

Listen im Virtuellen Informatiklabor – Konzeption und Umsetzung einer E-Learning Lektion

Alena Schönberg

Zitiervorschlag im APA Stil:

Schönberg, A. (2012). *Listen im Virtuellen Informatiklabor – Konzeption und Umsetzung einer E-Learning Lektion* (pp. 2–115). Hochschule Offenburg.

Abstract

Gestaltung und Umsetzung einer Lektion (Thema: Listen) in der E-Learning Umgebung "Virtuelles Informatik-Labor". Theoretische Grundlagen aus E-Learning, Didaktik und User-Interface-Design.

Nutzungsbedingungen

Dieses Dokument wird unter diesen Bedingungen zur Verfügung gestellt:
Veröffentlichungsvertrag für Publikationen mit Print on Demand
Für weitere Informationen siehe:
</docs/Veroeffentlichungsvertrag.pdf>

Kontakt

Hochschule Offenburg | Bibliothek
Badstraße 24
77652 Offenburg
Telefon: (0781) 205-240
E-Mail: bibliothek@hs-offenburg.de
www.hs-offenburg.de/bibliothek

Bachelor-Thesis

LISTEN IM VIRTUELLEN INFORMATIKLABOR – KONZEPTION UND UMSETZUNG EINER E-LEARNING LEKTION

Alena Schönberg

Sommersemester 2012

Abgabe: 27.07.2012

Betreut durch

Prof. Dr. Volker Sanger und Prof. Dr-Ing. Claudia Schmidt

Medien und Informationswesen

Hochschule Offenburg

Badstrae 24

77652 Offenburg



Eidesstattliche Versicherung	Seite 4
1 Einführung	
1.1 Aufgabenstellung und Ziel der Thesis	Seite 5
1.2 Gliederung der Thesis	Seite 6
2 Grundlagen	
2.1 Das Virtuelle Informatiklabor	Seite 7
2.1.1 Grundgedanke des VIL	Seite 7
2.1.2 Aufbau des VIL	Seite 8
2.1.3 Unterstützende Elemente im VIL	Seite 9
2.1.4 Navigation und Interaktion im VIL	Seite 10
2.2 Listen in der Informatik	Seite 11
2.2.1 Lineare verkettete Listen	Seite 11
2.2.2 Verschiedene Anwendungen aus dem Alltag	Seite 11
2.2.3 Algorithmenbeispiele	Seite 12
3 E-Learning	
3.1 Allgemeines	Seite 15
3.1.1 Definition „E-Learning“	Seite 15
3.1.2 E-Learning Formen	Seite 15
3.1.3 Stärken und Schwächen des Online-Lernen	Seite 16
3.1.4 Das Problem „E-Learning“	Seite 17
3.2 Didaktik im E-Learning	Seite 18
3.2.1 Lernen und Lerntheorien	Seite 18
3.2.2 Lernziele und Lernziel-Taxonomien	Seite 21
3.2.3 Lernaufgaben und -szenarien	Seite 22
3.3 User-Interface-Design	Seite 24
3.3.1 Normen und Richtlinien	Seite 24
3.3.2 Usability	Seite 28
3.3.3 „gute“ E-Learning-Gestaltung zur intuitiven Interaktion	Seite 29
3.4 Vorgehen bei der Entwicklung	Seite 33
3.4.1 Vorbereitungen	Seite 33
3.4.2 Didaktische Konzeption	Seite 33
3.4.3 Erstellung eines Multimedia-Drehbuches	Seite 34

4 Änderungen & Erweiterungen für die Lektionen

4.1 Änderungsvorschläge	Seite 35
4.1.1 Rahmen und Navigation	Seite 35
4.1.2 Verschiedene Interaktionsmöglichkeiten	Seite 36
4.1.3 Texte und Textdesign	Seite 37
4.1.4 Prototypisches-Testen von Navigation und Interaktion	Seite 38
4.2 Umsetzung der neuen Navigation	Seite 40
4.2.1 Neuplatzierung der vorhandenen Elemente	Seite 40
4.2.2 Implementierung des Aufgabe/Bedienung-Fensters	Seite 41
4.2.3 Implementierung der neuen Navigation	Seite 42
4.3 Erstellen einer neuen Lektionsstruktur (theme.xml)	Seite 47

5 Lektion „Listen“ - Idee und Konzeption

5.1 Überblick über die Lektion	Seite 50
5.2 Inhalte der Lektion	Seite 50
5.2.1 Einführungsseite Listen	Seite 51
5.2.2 Ver-kette-t (Visualisierung/Instruktionsanwendung)	Seite 53
5.2.3 Rangierbahnhof (Experiment)	Seite 62
5.2.4 To-Do Liste (Struktogrammentwicklung)	Seite 69
5.2.5 Lexikoneintrag	Seite 77
5.2.6 Tool-Tip Texte	Seite 79

6 Lektion „Listen“ - Umsetzung

6.1 Einbinden der neuen Lektion	Seite 80
6.2 Einstiegsseite Listen	Seite 82
6.3 Lexikoneintrag	Seite 83
6.4 Ver-Kette-t (Visualisierung/Instruktionsanwendung)	Seite 85
6.5 ICE-Bahnhof (Experiment)	Seite 88
6.6 Usability-Test	Seite 92
6.7 Online stellen der neuen Inhalte	Seite 95

7 Zusammenfassung & Fazit

Seite 96

Danksagung	Seite 97
Abbildungsverzeichnis	Seite 98
Quellenverzeichnis	Seite 100
Anhang	Seite 104

Eidesstattliche Versicherung

Hiermit versichere ich eidesstattlich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

Gengenbach, den 25.07.2012

d. Schönberg

1 EINFÜHRUNG

1.1 Aufgabenstellung und Ziel der Thesis

Seit einigen Jahren wird die E-Learning Plattform des „Virtuellen Informatiklabors“ aufgebaut. In dieser Umgebung sollen alle grundlegenden Begriffe und Anwendungen der Informatik anschaulich dargestellt und erklärt werden.

Ziel dieser Bachelor-Thesis ist es, das Virtuelle Informatiklabor zu verbessern und eine neue Lektion zum Thema „Listen“ umzusetzen. Hierzu soll eine neue Lektion mit dem Thema „Listen“ konzipiert, umgesetzt und in das Virtuelle Informatiklabor integriert werden.

Die zu integrierenden Änderungs- bzw. Erweiterungsvorschläge basieren hierbei auf einer bereits durchgeführten Untersuchung im Rahmen einer Projektarbeit im vorangegangenen Semester (Analyse des Virtuellen Informatiklabors im Wintersemesters 2011/12; Simone Fehrenbach, Claudia Kriegeskorte, Diana Pohla, Alena Schönberg und Adrian Weigel). Die dort ermittelten Punkte sollen in die Konzeption der neuen Lektion einfließen.

Die Vorgehensweise bei der Änderung und Erweiterung orientiert sich an dem Layout der bereits vorhandenen Lektion und dem dort verwendeten Interaktionskonzept. Auch die dort verwendeten Rahmen und Grafiken sollen soweit möglich integriert werden.

Es soll eine Lexikon- und Übersichtsseite, sowie je eine Anwendung als Animation, Experiment und Struktogrammentwicklung konzipiert und mindestens die Animation umgesetzt werden. Alle Anwendungen sollen sich hierbei an einem Anwendungsbeispiel aus der Realität orientieren.

Im Theorie-Teil soll zudem ein Regelwerk für die „gute“ Gestaltung des User-Interfaces, zur intuitiven Interaktion mit einer E-Learning Anwendung, aufgestellt werden. Diese Vorlage soll die zukünftige Erstellung neuer Interaktionen im virtuellen Informatiklabor und anderen E-Learning-Anwendungen vereinfachen.

1.2 Gliederung der Thesis

Das erste Kapitel dieser Arbeit ermöglicht einen Überblick über die Aufgabenstellung und Inhalte der Thesis.

Im folgenden zweiten Kapitel werden die benötigten Grundlagen beschrieben. Hierzu wird das Virtuelle Informatiklabor vorgestellt. In diese bereits existierende E-Learning Umgebung soll die neue Listen-Lektion integriert werden. Zudem wird, zum besseren Verständnis des zu entwickelnden Inhaltes dieser Lektion, das Informatik-Thema „Lineare Listen“ kurz beschrieben und veranschaulicht.

Das dritte Kapitel bildet die theoretische Voraussetzung zum Entwickeln eines E-Learning Angebots. Hier werden die Themen Didaktik im E-Learning sowie verschiedene Richtlinien zur Gestaltung einer Anwendung erläutert. Diese Punkte werden im letzten Unterpunkt des Kapitels zu einer Vorgehensweise beim Erstellen eines E-Learning Angebotes verknüpft. In den Richtlinien zur Gestaltung findet sich auch das von mir erstellte Regelwerk zur Gestaltung „intuitiver Interaktion“.

Im vierten Kapitel werden die Änderungen und Erweiterungen für die Lektion beschrieben. Hier werden die in der Projektarbeit ermittelten Änderungsvorschläge vorgestellt und daraus abgeleitete Maßnahmen konzipiert. Im zweiten und dritten Unterpunkt des Kapitels können die verschiedenen Vorgehensweisen nachgelesen werden. Auch Anleitungen und Hinweise für das Erstellen neuer Lektionen und Einträge können hier entnommen werden.

Kapitel fünf beschreibt alle Inhalte der neuen Listen-Lektion. Die einzelnen Inhaltsseiten der Lektion, Einstiegsseite, Instruktionsanwendung, Experiment und Struktogrammentwicklung, auch der Lexikon-eintrag der Lektion werden hier genau beschrieben. Hierzu findet sich jeweils ein Überblick über die Anordnung der Elemente, die verschiedenen Interaktions- und Navigationsmöglichkeiten sowie die Konzeption der grafischen und textlichen Elemente der verschiedenen Inhalte.

Im sechsten Kapitel wird beschrieben, wie die in Kapitel fünf entworfenen Inhalte umgesetzt wurden. Die hier beschriebene Übersicht bietet einen Ansatzpunkt für kommende Erweiterungen in der E-Learning Umgebung des Virtuellen Informatiklabors.

Das abschließende Kapitel sieben bietet einen Rückblick auf die in der Thesis gewonnen Erkenntnisse.

2.1 Das Virtuelle Informatiklabor

Die in der Bachelor-Thesis zu entwickelnden Module entstehen im Rahmen des Virtuellen Informatiklabors (VIL). Um ein grundlegendes Verständnis für die in Kapitel 4 bis 6 beschriebenen Vorgehensweisen und Kommentare zu garantieren, soll das VIL hier kurz vorgestellt werden.

2.1.1 Grundgedanke des VIL

Das Virtuelle Informatiklabor ist eine interaktive webbasierte E-Learning-Umgebung der Hochschule Offenburg. Sie wurde von Prof. Dr.-Ing. Claudia Schmidt, unter der Mitarbeit von Prof. Dr. Volker Sanger, entwickelt.

