum dieselben Aufgaben ein, wie die gleichnamigen Methoden in den anderen Komponenten. An dieser Stelle werden deshalb nur die Besonderheiten der Klasse JPGBildKomponente erläutert.
Der erste Unterschied liegt dabei wiederum im Aussehen des Eingabefenster, welches in der Abbildung 19 zu sehen ist. Ähnlich wie bei der SVG-Frage muss der Autor über die Schaltfläche „Datei auswählen“ ein Bild des Formats jpg laden. Dabei wird in der inneren Klasse DateiAuswahlAction diesmal nicht sofort überprüft, ob die externe Bilddatei eine andere externe Datei der Lerneinheit, die im gleichnamigen Unterverzeichnis gespeichert sind, überschreiben würde. Dies ist die Aufgabe der Methode datenVerarbeitung(), welche weiter unten beschrieben wird. Das

eingeladene Bild wird außerdem durch die Methode seite0PanelBildanzeigeDynamisch() angezeigt, nachdem es eingeladen wurde. Als weiteres besitzt der Autor schließlich die Möglichkeit, die Breite und Höhe des Bildes zu verändern. Wenn der Autor anschließend die Schaltfläche „Aktualisieren“ drückt, wird das eingeladene Bild mit den neuen Größenangaben angezeigt, was die Aufgabe der inneren Klasse AktualisierenAction ist, die beim Drücken der Schaltfläche „Aktualisieren“ aufgerufen wird. Die Eingabe der Größenangaben wurde dabei so abgesichert, dass der Autor nur Ziffern eingeben kann. Wenn der Autor einen Buchstaben eingibt, so wird von der inneren Klasse EingabeListener eine Fehlermeldung angezeigt und der vorherige Zustand wird wieder hergestellt.