Das VIL entstand vor allem fur Schulerinnen und Studentinnen. Es soll bei dieser Zielgruppe das Interesse fur die Informatik wecken und dazu beitragen, dass der immer noch vorherrschende groe Respekt vor diesem Thema abnimmt. (vgl. SCHMIDT (2011) S. 4)

Der grundlegende Ansatz zur Umsetzung liegt hier in einer an der konstruktivistischen Lerntheorie (s. Kapitel 3.2.1 Lernen und Lerntheorien) angelehnten Lernumgebung. Die verschiedenen Themen der Informatik sollen selbststandig entdeckt und erforscht werden.

Die Lernthemen im VIL sollen alle grundlegenden Algorithmen der Informatik umfassen. Hierbei steht jedoch nicht das theoretische Wissen, sondern die Relevanz der Informatik in der Praxis und dem taglichen Leben im Vordergrund. (vgl. SCHMIDT (2011) S. 4) So werden alle Themen anhand eines Beispiels erklart, das moglichst allen Lernenden aus dem eigenen Alltag bekannt ist, z.B. die Losung des Spiels „Turme von Hanoi“ zur Erklarung der Rekursion.

Dieses Ziel soll durch eine motivierende, zielgruppengerechte Gestaltung unterstutzt werden, indem die Anwendung jugendlich, frech, grafisch ansprechend und interaktiv gestaltet wird. (vgl. SCHMIDT (2011) S. 4)

2.1.2 Aufbau des VIL (vgl. SCHMIDT (2011) S. 31ff und VIL (2012))

Laborraum

Der Einstieg in die E-Learning-Anwendung geschieht über eine Raum-Metapher. Im „Laborraum“ können die Lernenden sich virtuell frei bewegen und die verschiedenen Themenfelder entdecken. Die zu entwickelnde Lektion „Listen“ findet man z.B. an der Pinnwand („abstrakte Datentypen“) wenn man die Abbildung des Merktzettels (oben, Mitte) anwählt.



Abb. 1 Laborraum des VIL (Quelle: VIL(2012))

Einstiegsseite

Wenn einer der Themenbereiche im Laborraum angewählt wird, öffnet sich die Einstiegsseite der jeweiligen Lektion. Hier wird das Thema inhaltlich kurz vorgestellt, wichtige Begriffe erläutert und Beispiele aus der Praxis genannt um das Thema zu veranschaulichen. Ziel dieser Seite ist es, den Lernenden einen motivierenden Einstieg in die Lektion zu bieten.

Durch die Lernweg-Navigation gelangt man innerhalb der Lektion zunächst auf die Einstiegsseite, gefolgt von der Animation, dem Experiment und der Struktogrammentwicklung.

Animation

Die Animation als „Instruktionsanwendung“ soll das Verstehen der vorgestellten Algorithmen fördern. Um dies zu erreichen wird, neben der Visualisierung eines Alltagsproblems als Animation (linkes Fenster), der zur Lösung des Problems verwendete Algorithmus angezeigt und parallel durchlaufen (rechtes Fenster). Dabei kann die Animation interaktiv über ein Kontrollpanel gesteuert werden.

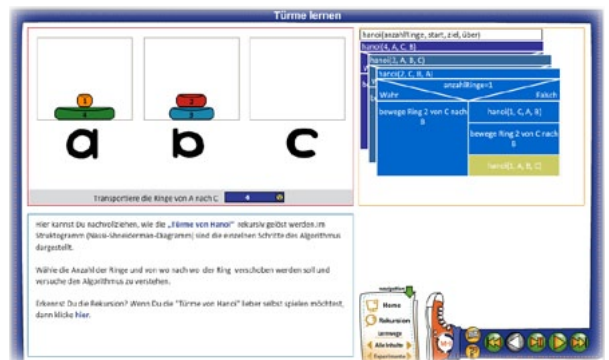


Abb. 2 Animation „Türme von Hanoi“ im VIL (Quelle: VIL(2012))

Experiment

Der vorgestellte Algorithmus kann nun auf ein konkretes Beispiel, eine vorgegebene Aufgabe, angewandt werden. Diese kann durch Ausprobieren und durch Anwenden des angegebenen Algorithmus gelöst werden. Durch eine individuelle Konstruktion kann der Algorithmus erprobt, getestet und somit besser verstanden werden.

Struktogrammentwicklung

Auch in dieser „Konstruktionsanwendung“ soll eine vorgegebene Aufgabe gelöst werden. Durch die selbstständige Algorithmenentwicklung, in Form eines zu entwerfenden Struktogramms, kann das in den vorangegangenen Schritten erarbeitete Wissen angewendet werden.

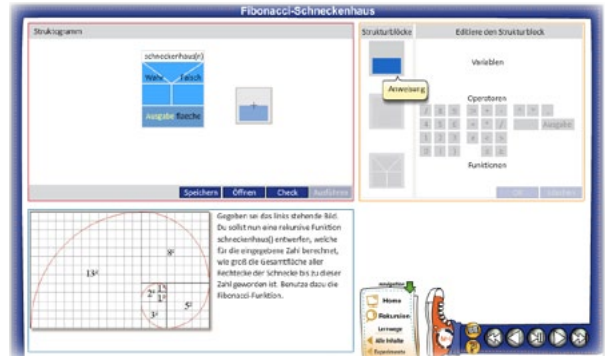


Abb. 3 Struktogramm „Schneckenhaus“ im VIL
(Quelle: VIL(2012))

2.1.3 Unterstützende Elemente im VIL

Nachdem der generelle Aufbau des VIL kurz beschrieben wurde, werden nun noch die Rückmelde- und Hilfefunktionen vorgestellt (vgl. SCHMIDT (2011) S. 24ff und VIL (2012)), welche die Lernenden unterstützend durch die Lektionen begleiten.

MIA

MIA („MI-Assistentin“) ist der fiktive Guide, der durch die Lektionen des VIL führt. Dabei gibt sie Hilfestellungen und korrigierende sowie motivierende Rückmeldungen zu den einzelnen Anwendungen.



Abb. 4 MIA, Guide des VIL (Quelle: VIL(2012))

Lexikon

Zusätzlich zu MIA werden auch statische Hilfe-Elemente verwendet. So ist z.B. ein Lexikon, in Form eines virtuellen Buches, zum Nachschlagen weiterer Informationen implementiert.

Hilfebutton

Eine zusätzliche Hilfe ist über den Hilfebutton zu erreichen. Über diesen kann man direkt in das relevante Lektions-Kapitel des Lexikons springen, sich den verwendeten Algorithmus erklären lassen oder Hinweise zur Lösung der gestellten Aufgabe erhalten.

Tool-Tipps

Tool-Tipps werden zur kurzen Erklärung von Begriffen oder Werkzeugen verwendet. Durch die Verwendung der Tool-Tipps wird der Aufgabentext nicht unnötig unterbrochen und dient so zur Wahrung des Überblicks.

2.1.4 Navigation und Interaktion im VIL

Gerade die Navigation und Interaktion spielen im Rahmen dieser Thesis eine sehr große Rolle. Um einen Überblick über die verschiedenen Elemente zu erhalten, werden die beiden Navigationsmöglichkeiten sowie die Interaktion mittels Kontrollpanel und Toolbox kurz vorgestellt. (vgl. SCHMIDT (2011) S. 20f und VIL (2012))

Wäscheschild-Navigation (Abb. 5, links unten)

Über diese Navigation gelangt man zurück zur Laborseite („Home“). Es lassen sich auch die verschiedenen Lektionen aufrufen; über die Lernwege gelangt der Lernende entweder nacheinander zu allen Inhalten der Lektion („Alle Inhalte“) oder lediglich zu den Konstruktionsanwendungen („Experimente“). Der zweite Weg ist für Lernende mit Vorwissen gedacht, da die Einstiegs- und Animationsseiten hier übersprungen werden.

Lupen-Navigation (Abb. 5, rechts oben)

Über das Wäscheschild ist auch die zweite Navigationsmöglichkeit, die Lupe (unter dem jeweiligen Namen der Lektion, in Abb. 5 z.B. „Rekursion“) zu finden. Von der Lupe aus sind alle Seiten der Lektion direkt anwählbar. Die einzelnen Themenbereiche sind hier in einer Baumstruktur präsentiert. Durch das Anwählen des gewünschten Themengebiete lassen sich die zugehörigen Anwendungen anzeigen und aufrufen.

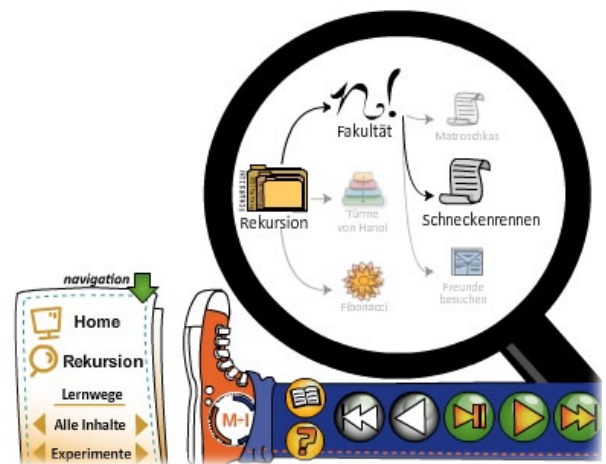


Abb. 5 Navigation und Kontrollpanel im VIL
(Quelle: VIL(2012))

Kontrollpanel (Abb. 5, rechts unten)

Über das Kontrollpanel können die Animationen gesteuert sowie die Experiment-Anwendungen wieder zurückgesetzt werden. Für die Animation bestehen hierbei fünf Interaktionsmöglichkeiten (Play/Pause, Schrittweise vorwärts bzw. zurück, Startposition, Endposition). Bei Experimenten funktioniert demnach nur der Button für „Startposition“. Auf Einführungsseiten wird das Kontrollpanel nicht angezeigt.

Toolbox zur Struktogrammentwicklung

Die Toolbox dient zum Erstellen der Struktogramme. Hier werden alle Elemente, die in das Struktogramm integriert werden können, zur Verfügung gestellt. Dazu gehören die einzelnen Strukturblöcke (Anweisung, Schleife und Verzweigung) sowie die Variablen, Operatoren und Funktionen.

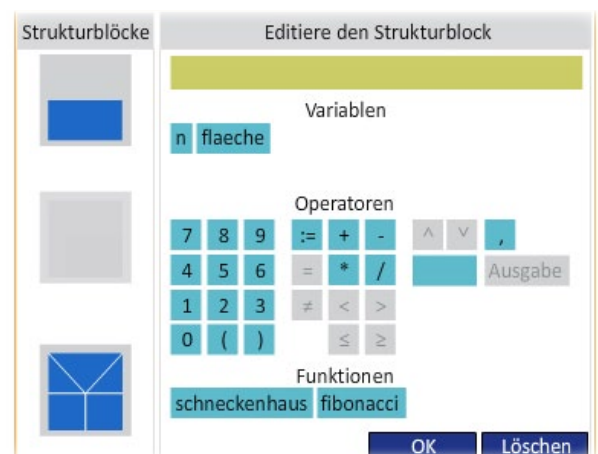


Abb. 6 Tool Box des VIL (Quelle: VIL(2012))

2.2 Listen in der Informatik

2.2.1 Lineare verkettete Listen

Die „Listen in der Informatik“ umfassen viele verschiedene Arten der Anwendung des Listenprinzips. Da im Rahmen des VIL die grundlegenden Elemente der Informatik vorgestellt werden, beschränkt sich die zu entwickelnde Listen-Lektion auf die sogenannten „linearen verkettenden Listen“ (vgl. SAAKE (2006) S. 307, BRÜGGE (2000) S. 23, 25, 28 und LETSCHERT (2002) S. 2, 11). Anhand dieser Listenart kann das Grundverständnis zur Funktion und Umgangsweise, auch mit allen weiteren Listenarten, erlernt werden.

Die erste Datenstruktur, die Lernenden in der Informatik vorgestellt wird, ist der „Array“ bzw. das „Feld“ (vgl. RÜDEBUSCH (2012) S. 39f). In einem Array kann eine zuvor festgelegte, begrenzte Anzahl an Elementen gespeichert und über den Index des Speicherortes wieder erreicht werden.

„Listen“ hingegen sind eine dynamische Datenstruktur. Sie können zur Laufzeit des Programms beliebig vergrößert (neue Elemente speichern) oder verkleinert (Elemente löschen) werden und passen sich so dem jeweiligen Speicherbedarf optimal an.

Der Aufbau einer Liste lässt sich am besten durch ein Schaubild verdeutlichen:



Abb. 7 Visualisierung einer linearen Liste (eigener Entwurf)

Die verkettete Liste besteht aus mehreren Elementen („Knoten“), die miteinander verkettet sind. Jedes Element besteht aus seinen spezifischen Daten (Rechteck) und einer Referenz bzw. einem Verweis (Pfeil) auf das Folgeelement. Das letzte Element der Liste wird durch den Verweis auf „NULL“ (kein Element) gekennzeichnet.

Um auf die komplette Liste zugreifen zu können, muss demnach nur das erste Element der Liste durch einem Anker-Verweis („Head“ oder „Kopf“) referenziert werden. Alle anderen Elemente können über das Navigieren über die Verweise, also das von vorne durchlaufen der Liste, erreicht werden. Wenn z.B. das zweite Element der Liste aufgerufen werden soll, lautet die Anweisung für die Listennavigation „Gehe an den Anfang (Head) der Liste, dann gehe zum nächsten Element“. Ähnlich einer Einbahnstraße kann bei einer linearen Liste jedoch nicht zum vorherigen Element zurückgekehrt werden.

2.2.2 Verschiedene Anwendungen aus dem Alltag

Eine dynamische Liste kann in vielen alltäglichen Situationen hilfreich sein und Anwendung finden. Da im VIL Erklärungen anhand von Praxisbeispielen gegeben werden, sollen nun einige Beispiele aus dem Bereich der Listen vorgestellt werden.

Beispiel „Partybesuch“ (vgl. BRÜGGE (2000) S. 23):

Simon ist auf einer Party muss aber nun nach Hause, er möchte seinem Kumpel Max aber noch eine wichtige Nachricht zukommen lassen. Das Problem ist, das Simon nur weiß wo Tom gerade ist. Tom weiß jedoch wo Susanne ist und Susanne weiß wo Max ist. Um Max die Nachricht zukommen zu lassen muss

Simon sie also an Tom weitergeben, der sie dann an Susanne gibt, die sie an Max weitergeben kann. Dies ist das Konstrukt einer typischen Liste: Simon → Tom → Susanne → Max.

Beispiel „Studentenverzeichnis“ (vgl. BRÜGGE (2000) S. 24):

Hier soll eine Liste, geordnet nach dem Namen der Studierenden, alle Studenten einer Vorlesung verwalten. Fünf Studenten melden sich nacheinander an (in der Reihenfolge: Thomas, Julia, Alex, Florian, und Hannah), einer besucht die Vorlesung dann doch nicht (Florian).

Die Liste würde sich dann Schrittweise auf- bzw. abbauen: zuerst nur (Thomas), dann (Julia → Thomas), dann (Alex → Julia → Thomas), dann (Alex → Florian → Julia → Thomas), dann (Alex → Florian → Hannah → Julia → Thomas) und schließlich (Alex → Hannah → Julia → Thomas).

Natürlich funktioniert diese Art der Beschreibung für alle geordneten Verzeichnisse oder Listen im Alltag; sei es eine Kundenliste (nach Name/Kundennummer geordnet), Telefonnummernliste (nach Name geordnet) oder To-Do-Liste (nach Datum der Fertigstellung/ Fälligkeit geordnet).

Beispiel „Rangierbahnhof“ (vgl. SCHWAIGER (2009) S. 123):

Auf einem Rangierbahnhof müssen Züge und Zugketten neu zusammengestellt oder getrennt werden. Wenn hier z.B. ein neuer Wagen (Nr. 4) in eine bestehende „Wagenkette“ (Wagen Nr. 1 → Wagen Nr. 3 → Wagen Nr. 6) aufgenommen werden soll, laufen ähnliche Vorgänge wie beim Einfügen in eine Liste ab. Der Wagen Nr. 6 muss hier zwischen geparkt werden, dann kann der Wagen Nr. 4 an den Wagen Nr. 3 gehängt werden und der Wagen Nr. 6 wieder an das Ende der Wagenkette.

2.2.3 Algorithmenbeispiele

Die Implementierung der Interaktion mit einer Liste soll nun exemplarisch dargestellt werden. Hierzu soll das Suchen, Einfügen und Löschen eines Listenelements aufgezeigt werden. Anhand dieser drei Vorgänge und deren Kombination lassen sich alle Listenprobleme (z.B. das Sortieren von Listen) lösen.

Zur Veranschaulichung dient das Beispiel einer To-Do-Liste, die nach der Fälligkeit der Elemente aufsteigend sortiert ist.

Die bestehende To-Do-Liste:

- Aufräumen (noch 2 Tage)
- Lernen (noch 4 Tage)
- Referat (noch 5 Tage)
- Hausarbeit (noch 10 Tage)

Jedes Element dieser Liste besteht aus folgenden Werten:

- Name (z.B. „Aufräumen“)
- Fälligkeit (z.B. noch „2“ Tage)
- nächstes Element (z.B. Pfeil zu „Lernen“)

Auf das erste Element, hier „Aufräumen“, kann über den Verweis „erstesElement“ (Head) zugegriffen

werden. Alle weiteren Elemente können nur durch das Durchlaufen der Liste vom „ersten Element“ aus erreicht werden.

2.2.3.1 Suchen in einer Liste

Für das Suchen muss die Liste vom ersten Element beginnend durchlaufen werden, bis das gewünschte Element gefunden oder das Ende der Liste erreicht ist. Es soll z.B. herausgefunden werden ob ein Eintrag mit einem bestimmten Namen vorhanden ist.

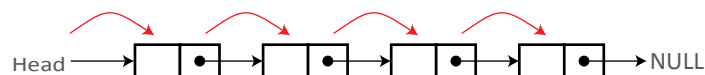


Abb. 8 Visualisierung – Suchen in einer Liste (eigener Entwurf)

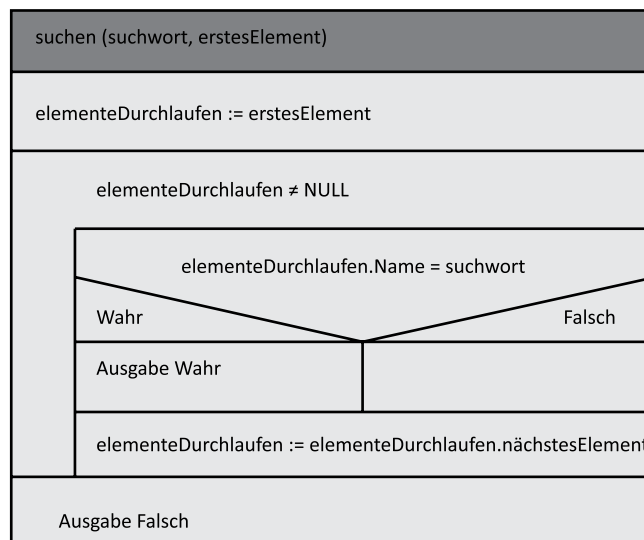


Abb. 9 Struktogramm – Suchen in einer Liste (eigener Entwurf)

2.2.3.2 Einfügen in eine Liste

Das Einfügen setzt eine vorgelagerte Suche voraus. Hier muss die Lücke gefunden werden, in die das neue Element eingefügt werden soll. Um dann ein neues Element einzufügen müssen beide Elemente, das Element vor und nach der Lücke, gespeichert werden. Das neue Element wird mit dem vor der Lücke verknüpft, dann wird das nachfolgende Element mit dem neu eingefügten verbunden.

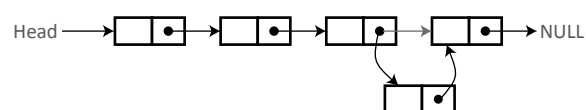


Abb. 10 Visualisierung – Einfügen in eine Liste (eigener Entwurf)

Um das eigentliche Einfügen zu veranschaulichen ist hier das Element, das sich direkt vor dieser Lücke befindet, durch den Verweis „vorherigesElement“ gegeben.

einfügen (vorherigesElement, neuesElement, head)	
vorherigesElement = NULL	
Wahr	Falsch
neuesElement.nächstesElement := head	merkVerweis := vorherigesElement.nächstesElement
head := neues Element	vorherigesElement.nächstesElement := neuesElement
	vorherigesElement.nächstesElement.nächstesElement := merkVerweis

Abb. 11 Struktogramm – Einfügen in eine Liste (eigener Entwurf)

2.2.3.3 Löschen aus einer Liste

Das Löschen setzt, wie das Einfügen auch, eine vorangegangene Suche des „zu löschenden Elements“ voraus. Hier muss der „Nachfolger“ des zu löschenden Elements mit dem „Vorgänger“ verknüpft werden. Da kein Weg mehr zu dem „zu löschenden Element“ führt ist dieses gelöscht.