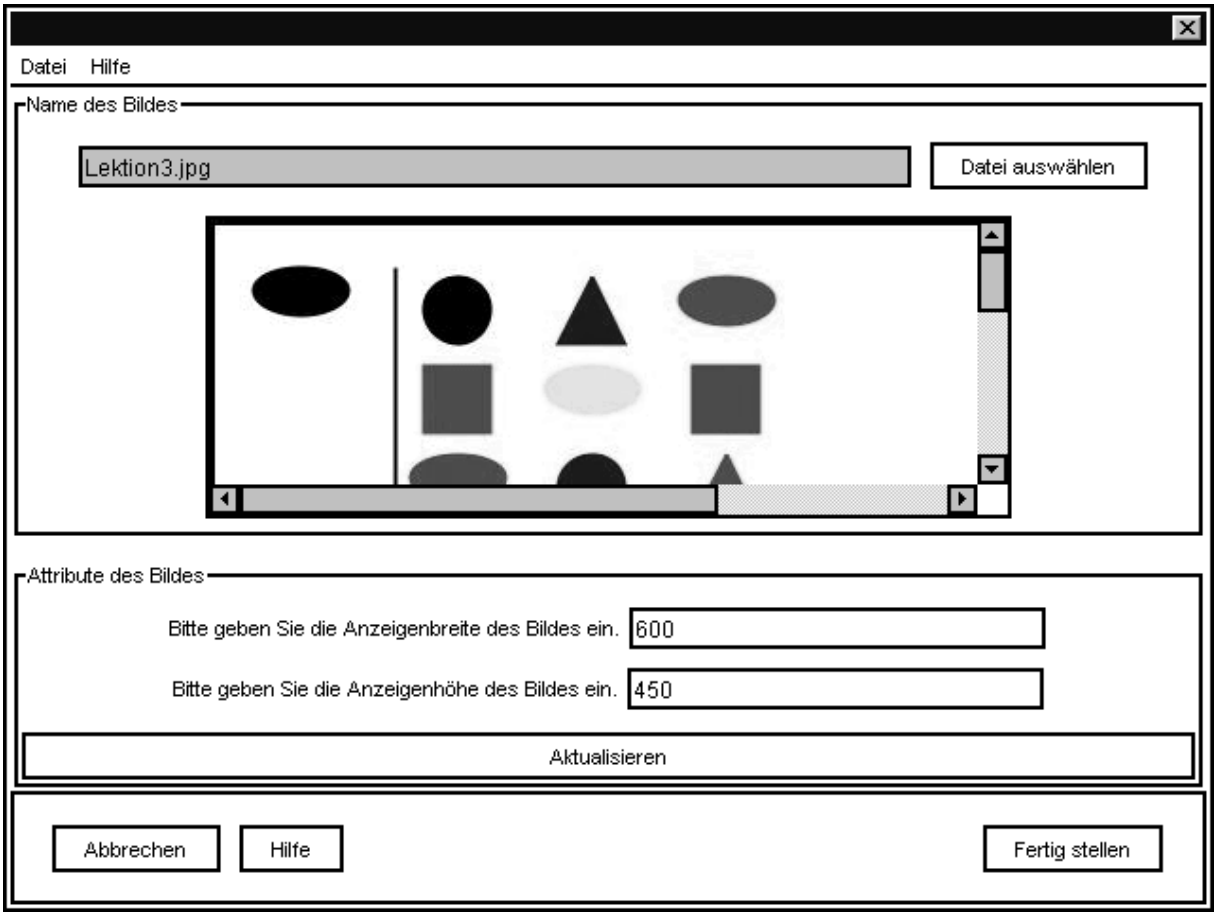

Abbildung 19: Eingabefenster der JPG-Bild-Komponente

Für die Anzeige des Eingabefenster sind dabei, wie in allen anderen Komponenten auch, die Methoden erzeugeEingabePanel() und erzeugeEingabePanelListe() zuständig. Die Methoden seite0PanelFehlerDynamisch(), seite0PanelDateinameDynamisch(), seite0PanelBreiteHoeheDynamisch() und seite0PanelBildanzeigeDynamisch() dienen dementsprechend für die Aktualisierung

von Teilen des Eingabefensters, wenn sich die entsprechenden Teile durch die Eingaben des Autors ändern.

Der zweite Unterschied in der Klasse JPGBildKomponente liegt in der Methode datenVerarbeitung(), die wie immer für die Überprüfung der eingegebenen Daten sorgt. Dabei hat diese Methode bei dieser Komponente nur zu überprüfen, ob der Autor eine Bilddatei geladen hat, die eine andere Bilddatei überschreiben würde. Sollte dies der Fall sein, geht ein Dialogfenster auf, der den Autor fragt, ob er die bereits vorhandene externe Datei überschreiben möchte, oder für die eingeladene Datei einen neuen Namen eingeben möchte. Wenn der Autor sich für die zweite Wahl entscheidet öffnet sich ein weiteres Dialogfenster, welches als Eingabe diesen neuen Namen erwartet. Da der Autor nun wiederum einen Namen eingeben kann, der dem Namen einer bereits vorhandenen externen Datei entspricht, wird dies wiederum geprüft. Die Prüfung wird solange fortgesetzt, bis der Autor einen Namen eingegeben hat, der keine externe Datei überschreibt oder bis er sich einverstanden erklärt, dass die bisherige externe Datei überschrieben werden darf. Daneben besitzt der Autor immer noch die Möglichkeit den Dialog abzubrechen, um sich in Ruhe einen Namen auszudenken oder den Editor über die Schaltfläche „Abbrechen“ zu verlassen.

Die angezeigten Dialoge werden dabei durch die innere Klassen umbenennenFrage und neuerNameEingabe erzeugt.

Nachdem der Autor dann das Eingabefenster über die Schaltfläche „Fertig stellen“ verlassen hat, sorgt die innere Klasse JPGBildKomponenteDarstellung für die Anzeige der Kurzinformationen, die im wesentlichen aus der Anzeige des eingeladenen Bildes besteht. Da hier wiederum für das angezeigte Bild Speicher benötigt wird, besitzt auch die Klasse JPGBildKomponente die Methode ressourceFreigeben(), die für die Freigabe des Bildspeichers sorgt, wenn dieser nicht mehr nötig ist.

Die übrigen Methoden der Klasse schließlich haben, wie oben schon erwähnt, dieselben Aufgaben wie ihr entsprechendes Pendant in der Klasse SVGFrage (Kapitel 4.3) bzw. wie in den Klassen der Bewertungskomponenten (Kapitel 3.1.4).

Klasse GIFBildKomponente und Klasse SVGBildKomponente

Bei den beiden anderen Klasse, die dazu dienen, dass ein Bild eines bestimmten Formats eingeladen wird, gibt es nun kaum noch Unterschiede im Bezug auf die Klasse JPGBildKomponente. So besteht der Unterschied der Klasse GIFBildKomponente nur darin, dass nun ein Bild des gif-Formats statt des jpg-Formats in die Lerneinheit eingebaut werden kann. Der Aufbau des Eingabefensters, so wie die Funktionsweise der datenVerarbeitung entspricht hingegen genau der Klasse JPGBildKomponente.

Die GIF-Bild-Komponente wurde dabei im Lernprogramm nicht eingesetzt, da alle Bilder des Lernprogramms entweder vom Format jpg oder vom Format svg sind. Die Komponente wurde im Rahmen dieser Diplomarbeit aber aus diesem Grund mit entwickelt, da das gif-Format neben dem jpg-Format in HTML-Seiten gerne Verwendung findet. Somit stehen dem Autor zwei wichtige Grafikformate für die Erstellung seiner Lernprogramme zur Verfügung.

Neben diesen zwei Grafikformaten hat der Autor zusätzlich noch die Möglichkeit, Bilder des Grafikformats svg einzubauen. Dies ermöglicht die Klasse SVGBildKomponente. Der Aufbau dieser Komponente ähnelt dabei dem Aufbau der beiden anderen Komponenten, besitzt aber geringfügige Unterschiede. Im Gegensatz zu den beiden anderen Komponenten wird bei der SVG-Bild-Komponente sofort beim Einladen der Datei überprüft, ob sie eine externe Datei überschreiben würde, und nicht erst in der Methode datenVerabeitung(). Außerdem besitzt der Autor hier nicht die Möglichkeit die Angaben zur Größe des Bildes zu verändern, sondern diese werden automatisch aus der svg-Datei ausgelesen und bleiben dann fest. Für die Anzeige des Bildes in der Kurzinformation schließlich muss der Autor noch ein jpg-Bild einladen. Ansonsten entspricht die Klasse SVGBildKomponente ebenfalls dem Aufbau der Klasse JPGBildKomponente (siehe oben).

Beispielhaft wurde dabei die SVG-Bild-Komponente im Lernprogramm auf der Kursstartseite eingesetzt, welche das Titelbild des Kurses beinhaltet. Dieses Titelbild zeigt dabei eine der Möglichkeiten des svg-Formats, indem verschiedene Grafikelemente animiert sind und dem Lernenden, in diesem Fall das Kind, für die Arbeit mit dem Lernprogramm motivieren soll.

Klasse SoundKomponente

Die letzte der Präsentationskomponenten stellt schließlich die Sound-Komponente dar, womit der Autor nun auch wav-Dateien in sein Lernprogramm einbauen kann. Im Beispiel des Lernprogramms zur Steigerung der Wahrnehmungsleistung sehbehinderter Kinder wurde diese Komponente dazu benutzt, dass das Kind die Möglichkeit hat, vor jeder Bildfrage eine Kurzanweisung zu dieser abspielen zu können.

Dabei entspricht der Aufbau, wie sollte es auch anders sein, im Wesentlichen dem Aufbau der Klasse JPGBildKomponente. Im Gegensatz zu dieser Klasse wird nun aber eine Datei mit dem Format wav eingeladen. Es wird hierbei, wie bei den anderen Verweiskomponenten auch, die Sounddatei nicht direkt eingeladen, sondern es wird ein Verweis auf die Sounddatei gespeichert, mit dessen Hilfe das Backend dann auf die richtige Datei zugreifen kann. Die Sounddatei befindet sich dabei in dem gleichnamigen Unterverzeichnis der entsprechenden Lerneinheit, nachdem sie von der Methode fertig stellen() dorthin gespeichert wurde.