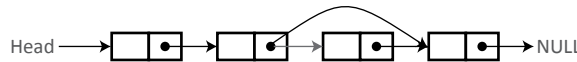


Abb. 12 Visualisierung – Löschen aus einer Liste (eigener Entwurf)

Um das eigentliche Löschen zu veranschaulichen ist hier der Vorgänger des Elements, durch den Verweis „vorgängerElement“ gegeben.

löschen (vorgängerElement, löschelement, head)	
vorgängerElement = NULL	
Wahr	Falsch
head := löschelement.nächstesElement	vorgängerElement.nächstesElement := löschelement.nächstesElement

Abb. 13 Struktogramm – Löschen aus einer Liste (eigener Entwurf)

3.1 Allgemeines

3.1.1 Definition „E-Learning“

Das VIL ist eine E-Learning Anwendung. Da es viele verschiedene Formen dieses Lernens gibt, sollen hier die Hintergründe zu diesem Thema erklärt werden, um die E-Learning Umgebung VIL richtig einordnen zu können.

E-Learning („electronic learning“) (vgl. ALTBAUER (2012b) S.1, RZAZA (2012a), LIPINSKI (2012) und KLIMSA (2011) S. 14, 62) ist der Oberbegriff für alle Formen des elektronisch unterstützten Lernens. Im allgemeinen Sprachgebrauch wird darunter jedoch eher das computergestützte Lernen oder nur das Online-Lernen verstanden.

Eine allgemeine Definition findet sich z.B. in LIPINSKI (2012): „E-Learning ist die interaktive Kommunikation zwischen dem wissensvermittelnden Programm und dem Lernenden, in aller Regel eine Einzelperson.“ Ein E-Learning-Angebot bietet hierbei Lernmaterial in ausschließlich digitaler, meist multimedialer, Form.

Was hier in wenigen Worten zusammengefasst werden kann umfasst ein interdisziplinäres Themengebiet bestehend aus Bildungsforschung, Medien-/Didaktik, Gestaltung, Informatik, Psychologie, Medienproduktion, Rechtswissenschaften, Wirtschaftswissenschaften und der jeweiligen Fachdisziplinen.

3.1.2 E-Learning Formen

Nachdem E-Learning allgemein definiert wurde, soll ein Einblick in verschiedenen Unterarten gegeben werden. Da es ein großes Spektrum von E-Learning Formen gibt, werde ich im Folgenden auf die drei wichtigsten organisierten E-Learning-Angebote eingehen (vgl. KLIMSA (2011) S. 14, LIPINSKI (2012), ALTBAUER (2012b) und WEBER (2008)). Formen die durch die Eigeninitiative von Lernenden entstehen werden somit ausgeklammert.

Offline Lernen

Computer Based Training (CD/DVD)

Unter dem Computer Based Training versteht man die klassische Lern-CD oder -DVD. Sie dient vor allem dem Selbststudium, kann flexibel verwendet und unabhängig von Lehrenden oder anderen Lernenden genutzt werden. Ein Vorteil dieser E-Learning-Variante ist, dass viele Inhalte multimedial aufbereitet werden können, ohne sich an Bandbreitenbeschränkungen zu orientieren. Nachteilig ist hingegen, dass die Aktualität des Inhalts nicht automatisch gesichert ist und meist teuer nachgekauft werden muss.

Online Lernen

Web Based Training (Internet- oder Intranetbasiert)

Das Web Based Training stellt die Weiterentwicklung des computergestützten Lernens dar. Hier kann auf

Lerninhalte im Internet zugegriffen und mit ihnen gelernt werden. Auch in dieser Form findet die Wissensvermittlung meist auf Grundlage eines Selbststudiums statt. Als Erweiterung zum Offline Lernen ist vor allem die Möglichkeit der Kommunikation zwischen den Lernenden und/oder Lehrenden zu nennen, sowie die Möglichkeit die dargebotenen Inhalte jederzeit zu aktualisieren.

Blended Learning (Vermischtes Lernen)

Blended Learning bezeichnet Lernen in einer Kombination aus Online-Lernen und Präsenzteil (Schulunterricht, Vorlesung, Seminar etc.). Hierbei sollen beide Lernmethoden die jeweiligen Stärken ausspielen. So wird die Onlinephase meist für die Vor- und Nachbereitung des Lerninhalts verwendet. In der Präsenzphase können die gelernten Inhalte dann durch Übungen und Anwendungen gefestigt sowie durch Diskussionen reflektiert werden.

3.1.3 Stärken und Schwächen des Online-Lernen

Um einen umfassenden Einblick in die Thematik des E-Learning zu erhalten ist es wichtig die Stärken und Schwächen dieser Anwendungen im Vergleich zu einer Präsenzveranstaltung (vgl. RZA (2012b), SCHOLZ (2012), WEBER (2008) und ARNOLD (2011) S. 45-47) zu kennen. Da im Rahmen dieser Arbeit das Online-Lernen im Vordergrund steht, sollen diese Aspekte erklärt werden. Die Berücksichtigung verschiedener Sichtweisen erleichtern dem Entwickler einen optimalen Entwurf eines Angebotes..

Zu den größten Stärken des E-Learning gehört sicherlich die Zeit- und Ortsunabhängigkeit des Lernens und Lehrens. Hierdurch können z.B. Fahrtkosten eingespart werden und eine angenehme Lernatmosphäre, durch eine gewohnte Umgebung, geschaffen werden.

Der Lernende kann sein eigenes Lerntempo beibehalten, sich seine Zeit frei einteilen und gelangt schnell zu den für ihn individuell relevanten Inhalten. Nicht verstandene Inhalte können beliebig oft wiederholt werden, ohne sich an andere Lernende anpassen zu müssen.

Es ist auch möglich durch E-Learning eine größere Teilnehmeranzahl zu erreichen, da es prinzipiell keine Einschränkung der Anzahl an Lernenden gibt. Die Inhalte können zudem schnell aktualisiert und personalisiert werden.

Natürlich gibt es auch Nachteile beim E-Learning. Hier ist das Kosten-Nutzen-Verhältnis zu nennen das oft, aufgrund der Vielzahl an verschiedenen Angeboten, nicht klar ersichtlich ist.

Doch auch für die Lernenden selbst kann eine E-Learning-Anwendung eine Herausforderung sein. So fordert es einen hohen Grad an Selbstdisziplin und Eigeninitiative, vor allem im Vergleich zu Präsenzveranstaltungen. Auch der fehlende persönliche Kontakt zu anderen Lernenden oder Lehrenden kann sich negativ auswirken. Es kommt meist zu weniger Erfahrungsaustausch und je nach E-Learning Form können Fragen ungeklärt und Anmerkungen ungeachtet bleiben.

Technische Hindernisse oder Probleme können zudem demotivierend wirken und stark behindern. Auch eine eventuelle Überforderung durch die Komplexität und Vielfalt des Lernangebots muss vermieden werden.

3.1.4 Das Problem „E-Learning“

Die Schwächen des Online-Lernen haben leider dazu geführt, dass E-Learning bei vielen Menschen mit einer negativen Erfahrung verbunden ist. Auslöser dieser ablehnenden Haltung sind vor allem die vielen gescheiterten Projekte in diesem Bereich, bei denen sich die Lernenden überfordert fühlten.

Nach ARNOLD (2011) und vielen weiteren Untersuchungen (z.B. bestätigt durch die Ergebnisse im Rahmen der Projektarbeit zur Analyse des VIL, vgl. FEHRENBACH (2012)) gibt es jedoch einen klaren Grundsatz in der Entwicklung einer E-Learning-Anwendung:

„Überall, wo die bildungstechnologischen Innovationen als Medium – und nicht als Ersatz von Menschen – im pädagogischen Verhältnis zwischen Lehrenden und Lernenden dienen, sind sie keineswegs misslungen, sondern tragen unbestritten zur Verbesserung der Qualität, der Wirksamkeit und Effizienz des Lehrens und zu einem motivierenden und erfolgreichen Lernen bei.“ (ARNOLD (2011) S. 27)

Werner Stangl präsentiert hierzu die „10 goldenen Regeln des E-Learning“ (nach ALTHBAUER (2012b) S. 8, vgl. KLIMSA (2011) S. 265-268 und ARNOLD (2011) S. 27, 296-298, 306-307), die eine erfolgreiche E-Learning-Umgebung hervorbringen sollen. Diese Regeln werden von Aussagen vieler weiterer E-Learning-Autoren unterstützt und ergänzt:

Der Lernende soll ...

- zwischen verschiedenen Lernstilen wechseln können
- individuell und kooperativ lernen können
- durch menschliche Mentoren bzw. Tutoren betreut werden (Vereinbarung der Ziele, Inhalte und Methoden zwischen Lernenden und Lehrenden; individuelle Betreuung bei Fragen und Anmerkungen)
- wenn nötig aktiviert werden und aktiv gehalten werden
- jederzeit seinen Wissensstand überprüfen können
- alle 20 bis 30 Minuten ein Erfolgserlebnis haben
- eigene Wissensbausteine erstellen können

Der Lernstoff soll ...

- multimedial an den Inhalt angepasst werden (das Bildungsproblem sollte im Fokus stehen „problem driven“ und nicht die technischen Möglichkeiten „technology driven“, es gilt das Potenzial der Medien richtig zu nutzen)
- sinnvoll intern und sparsam extern verlinkt sein
- aktuell gehalten werden

Das Konzept sollte zudem insgesamt einen Mehrwert gegenüber herkömmlichen Angeboten bieten und nicht nur bereits vorhandene Materialien wiederverwenden.

Um diesen und weiteren Problemen vorzubeugen, sollen nun die didaktischen und gestalterischen Grundlagen für die Entwicklung einer E-Learning-Umgebung aufgezeigt werden.

3.2 Didaktik im E-Learning

Didaktik ist die Wissenschaft vom Unterrichten. Sie versucht die Prozesse von lernen und lehren zu beschreiben, unabhängig vom jeweiligen Lerninhalt. (vgl. ALTHBAUER (2010a))

Auch im Erstellen von E-Learning-Anwendungen muss eine didaktische Grundlage geschaffen werden. Die Gebiete der Lerntheorien und die Formulierung von Lernzielen sollen nun, im Bezug auf die Anwendung auf E-Learning Module, vorgestellt werden. Im abschließenden Unterkapitel wird darauf aufbauend ein Ablauf zur Erstellung von E-Learning-Anwendungen skizziert.

3.2.1 Lernen und Lerntheorien

Lernen bezeichnet alle Veränderungen des Verhaltens, Denkens oder Fühlens, die durch eine vorausgegangene Erfahrung angestoßen wurden. Es ist die Fähigkeit auf die Umwelt effizient zu reagieren und umschließt einen Prozess aus aufnehmen, einordnen und bereithalten von Wissen jeder Art. (vgl. ALTHBAUER (2010b))

Wie diese Definition schon andeutet ist lernen schwer zu definieren; es ist ein individueller Prozess. So lassen sich z.B. verschiedene Lerntypen voneinander abgrenzen (visuelle, auditive, kommunikative oder kinästetische Lerntypen) (vgl. ALTHBAUER (2011a)). Auch Eigenschaften wie Vorwissen zum Lernthema oder räumliches Vorstellungsvermögen beeinflussen den jeweiligen Lernprozess (vgl. REY (2009a)).

Hinzu kommen auch Lernstrategien, also Vorgehensweisen wie gelernt wird. Jeder Mensch verfügt über verschiedene Strategien, welche bewusst oder auch unbewusst angewendet werden. Diese Strategien können sowohl die eigentliche Informationsaufnahme und -verarbeitung betreffen, oder auch Rahmenbedingungen wie die Lernumgebung und die Planung der Lernaktivität. (vgl. ALTHBAUER (2011b))

Als weiterer Punkt ist noch die eigene Motivation zum Lernen zu nennen. Hier gibt es vor allem drei wichtige Aspekte: die Leistungsmotivation (lernen um einen gewissen Maßstab zu erreichen), den Erreichbarkeitsgrad (Erfolgswahrscheinlichkeit) und der eigentliche Anreiz der gestellten Aufgabe. (vgl. ALTHBAUER (2012a))

Lerntheorien

Lerntheorien (vgl. ARNOLD (2011) S. 101, 110-111, REINMANN (2011) S. 1-2, 5, HESSE (2010) und REY (2009b)) wurden entwickelt, um das Lernen trotz dieser Fülle an Aspekten zu beschreiben. Sie versuchen den individuellen Prozess des Lernens bzw. der Wissensaneignung möglichst allgemein zu beschreiben und bilden somit das Fundament zum Verständnis von Lehren und Lernen.

Obwohl die Lerntheorien keine direkte handlungspraktische Relevanz haben ist es sinnvoll, sich einen Überblick über deren verschiedene Aspekte zu verschaffen. Durch diese Erklärungsansätze können Rahmenbedingungen für die didaktische Konzeption erstellt und fundierte Entscheidungen getroffen werden.

Da es keine einheitliche Lerntheorie gibt, die alle denkbaren Lernformen und Aspekte des Lernens beschreiben oder erklären könnte, wird in der Literatur oft zwischen drei Hauptströmungen unterschieden:

3.2.1.1 Behaviorismus

Im behavioristischen Lernansatz (vgl. ARNOLD (2011) S. 101-102, ARNOLD (2005) S. 2-3, 6-7, REINMANN (2011) S. 3, REY (2009c)) wird das Entstehen von Wissen durch ein Reiz-Reaktions-Modell erklärt. Hierbei werden objektive Fakten (Wissen) durch die Reaktion eines Individuums auf einen Umweltreiz gelernt. Es werden nur sichtbare Verhaltensänderungen betrachtet, innere Denk- und Verstehensprozesse wie z.B. Denken, Wahrnehmen und Problemlösen werden ausgeblendet (Black-Box-Denken).

Der Behaviorismus unterstellt, dass Verhalten geformt werden kann indem Belohnungen (positiver Reiz oder das Entfernen eines negativen Reizes) oder Sanktionen (negativer Reiz oder das Entfernen eines positiven Reizes) entsprechend dem gewünschten Verhalten eingesetzt werden.

Die Kritik am behavioristischem Modell beruht vor allem auf der Gewinnung der Erkenntnisse. Da sich die Theorie ausschließlich auf Tierexperimente und Laborsituationen stützt, hat sie aus heutiger Sicht kaum noch Relevanz für menschliche Lernvorgänge. Trotz der mangelnden Zustimmung wird dieser Ansatz in der mediendidaktischen Praxis wirkungsvoll angewandt.

Behaviorismus im E-Learning

Ein E-Learning-Programm auf der Basis des Behaviorismus ist geeignet um einfache Lernziele zu erreichen, z.B. die Aneignung von Faktenwissen.

Hier wird ein lineares Lernprogramm mit einer vorgegebenen Reihenfolge im Lernmaterial eingesetzt, z.B. Vokabelprogramme, Tutorials oder Drill-and-Practice. Die Anwendungen führen den Lernenden in vielen kleinen Schritten durch den Lerninhalt. Nach jeder Übung wird sofort eine Rückmeldung und der Lernfortschritt angegeben.

3.2.1.2 Kognitivismus

Im Gegenteil zum Behaviorismus stellt der Kognitivismus (vgl. ARNOLD (2011) S. 102-103, ARNOLD (2005) S. 3-4, 7-9, REINMANN (2011) S. 3-4 und REY (2009d)) das Lernen als individuellen Informationsverarbeitungsprozess dar. Aspekte wie Wahrnehmung, Problemlösen oder Sprachverstehen rücken hier in den Mittelpunkt der Forschung.

Nach diesem Modell werden objektive Fakten und Regeln durch den Aufbau mentaler Modelle bzw. Schemata erlernt. Das Gelernte wird also mit bereits Gelerntem verknüpft, wodurch Strukturen weiterentwickelt und immer wieder verändert und erweitert werden. So bauen sich komplexe Wissenskonstrukte auf.

Kritisiert wird diese Theorie vor allem, da die Bedeutung sozialer, emotionaler und motivierender Pro-

zesse weiterhin vernachlässigt wird. Auch hier wird, wie im Behaviorismus nach berechenbaren Regeln zwischen den verschiedenen Prozessen gesucht.

Kognitivismus im E-Learning

Die Prinzipien des Kognitivismus werden im E-Learning besonders in „Intelligenten Tutoriellen Systemen“ angewendet.

Diese Programme passen sich an die Voraussetzungen der Lernenden an. Dies soll gewährleistet werden, indem Inhalte nach eigenem Interesse und Vorwissen, sowohl in der Reihenfolge als auch der Intensität, bearbeitet werden können. Zudem sollen Lerninhalte möglichst realitätsnah abgebildet sein. Auch gestufte Hilfesysteme werden hier eingeführt, die sich dem aktuellen Lernstand anpassen und individuelle Hinweise und Rückmeldung geben.

Wichtig ist bei dieser Lernart, dass ein guter Überblick über die angebotenen Lerninhalte vorhanden ist. Es sollte auch gekennzeichnet werden, welche Inhalte bearbeitet wurden und welche noch ausstehen.

3.2.1.3 Konstruktivismus

Der Konstruktivismus (vgl. ARNOLD (2011) S. 103-105, ARNOLD (2005) S. 4-6, 10-13, REINMANN (2011) S. 4 und REY (2009d)) ist die jüngste der hier vorgestellten Lerntheorien, sie versteht Lernende als selbstverantwortliche und aktive Personen im Hinblick auf ihren Wissenserwerb.

Anders als die vorangegangenen Modelle wird hier das Wissen nicht als objektiv vorhanden beschrieben, sondern als jeweilige Interpretation und Konstruktion des Lernenden. Dieses Wissen wird aufgrund eigener Erfahrungen oder Tätigkeiten erworben. Lernen kann somit von außen nur angeregt oder gehemmt, aber nicht direkt beeinflusst oder weitergegeben werden.

Der Ansatz wurde entwickelt, um dem „trägen Wissen“ (in realen Situationen oft nicht anwendbares Wissen) entgegenzuwirken. Das Modell steht bislang noch stark in der Kritik, da nur wenig empirische Belege für diesen Ansatz erbracht wurden. Laut Experten muss hier erst die eventuelle Überforderung der Lernenden (durch entdeckendes Lernen) mit den positiven Auswirkungen, die ein solches System haben kann (aber eben nicht immer hat), abgewogen werden.

Konstruktivismus im E-Learning

Im E-Learning wird dieser Ansatz durch das eigenständige Entdecken eines komplexen Ausgangsproblems umgesetzt. Die Lernumgebung dient hier nicht als Wissensträger, sondern stellt Werkzeuge zum eigenen aktiven Lösen von gestellten, praxisnahen Problemen zur Verfügung.

Auch das kooperative Lernen nimmt einen großen Stellenwert ein. Durch den Austausch mit anderen Lernenden sollen Problemstellungen möglichst vielseitig durchleuchtet werden. Auch die Fähigkeit zur Selbstregulation wird gefördert, da Lernziele selbstständig gesetzt und Selbstbewertungen erstellt werden sollen.

3.2.2 Lernziele und Lernziel-Taxonomien

Die zuvor betrachteten Lerntheorien bilden ein Fundament für die Entwicklung von Lernanwendungen. Für die praktische Umsetzung sind klare und ausformulierte Lernziele jedoch entscheidender.

Lernziele (vgl. KELLNER (2006), DÖRING (2010), ALTHBAUER (2011c), HESSE (2011), REINMANN (2011) S. 5-6)

Lernziele zu vereinbaren bildet die Grundlage für zielorientiertes Lernen. Diese Ziele sollen die Fähigkeiten bzw. das Wissen beschreiben die der Lernende besitzt, nachdem er erfolgreich eine Lerneinheit bearbeitet hat.

Sie haben somit einen entscheidenden Einfluss auf die Auswahl des Lernmaterials, -inhaltes und den verwendeten Methoden. Lernziele helfen die eigenen Absichten darzulegen und zu überdenken; sie sollten vor Beginn der eigentlichen Arbeit feststehen.

Damit Lernziele sinnvoll formuliert werden können, muss zunächst ein Überblick über deren Funktionen und Eigenschaften gegeben werden. Die Funktionen der Lernziele umfassen die Abgrenzung des Inhalts, Angaben darüber wie unterschiedlich vermittelte Inhalte ausgearbeitet werden können, sowie Kriterien für die Selbst- bzw. Fremdbeurteilung. Lernziele sollen dem Lernenden auch verdeutlichen welche Inhalte zentral sind, und ihm so bei der Planung des Lernprozesses helfen.

Lernziel-Taxonomien (vgl. DÖRING (2010), SCHMIDT (2011) S. 9-10 und REINMANN (2011) S. 6-7)

Lernziel-Taxonomien sind Klassifikationsschemata, die eine Analyse und Einordnung von Lernzielen unterstützen. Sie können die Planung eines Lernangebotes vereinfachen und überschaubarer machen indem sie klar spezifizieren, was erreicht werden soll.

Eine klassische Taxonomie ist die nach Benjamin Bloom. Diese teilt Lernziele in drei Gruppen ein, jede sollte in der Formulierung von umfassenden Lernzielen berücksichtigt werden:

- **Kognitive Lernziele** beschreiben das Wissen (Fakten und Regeln), das erlangt werden soll.
- **Affektive Lernziele** beinhalten Interessen, Einstellungen und Werte.
- **Psychomotorische Lernziele** beschreiben Verhaltensweisen, z.B. Bewegungsabläufe oder manuelle Fähigkeiten.

Untergliederung der kognitiven Lernziele (vgl. DÖRING (2010), KELLNER (2006) und BLOMERT (NN))

Unter den Lernziel-Taxonomien gilt die kognitive nach Benjamin Bloom als die bedeutendste. Hier lassen sich weitere Unterklassen herausbilden, so werden die Lernziele hier von einfachen (Kenntnisse) zu komplexen Verhaltensweisen (Evaluation) aufeinander aufgebaut.