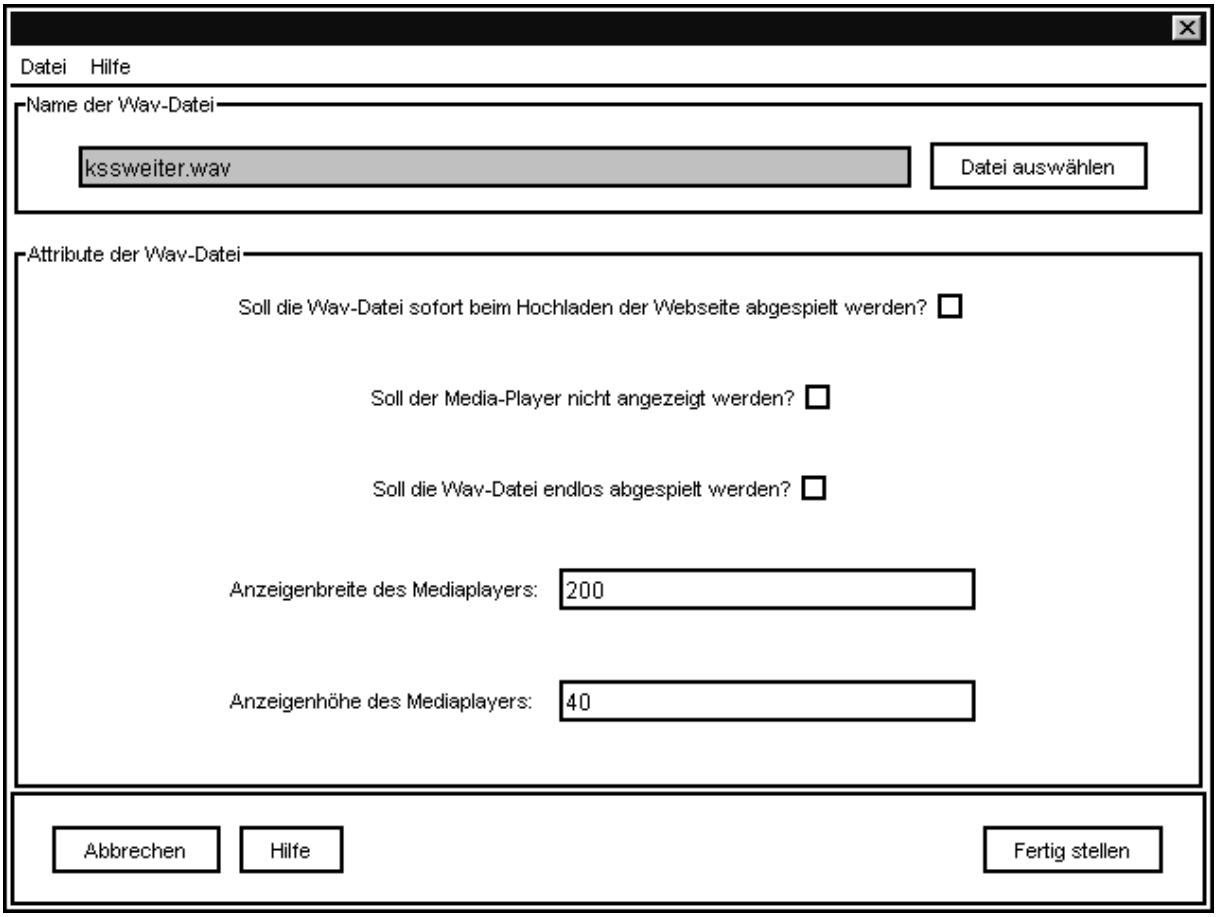

Abbildung 20: Eingabefenster der Sound-Komponente

In der Abbildung 20 ist nun das Eingabefenster der Sound-Komponente zu sehen, welches sich von den anderen Präsentationskomponenten unterscheidet, da die Sound-Komponente im Gegensatz zu diesen andere Attribute benötigt. Neben der Auswahl der Sounddatei selbst kann der Autor bei einer Sound-Komponente nun noch wählen, ob die wav-Datei sofort beim Hochladen der HTML-Seite abgespielt werden soll und ob diese wav-Datei endlos abgespielt werden soll. Daneben besitzt der Autor die Möglichkeit den Media-Player im Backend anzeigen zu lassen, damit es dem Lernenden selbst ermöglicht wird, die wav-Datei, so oft wie er will, abspielen zu können. Hierzu muss der Autor aber zusätzlich noch die Breite und Höhe für diesen Media-Player eingeben. Die inneren Klassen mausEingabeListener und EingabeListener wurden dabei so programmiert, dass automatisch die Checkbox für das sofortige Abspielen der wav-Datei beim Hochladen der HTML-Seite gesetzt wird, wenn der Autor die Checkbox aktiviert, dass der Mediaplayer im Backend nicht angezeigt werden soll. Dies geschieht deswegen, da ansonsten die Sound-Datei nie abgespielt werden könnte und damit sinnlos für die Lerneinheit wäre.
Die übrigen Funktionsweisen, wie u.a. die Überprüfung auf bereits existierende externe Dateien mit denselben Namen, entsprechen dabei genau den Funktionsweisen der Klasse JPGBildKomponente (siehe oben). Hierbei ist nur zu beachten, dass die Sound-Komponente kein Anzeigebild für die Kurzinformationen bzw. im Eingabefenster selbst verwalten muss, weswegen an diversen Stellen des Programmcodes in der Klasse SoundKomponente der Code für die Verwaltung dieser Bilder wegfällt.

5. Schlussbetrachtung

Im Rahmen dieser Diplomarbeit wurde ein Werkzeug erstellt, welches dem Autor auf eine einfach zu bedienende Weise ermöglichen sollte, den Ablauf eines Lernprogramms zusammenstellen zu können. Dabei wurde vor allem auch auf das Konzept der Wiederverwendung acht gelegt, damit der Autor seine erstellten Komponenten, Lerneinheiten bzw. Lektionen auch in anderen Lernprogrammen verwenden kann.