1. Wissen/ Kenntnisse

Die einfachste Verhaltensweise stellt das Erinnern an bekannte Informationen dar, Fakten können nur wiedergegeben werden.

2. Verstehen/ Erkenntnisse

Die nächst höhere Stufe umfasst das Verarbeiten neuer Informationen. Hierzu gehört z.B. das Zusammenfassen und Interpretieren neuer Informationen.

3. Anwenden

Das Anwenden beinhaltet sowohl das Verwenden von Regeln zur Lösung von Problemen, als auch das Abstrahieren der erhaltenen Informationen auf andere Sachverhalte.

4. Analyse

Die Analyse-Verhaltensweisen umfassen die Zerlegung von Informationen in ihre Bestandteile. Hierzu soll der Lernende wichtige Informationen, deren Zusammenhänge und die Struktur zwischen den einzelnen Elementen benennen können.

5. Synthese/ Einsichten

In dieser Stufe werden Teilaspekte oder Elemente zusammengefügt und kombiniert. Dieses „neue Ganze“ umschließt dabei neue und bereits gelernte Informationen.

6. Evaluation

Die komplexeste Verhaltensweise stellt die Beurteilung der erhaltenen Informationen dar. Hier soll die Qualität und Quantität der Informationen kritisch betrachtet werden.

3.2.3 Lernaufgaben und -szenarien

Auf Grundlage der gewählten lerntheoretischen Grundlage (s. Kapitel 3.2.1 Lernen und Lerntheorien) sowie der entwickelten Lernziele (s. Kapitel 3.2.2 Lernziele und Lernziel-Taxonomien) können die konkreten Lernaufgaben entwickelt werden.

Hierbei können bestimmte Fragestellungen helfen neue Ideen zu entwickeln, oder vorhandene auf ihre Gültigkeit für die gewählten Lernziele zu überprüfen:

Fragen zu den Eigenschaften von Aufgaben (nach ARNOLD (2011) S. 112):

- Welche Kompetenzen werden durch die Bearbeitung der Aufgabe erworben? (Beurteilungswissen, Handlungsinteressen, Fach-, Bewertungs- oder Selbstkompetenz)
- In welcher Sozialform werden die Aufgaben bearbeitet? (Einzel- oder Gruppenarbeit; Gruppenarbeit mit gleichen oder arbeitsteilige Aufgaben)
- Beurteilung der Aufgaben bzw. Teilaufgaben aufgrund folgender Merkmale:
 - Einfachheit und Komplexität (einfache, schnell auswertbare Aufgabenstellungen und komplexe, realitätsnahe Aufgabenstellungen)
 - Position und Funktion (in Gesamtkontext des Lernangebots; von der Überprüfung des Vorwissens bis hin zur abschließenden Lernkontrolle, Reflexion oder Anwendung)

3 E-LEARNING (DIDAKTIK IM E-LEARNING)

Darüber hinaus muss festgestellt werden, in wiefern die Aufgaben geeignet sind den Lernenden beim Kompetenzerwerb zu unterstützen. Hierzu gibt es Rahmenbedingungen zur Auswahl der Lernszenarien (nach ARNOLD (2011) S. 119, vgl. REINMANN (2011) S. 8) die mit den Möglichkeiten und Lernzielen abgestimmt werden müssen:

- Quantitative Kategorien
 - Grad der Virtualität (angereicherte Präsenzveranstaltungen, Blended Learning bis hin zu virtuellen Veranstaltungen)
 - Gruppengröße (Einzelperson, Kleingruppen, Großgruppen)

- Technische Kategorien
 - Grad der Synchronisation (asynchron und/oder synchron)
 - Grad der Medialität (gering/elektr. Kommunikation, gemischt/Selbstlernprogramm, hoch/interaktive Übung)
 - Technische Möglichkeiten (ist es umsetzbar?)

- Didaktische Kategorien
 - Anteil Inhalt und Kommunikation ((nur) Inhalte und/oder (nur) Kommunikation)
 - Grad der Aktivität (rezeptives Lernen bzw. Lesen, Mischformen, aktive Lernform bzw. Produktion)

3.3 User-Interface-Design

Das User-Interface („Benutzeroberfläche“) (vgl. STAPELKAMP (2010) S. 151, THESMANN (2010) S. 7 und ARNOLD (2011) S. 135) ist die Kommunikationsmöglichkeit zwischen Mensch und Maschine. Das User-Interface-Design ist eine Schnittstelle, die menschliche Anforderungen und maschinelle Gegebenheiten bestmöglich in Verbindung bringt.

Die Gestaltung der Benutzeroberfläche muss dementsprechend, hinsichtlich der zu lösenden Aufgabe, den Bedürfnissen und Erfahrungen des Nutzers sowie der technischen Funktionalität z.B. des Computers optimiert werden. Eine ansprechende Gestaltung sollte ebenfalls stets gewährleistet sein. Die benutzer- und aufgabengerechte Gestaltung steht dabei immer vor der Ästhetik der Anwendung.

In diesem Kapitel sollen zunächst die Grundlagen für das User-Interface-Design ermittelt werden, indem die verschiedenen Richtlinien und Normen zur Gestaltung eines Interfaces vorgestellt werden. Im abschließenden Kapitel werden diese Punkte zu einer Richtlinie für „gute Gestaltung zur intuitiven Interaktion“ zusammengefasst.

3.3.1 Normen und Richtlinien

Es existieren zahlreiche Normen, Richtlinien und Empfehlungen für die Erstellung eines User-Interfaces. Da im Rahmen dieser Arbeit nicht alle Aspekte beleuchtet werden können, beschränke ich mich auf anerkannte und fundierte Empfehlungen.

3.3.1.1 ISO-Normen

Unter den ISO-Normen befinden sich viele Richtlinien zur Entwicklung eines gelungenen User-Interfaces. Die bedeutendste Norm für das User-Interface-Design ist die DIN EN ISO 9241, sie stellt Qualitätsstandards für die Ergonomie der Mensch-System-Interaktion auf. Innerhalb dieses Standards sind die Teile 8 und 10 bis 17 die wichtigsten für die Gestaltung der Interaktion (vgl. ISO 9241-10 (1995) S. 3, THESMANN (2010) S. 152-153 und STAPELKAMP (2010) S. 328):

- Teil 8: Anforderungen an Farbdarstellung
- Teil 10: Grundsätze der Dialoggestaltung
- Teil 11: Anforderungen an die Gebrauchstauglichkeit
- Teil 12: Informationsdarstellung
- Teil 13: Benutzerführung
- Teil 14: Dialogführung mittels Menüs
- Teil 15: Dialogführung mittels Kommandosprachen
- Teil 16: Dialogführung mittels direkter Manipulation
- Teil 17: Dialogführung mittels Bildschirmformularen

Unter den genannten Teilen der DIN EN ISO 9241 ist Teil 10 (Grundsätze der Dialoggestaltung) für den Entwurf der Interaktion nach zahlreichen Quellen am entscheidendsten. Die „Grundsätze der Dialoggestaltung“ beinhaltet sieben Empfehlungen, die es ermöglichen mit einem System einfach und intuitiv zu interagieren (vgl. ISO 9241-10 (1995) und THESMANN (2010) S. 233-234):

Aufgabenangemessenheit - „Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“ (ISO 9241-10 (1995) S. 5) Diese Kriterien sind erfüllt, wenn die Aufgabe mit einer geeigneten Funktionalität erledigt werden kann und nur relevante Information und Interaktionsmöglichkeiten gegeben werden.

Selbstbeschreibungsfähigkeit - „Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird.“ (ISO 9241-10 (1995) S. 6)

Steuerbarkeit - „Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.“ (ISO 9241-10 (1995) S. 8)

Erwartungskonformität - „Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z.B. den Kenntnissen aus dem Arbeitsgebiet, der Ausbildung und der Erfahrung des Benutzers sowie den allgemein anerkannten Konventionen.“ (ISO 9241-10 (1995) S. 9)

Fehlertoleranz - „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann.“ (ISO 9241-10 (1995) S. 10)

Individualisierbarkeit - „Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben des Benutzers und Benutzerfähigkeiten zulässt.“ (ISO 9241-10 (1995) S. 11) Hierzu gehört z.B. die Spracheinstellung, oder die Individualisierung bezüglich des Wahrnehmungs-, Reaktions- und Steuerungsvermögen eines Nutzers.

Lernförderlichkeit - „Ein Dialog ist lernförderlich, wenn er dem Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.“ (ISO 9241-10 (1995) S. 12) Dies wird meist durch die Selbstbeschreibungsfähigkeit, Erwartungskonformität und Fehlertoleranz der Anwendung erreicht.

Neben den Richtlinien der „ISO 9241“ gibt es noch weitere, für das User-Interface-Design relevante ISO-Normen. Die „DIN EN ISO 13407“ (Benutzerorientierte Gestaltung interaktiver Systeme) stellt z.B. Empfehlungen bereit, wie Nutzer der zu entwickelnden Anwendung optimal in den Entwicklungsprozess einbezogen werden. (vgl. ISO 13407 (1999))

Eine weitere User-Interface-Design relevante Norm ist die „DIN EN ISO 14915“ (Software-Ergonomie für Multimedia-Benutzerschnittstellen). Sie zeigt Gestaltungsgrundsätze sowie Möglichkeiten multimedialer Navigation auf und bietet eine Richtlinie für die Auswahl und Kombination von Medien. (vgl. STAPELKAMP (2010) S. 331)

3.3.1.2 Barrierefreiheit und leichte Sprache

Da die Barrierefreie Informationstechnikverordnung (BITV) (vgl. ARNOLD (2011) S. 162, THESMANN (2010) S. 37 – 39 und REICH (2011) S. 3) aus dem Gesetz zur Gleichstellung behinderter Menschen hervorgeht, wird diese Richtlinie oft vernachlässigt. Man sollte jedoch beachten, dass Websites auch ohne körperliche Beeinträchtigung Barrieren aufbauen können, sei es durch veraltete Technik, ein mobiles Endgerät, einen hohen Lärmpegel, besondere Lichtverhältnisse oder durch die Notwendigkeit die Anwendung ausschließlich per Tastatur zu steuern.

Es geht hierbei also nicht nur um die ca. 10 % der Menschen in der EU mit körperlichen Beeinträchtigung, sondern darum möglichst vielen Benutzergruppen eine Website oder Anwendung zugänglich zu machen.