Als Beispiel für die Funktionsweise ist anschließend das Lernprogramm zur Steigerung der Wahrnehmungsleistung sehbehinderter Kinder erstellt worden, welches den Vorteil eines offenen Autorensystems aufzeigt, da für dieses Lernprogramm Komponenten entwickelt wurden, die sich speziell dafür eignen. Dabei können diese Komponenten mit Sicherheit auch in anderen Lernprogrammen eingesetzt werden.

Das Werkzeug zur Festlegung der Autorensteuerung, sowie das Lernprogramm mit den dazu entwickelten Komponenten befinden sich hierbei auf der beiliegenden CD.

Im folgenden werden nun ein paar Vorschläge gegeben, wie das Autorensystem noch erweitert werden könnte.

Zum einen könnte das Konzept der Feedbackseiten dahingehend verbessert werden, dass es dem Autor ermöglicht wird, verschiedene Präsentationskomponenten in diese einbauen zu können. D.h. die Rückmeldung an den Lernenden sollte nicht nur aus reinem Text bestehen, sondern es sollte auch möglich sein, dass in der Feedbackseite z.B. ein Bild mit angezeigt werden kann, bzw. dass eine Sounddatei abgespielt wird oder eine andere Präsentationskomponente dargestellt wird.
Hierzu müsste das Autorensystem dahingehend erweitert werden, das der Autor bei der Eingabe der Rückmeldung in den verschiedenen Komponenten die Möglichkeit besitzt, neben einem Text auch andere Präsentationskomponenten in die jeweilige Rückmeldung einfügen zu können.

Ein weiterer Erweiterungspunkt wäre die Einführung eines Options-Menü, welches den Autor bei der Zusammenstellen seines Lernprogramms behilflich ist. So wurde in dieser Diplomarbeit das Autorensystem so programmiert, dass der Autor den Ablauf seiner Lernprogramme so gestalten kann, wie er möchte. Er hat also die größtmögliche Freiheit und wird nur dahingehend eingeschränkt, dass er bei fehlerhaften Eingaben diese korrigieren muss. Diese Offenheit ist aber bei einigen Lernprogrammen eventuell nicht erwünscht. So kann es durchaus Lernprogramme geben, in denen keine Zyklen erlaubt sein sollen oder von vornherein bei den Lerneinheiten nur zwei Bewertungsbereiche erlaubt sind. Des weiteren sind noch andere Lernprogramme mit anderen Einschränkungen denkbar. Das Options-Menü könnte nun dem Autor die Möglichkeit bieten, diese Einschränkungen anzugeben, damit er selbst bei der Arbeit mit dem Autorensystem nicht darauf achten muss. Denkbar wäre hier auch, dass es sinnvolle Musterbelegungen der einzelnen Optionen gibt, die verschiedene Lernprogramme nachbilden.

Ein letzter Punkt schließlich liegt in der Erweiterung der Funktionalität der einzelnen Komponenten. So können beispielsweise neben den Größenattributen der jpg-Bilder in den HTML-Seiten noch weitere Attribute für das <img>-Tag gesetzt werden, die u.a. der Ausrichtung des Bildes dienen, bzw. andere Effekte besitzen. Für die Eingabe dieser Attribute könnte das Eingabefenster der JPG-Bild-Komponente dementsprechend erweitert werden, wobei es auch hier sinnvoll wäre, die einzelnen Attribute mit standardmäßigen Werten vor zu belegen. Ähnliches gilt auch für die Sound-Komponente, bei der es auch weitere Attribute zur Darstellung des Media-Players gibt.
Bei der SVG-Bild-Komponente schließlich würde es sich eventuell anbieten, einen internen Editor für das Erstellen des Bildes zur Verfügung zu stellen. Dies kann in der einfachen Version ein Texteditor sein oder in Rahmen eines größeren Projekts könnte ein Editor erstellt werden, der eine grafische Oberfläche besitzt. Für die Komponente SVGFrage wäre es dann noch sinnvoll einen Editor für die Eingabe des Javascripts anzubieten, damit der Entwickler einer Bildfrage, diese mit dem Autorensystem direkt erstellen kann.

Literaturverzeichnis

[Conradt] Conradt, T.: Entwicklung von Fragekomponenten zur Lernkontrolle von e-Learning-Kursen am Beispiel eines Kurses zum Erlernen der Blindenschrift, Diplomarbeit Nr. 2076, Universität Stuttgart 2003

[Co/Mü] Conradt, T.; Mühlleitner S.: Übersicht und Untersuchung existierender Autorensysteme und Entwurf für ein System zum Erstellen multimedialer Lernprogramme, Studienarbeit Nr. 1852, Universität Stuttgart 2002

[Frostig] Frostig, M.: Visuelle Wahrnehmungsförderung, Herausgeber: A. und E. Reinhartz, Arbeitshefte 1 + 2, Schroedel Schulbuchverlag GmbH, Hannover 1979

[Hall] Hall, M.: Core Servlets und Java Server Pages, München: Markt+Technik Verlag 2001

[Hanke] Hanke J.-C.: HTML/XML & Javascript, DATA BECKER GmbH & Co. KG, Düsseldorf 2000

[Herczeg] Herczeg, M.: Software-Ergonomie – Grundlagen der Mensch-Computer-Kommunikation, Bonn Paris Reading, Mass. [u.a.]: Addison-Wesley 1994

[Horst/Cor] Horstmann, C. S.; Cornell, G.: Core Java 2 Band 1 – Grundlagen, München: Markt + Technik Verlag 2001

[Horst/Cor2] Horstmann, C. S.; Cornell, G.: Core Java 2 Band 2 – Expertenwissen, München: Markt + Technik Verlag 2002

[McLau] McLaughlin, B.: Java und XML, Köln: O'Reilly Verlag GmbH & Co. KG 2001

[Mühlbradt] Mühlbradt, H.: Entwurf und Implementierung eines rechnerunterstützten Lernprogramms zum Steigern von Wahrnehmungsleistungen sehbehinderter Kinder, Studienarbeit Nr. 1655, Universität Stuttgart 1997

[Mühlbradt2] Mühlbradt, H.: Entwurf und Implementierung eines rechnerunterstützten Lernprogramms zum Steigern von Wahrnehmungsleistungen Sehbehinderter unter Verwendung von Text, Sprache, stehenden und bewegten Graphiken, Tönen, Geräuschen und Melodien, Diplomarbeit Nr. 1637, Universität Stuttgart 1998

[Seeboe] Seeboerger-Weichselbaum, M.: Das Einsteigerseminar XML, Landsberg: verlag moderne industrie Buch AG & Co KG 2001

[Seeboe2] Seeboerger-Weichselbaum, M.: Java/XML, Landsberg: verlag moderne industrie Buch AG & Co KG 2002

[Spona] Spona, H.: Das Einsteigerseminar SVG Webgrafiken mit XML, Landsberg: verlag moderne industrie Buch AG & Co. KG 2001