Prinzipien und Richtlinien

Die folgenden Richtlinien stellen eine gute Grundlage für das User-Interface-Design dar und sollten selbstverständlich für jede gute Website sein. Der stärkste Einwand gegen diese Prinzipien sind die meist eingeschränkten Gestaltungsmöglichkeiten. Man sollte hier bei jedoch beachten, dass die Hauptaufgabe des User-Interface-Design die Zugänglichkeit der Anwendung und nicht deren Ästhetik ist. (vgl. THESMANN (2010) S. 39)

Auch wenn im VIL viele dieser Richtlinien aufgrund der hohen Interaktivität nicht umgesetzt werden können, sollten sie doch im Hinterkopf behalten und bestmöglich umgesetzt werden. Die Kriterien nach W3C bzw. WCAG (vgl. ARNOLD (2011) S.162, W3C (2002), WCAG (2008) und REICH (2011) S. 7) beschreiben die Anforderungen an Barrierefreiheit in Internetauftritten in vier grundlegenden Punkten:

- **Wahrnehmbarkeit:** Alle Informationen sollen so dargestellt werden, dass jeder Nutzer darauf zugreifen kann. So sollten Textalternativen für Audio- und Bildinformationen vorhanden sein, um sie allen Nutzern zugänglich zu machen (z.B. Umwandlung in Braille, Großschrift, Untertitel für Videos etc.). Auch auf einen genügend großen Kontrast zwischen Elementen und auf skalierbare Schriften sollte geachtet werden.
- **Bedienbarkeit:** Die Interaktionselemente sollen von jedem Nutzer bedienbar sein. Hier sollte auf die Möglichkeit einer Bedienung ausschließlich per Tastatur Wert gelegt und eine Unterstützung des Nutzers bei Navigation und Interaktion gegeben werden. Auch auf flackernde, blinkende oder sich schnell wechselnde Inhalte soll verzichtet werden, da sie ablenken und zu Anfällen führen können.
- **Verständlichkeit:** Die Inhalte und Struktur der Website sollte möglichst einfach und schnell erfassbar sein (Verwenden von Zwischenüberschriften, Aufzählungen etc.). Hierzu zählen auch Texte einfach darzubieten und visuelles Rauschen zu vermeiden (zu viele Farben, besondere Schriftarten etc.). Auch sollte auf die Erwartungskonformität und Fehlervermeidung (s. Kapitel 3.3.1.1 ISO-Normen) geachtet werden.
- **Robustheit:** Die Anwendung soll so gestaltet und entwickelt sein, dass sie mit allen gängigen Eingabe- und Ausgabetechniken ausführbar ist. Hierzu sollte eine Optimierung der Kompatibilität durch standardkonforme Entwicklung realisiert werden.

Leichte Sprache

Ein Aspekt der Barrierefreiheit ist die Verwendung von leichter Sprache zur besseren Verständlichkeit. Diese soll so erstellt werden, dass es jedem Menschen möglich ist den Inhalt zu verstehen. Da im Rahmen der VIL auch Informatik-Neulinge angesprochen werden, sollen hier kurz die wichtigsten Aspekte nach FREYHOFF (1998) aufgezeigt werden:

- Sätze sollen möglichst kurz und prägnant gehalten werden und nur eine Aussage beinhalten. Sobald mehr als 15 Wörter in einem Satz sind sollte dieser in mehrere aufgeteilt werden.
- Auch gilt es geläufige Begriffe zu verwenden. Der Konjunktiv (Möglichkeitsform), abstrakte Begriffe, Fremdwörter, Fachwörter oder lange Zusammensetzungen sind zu vermeiden oder gegebenenfalls anschaulich zu erklären.
- Abkürzungen werden beim ersten Vorkommen durch die ausgeschriebene Form erklärt.
- Trotz der leichten Sprache soll darauf geachtet werden, dass keine Kindersprache verwendet wird.

3.3.1.3 Gestaltgesetze

Um eine gestalterische Richtlinie zu erhalten hat sich die Anwendung der Gestaltgesetze bewährt (vgl. THESMANN (2010) S. 185 und BÖHRINGER (2008) S. 40). Diese Gesetze basieren auf den Erkenntnissen der Gestaltungspsychologie und stellen die Grundlage der Wahrnehmung von Formen und deren Beziehung zueinander dar. Die aus der empirischen Forschung entstandenen Erkenntnisse geben Hinweise für die Strukturierung eines groben Layouts.

Durch Anwendung der Gestaltgesetze können Benutzeroberflächen ansprechend und zugänglich entworfen werden. Eine Abweichung von diesen Gesetzen sollte nur gezielt eingesetzt werden, um die Aufmerksamkeit auf einen bestimmten Punkt zu lenken. Da diese Gesetze bereits in zahlreichen Veröffentlichungen beschrieben wurden, soll hier nur ein kurzer Überblick, über die für das User-Interface wichtigen Aspekte gegeben werden:

Gesetz der Ähnlichkeit (vgl. THESMANN (2010) S. 185 -187 und BÖHRINGER (2008) S. 43)

Elemente, die ein oder mehrere Merkmale (z.B. Form oder Farbe) gemeinsam haben werden als eine Gruppierung wahrgenommen. Dieses Gesetz fordert vor allem die gestalterische Konsistenz innerhalb einer Anwendung.

Gesetz der Nähe (vgl. THESMANN (2010) S. 188 und BÖHRINGER (2008) S. 42)

Nahe beieinander liegende Elemente werden als zusammengehörig wahrgenommen. Bei Menüs, Absätzen und inhaltlich zusammengehörigen Elementen findet dieses Gesetz Anwendung. Durch die räumliche Trennung von verschiedenen Inhalten wird zudem eine bessere Übersicht geschaffen.

Gesetz der Geschlossenheit (vgl. THESMANN (2010) S. 189f und BÖHRINGER (2008) S. 44)

Ein geschlossenes Objekt wird als eine Einheit betrachtet. So können Elemente z.B. durch einen Rahmen gruppiert und damit Zusammenhänge oder Abgrenzungen sichtbar gemacht werden.

Gesetz der Erfahrung (vgl. THESMANN (2010) S. 193f und BÖHRINGER (2008) S. 45)

Durch gelernte Muster können auch stark abstrahierte Formen oder Zeichen wiedererkannt werden. Diese Eigenschaft der menschlichen Wahrnehmung lässt sich vor allem für das Icondesign oder die Verwendung von Metaphern anwenden.

Gesetz der Prägnanz (vgl. THESMANN (2010) S. 194f und BÖHRINGER (2008) S. 47)

Durch die Verwendung von Konturen, Kontrasten, Bewegung oder Farbe können Elemente hervorgehoben werden. Dieses Gesetz sollte Anwendung finden, wenn der Blick auf wichtige Elemente gelenkt werden soll.

3.3.2 Usability

„Eine Anwendung soll eine Erleichterung und Bereicherung zur bisherigen Vorgehensweise, aber keine zusätzliche Herausforderung darstellen.“ Dieser Leitsatz zeigt, dass bei der Usability-Optimierung (vgl. KRUG (2006) und NIELSEN (2002)) Nutzer, mit ihren individuellen Bedürfnissen und Ansprüchen an die Anwendung, im Mittelpunkt der Entwicklung stehen.

Viele der Normen und Richtlinien aus Kapitel 3.3.1 unterstützen auch die Usability einer Anwendung. Wenn diese Normen in der Entwicklung einer neuen Anwendung beachtet werden, ist bereits mit einer guten Usability zu rechnen.

Anregungen für eine individuelle Usability-Optimierung können durch einen Usability-Test am Prototyp oder der bereits umgesetzten Anwendung ermittelt werden. So können Expertenbefragungen (z.B. Experten-Walk-Through), Nutzerbefragungen (z.B. Fragebögen) oder Nutzerbeobachtungen (z.B. Thinking Aloud, Eye-Tracking) durchgeführt werden, um Defizite in der Anwendung zu ermitteln.

Durch Usability-Tests können Fehler frühzeitig entdeckt und behoben werden, auch neue Anregungen und Ideen für die Entwicklung können entstehen. Ein zielgruppenorientierter Test sichert zudem die exakte Ausrichtung an den Bedürfnissen der späteren Nutzer.

Usability-Bewertung

Zur Bewertung der Usability-Ergebnisse lassen sich die fünf Hauptaspekte nach Nielsen heranziehen (vgl. KLIMSA (2011) S. 332f):

1. Erlernbarkeit (Learnability) – Wie viel Zeit wird benötigt um mit der Anwendung umgehen zu können?
2. Erinnerbarkeit (Memorability) – Ist die Anwendung auch nach längerer Nichtverwendung gut bedienbar und auf Anhieb verständlich?
3. Fehlerrate (Errors) – In wieweit werden Fehler minimiert bzw. wie hilfreich sind die Rückmeldungen bei Fehlern?
4. Zufriedenheit (Subjective Satisfaction) – Wie ist die Grundeinstellung zu der Anwendung (Maß für die Ausgangsbedingung zum erfolgreichen anwenden)?
5. Effizienz (Efficiency of use) – Wie viel Zeit wird benötigt um die Aufgabenstellung korrekt durchzuführen (evtl. auch Anzahl der Klicks)?

Auch verschiedene ISO-Normen (s. Kapitel 3.3.1.1 ISO-Normen), z.B. die DIN EN ISO 9241 - Teil 10 (Grundsätze der Dialoggestaltung), lassen sich zur Beurteilung der Usability heranziehen.

Sobald ein oder mehrere Punkte als Problem erkannt wurden, gilt es die Ursache zu finden (meist durch den Usability-Test schnell ermittelbar) und Verbesserungen zu entwickeln. Dies gelingt besonders gut in ständiger Rücksprache mit der Zielgruppe, indem von ihnen geäußerte Wünsche in die Verbesserung einfließen und wiederum getestet werden.

3.3.3 „gute“ E-Learning-Gestaltung zur intuitiven Interaktion

Der Lernerfolg im VIL ist stark von der Qualität der Interaktion abhängig, da alle Lernanwendungen erst durch deren Interaktionsmöglichkeiten zum Verstehen der Lektion beitragen. Aus diesem Grund soll hier eine Richtlinie für eine gute E-Learning-Gestaltung zur intuitiven Interaktion zusammengestellt werden. Bestehend auf den recherchierten gestalterischen Grundlagen sowie den Definitionen und Beschreibungen von Interaktion und Intuition.

3.3.3.1 Interaktivität und Interaktion

Interaktivität ist ein Begriff aus der Technik. Er umfasst die Möglichkeiten und Eigenschaften eines Systems, dem Benutzer einen aktiven und individuellen Einfluss auf Inhalte oder deren Darbietung zu geben. Interaktion ist dagegen ein psychologischer Begriff und beschreibt einen Informationsaustausch, auf sprachlicher oder symbolischer Ebene. (vgl. HOLZINGER (NN) S. 2 und STAPELKAMP (2010) S. 102)

Die verschiedenen Interaktionsmöglichkeiten lassen sich in fünf Grade aufteilen. Um einen besseren Überblick über diese Möglichkeiten zu erhalten wurden die Ausprägungen folgendermaßen beschrieben (vgl. SCHULMEISTER (2005), REY (2009e) und STAPELKAMP (2010) S. 102ff):

keine Interaktion

- Als nicht-Interaktiv bezeichnet man passives Lesen, Anhören und Betrachten von Inhalten. Es besteht dabei keine Möglichkeit der Beeinflussung des Inhaltes oder der Darbietung.

Mensch-Computer-Interaktion

- Einfache Benutzerinteraktion: Die einfachste Stufe der Interaktivität umfasst den individuellen Zugriff auf bestimmte vorgegebene Informationen. Hierzu gehört z.B. das Steuern einer Animation (über Kontrolltasten: Play, Pause, Vor, Zurück etc.) oder die Navigationsmöglichkeiten einer klassischen Website.
- Fragen und Rückmeldungen: Hier bestimmt der Nutzer individuell die nächste Darstellung oder den Inhalt. Sei es das Feedback einer Aufgabenstellung, eine Website mit individuellen Inhalten (z.B. über PHP) oder ein nonlineares Computerspiel.
- Verändern und Erstellen: Die komplexeste Interaktionsmöglichkeit zwischen Mensch und Computer stellt die vom Benutzer selbstständig durchgeführte Veränderung oder Erstellung von neuen Inhalten dar.

computergestützte Mensch-Mensch-Kommunikation

- Diese Stufe der Interaktivität umfasst computergestütztes Feedback, Diskussion und Kommunikation zwischen Menschen. Zu dieser Form der Interaktion zählen z.B. Chats, Foren oder Second Live.

Um einen Überblick über die Möglichkeiten der verschiedenen Mensch-Computer-Interaktionen zu geben werden nun die direkte Interaktion, Menüs und Navigationsmöglichkeiten sowie die Wichtigkeit von Rückmeldungen in diesem Kontext kurz umrissen.

Direkte Interaktion

Interaktion mit einem System wird meist mit der direkten Interaktion gleichgesetzt. Sie entspricht am meisten der natürlichen, realen Form der Interaktivität, da sie reale Interaktion z.B. das Greifen von Objekten in die virtuelle Welt überträgt. (vgl. DORAU (2011) S. 110 und STAPELKAMP (2010) S. 66)

Zur direkten Interaktion (vgl. STAPELKAMP (2010) S. 66) zählt z.B.

- das „Drag and Drop“ (Greifen und Verschieben von Gegenständen entlang aller Achsen, sowie scrollen oder rotieren eines „ergriffenen“ Objekts.),
- der „Schiebe- oder Drehregler“ (Verschieben oder Drehen eines Objekts um z.B. die Lautstärke zu verändern oder zu einer bestimmten Filmsequenz zu springen.) oder
- die „Scrollbar bzw. -balken“ (Bewegen eines Ausschnitts in eine vorgegebene Richtung um weitere/ andere Inhalte sichtbar zu machen.)

Menüs und Navigation

Auch die verschiedenen Menüformen und Navigationsmöglichkeiten (vgl. STAPELKAMP (2010) S. 20, 310f) gehören zur Interaktion mit einem System. Es können vorgegebene Inhalte ausgewählt oder aufgerufen werden. Ein großer Nachteil ist hierbei, dass meist nicht klar ist was passiert nachdem eine Auswahl getroffen wurde. Es sollte also auf eine möglichst eindeutige Beschriftung geachtet werden. Um die Übersichtlichkeit der Navigation bzw. des Menüs zu gewährleisten ist es zudem sinnvoll sich auf höchstens sieben Navigations- bzw. Menüpunkte zu beschränken. Nur bis zu dieser Anzahl kann das Kurzzeitgedächtnis alle Informationen zusammen aufnehmen.

Ein Menü (vgl. STAPELKAMP (2010)S. 20-37) kann zur Übersicht oder Navigation verwendet werden. Es kann aus einer einfachen Auflistung von Begriffen bestehen, als Register bzw. Reiter (angelehnt an Bibliotheksregister) oder als Pull- bzw. Drop-Down-Menü angelegt sein. Weniger gängige Menüs sind z.B. Menüs in der Erzählform (besonders bei Computerspielen, situationsabhängige Menüs) oder Pie Menüs (kreisförmige Anordnung der Auswahlpunkte, für sehr komplexe Software oder Hardwareinterfaces).

Weitere Navigationsformen sind Textlinks (Verweise/ Hyperlinks), der Navigationspfad (Brotkrumen-Navigation) oder die Suchfunktion einer Anwendung. Sitemaps (Überblick über alle Inhalte) oder Image-maps (Informationsvisualisierung) werden meist verwendet um komplexere Inhalte besser sichtbar zu machen. (vgl. STAPELKAMP (2010)S. 68-82)

Rückmeldungen

Um mit einem System interagieren zu können sind, neben der eigentlichen Interaktionsmöglichkeit, auch verschiedene Rückmeldungen auf Benutzereingaben nötig.

In der Interaktionsgestaltung sollte immer darauf geachtet werden Benutzerfehler zu vermeiden (s. Kapitel 3.3.1.1 ISO-Normen / Fehlertoleranz). Wenn doch ein Fehler begangen wurde muss die Anwendung, mit einer hilfreichen Reaktion zur Behebung dieses Fehlers, antworten. Hier gilt es vor allem den Nutzer nicht auf den Fehler hinzuweisen, sondern Hilfestellungen anzubieten um erneute Missverständnisse zu vermeiden. (vgl. DORAU (2011) S. 21)

Ein häufig auftretender Fehler entsteht bei nicht zur Verfügung stehenden, aber sichtbaren Interaktionselementen. Hier sollte eine aufschlussreiche Rückmeldung über den Grund der Inaktivität gegeben werden, wenn dieses Element vom Nutzer trotzdem ausgewählt wird. (vgl. DORAU (2011) S. 251)

3.3.3.2 Kriterien zur Gestaltung intuitiver Interaktion

Die Entwicklung von Systemen mit intuitiver Interaktion hat das Potenzial die Effizienz, Zufriedenheit und Kreativität der Nutzer zu steigern und sollte somit Ziel jeder Anwendung sein. (vgl. FRAUENHOFER IPK (2012)) Diese Effekte werden erzielt, da der Nutzer sich dem Inhalt und der zu lösenden Aufgabe optimal widmen kann, ohne von Interaktionsproblemen (z.B. Gedanken ob man richtig mit der Anwendung umgeht oder getätigte Klicks, ohne dass das genaue Resultat dieses Klicks bekannt ist) abgelenkt zu werden. (vgl. DORAU (2011) S. 24 und TSE (2012a))

Nachdem ein Einblick in die verschiedenen Interaktionsmöglichkeiten gegeben wurde, soll nun der Begriff der intuitiven Interaktion genauer betrachtet werden. Auf dieser Grundlage werden anschließend die Kriterien zur Erstellung intuitiver Interaktion abgeleitet und das kreieren von neuen Interaktionskonzepten erklärt.

Eine intuitive Handlung ist eine Reaktion, auf Basis eines gefühlten Wissens. Diese Reaktionen entstehen demnach nicht aus einem Prozess des Überlegens und anschließenden Handelns, sondern spontan und meist unterbewusst. Sie beruht auf dem Wissen, dass in vergangenen Situationen erlernt wurde und nun auf eine ähnliche Situation wieder angewendet wird. (vgl. ALTBAUER (2011d))

Intuition ist also die Fähigkeit alltägliche Handlungen in einer ähnlichen Situation anzuwenden. Eine Anwendung ist demnach intuitiv, wenn sie durch die Anwendung von Vorwissen leicht verständlich und ohne Erklärungen und Einschränkungen verwendbar ist. (vgl. ALTBAUER (2011d) und FRAUENHOFER IPK (2012))

Kriterien

Auf Grund der in diesem Kapitel (3.3 User-Interface-Design) gewonnenen Einblicke in Gestaltung, Interaktion und Intuition sowie bekannten Ergebnissen aus Usability-Tests (eigene Praxiserfahrung und z.B. Nielsen (2002)) lassen sich folgende Grundsätze aufstellen:

1. Grundsatz – etablierte Lösungen weiterverwenden

Intuitiv ist etwas, das bereits gelernt wurde, sei es in anderen Anwendungen oder alltäglichen Handlungen. Somit sollte, wenn es eine etablierte Lösung gibt, diese bevorzugt eingesetzt werden. (Etablierte Interaktionsmöglichkeiten wurden bereits vorgestellt, s. Kapitel 3.3.3.1 Interaktivität und Interaktion)

2. Grundsatz – Überlegenheit von Bildern

Bilder und Symbole sind leichter verständlich und meist auch besser zu merken als textliche Beschreibungen. Demnach ist eine Interaktionsmöglichkeit mittels Symbol intuitiver anwendbar, als eine rein textliche. Diese Symbole müssen jedoch eine natürliche und offensichtliche Beziehung zu den Dingen bzw. Abläufen haben. Die Darstellung sollte eindeutig, klar und missverständnisfrei sein. (vgl. TSE (2012b))

3. Grundsatz - Gestaltungsgesetze anwenden

Die systemgerechte Modifizierung von Interaktionsmöglichkeiten sollte auf Grundlage der Gestaltungsgesetze stattfinden um eine optimale, gestaltungspsychologisch korrekte und damit verständliche Darstellungen zu erhalten. (s. Kapitel 3.3.1.3 Gestaltungsgesetze)

Kreieren von „neuen“ Interaktionskonzepten

Da bereits erlernte Möglichkeiten zur Interaktion intuitiv anwendbar sind sollte sich, bei der Erstellung einer neuen Anwendung, an diesen Möglichkeiten orientiert werden. Vollkommen neue Konzepte zu etablieren stellt sich meist als schwierig dar. Hier sollte besonders auf eine gelungene und mindestens der Zielgruppe bekannte Metapher einer alltäglichen Handlung zurückgegriffen werden.

Es gilt diese Metapher mit Hilfe der „Fragen und Empfehlungen zur Entscheidungsfindung und Gestaltung von Metaphern“ STAPELKAMP (2010) zu modifizieren und möglichst realitätsnah umzusetzen. In jedem Fall sollte nach dem Entwurf eines Interaktionskonzeptes ein Usability-Test durchgeführt werden, um die Wirksamkeit der Metapher (was mit dem entworfenen Symbol assoziiert wird) zu testen.

Fragen und Empfehlungen zur Entscheidungsfindung und Gestaltung von Metaphern (Auszug aus STAPELKAMP (2010) S. 63):

- **„[Die] Komplexität der Metapher an die Komplexität der Produktion angleichen.** ([...] Imitiert die Metapher die äußeren Realität? [...] Wie viel kann aus der Welt, aus der die Metapher entlehnt wurde, übernommen werden? [...] Kann die Metapher Sachverhalte veranschaulichen und Orientierung bieten?)“
- **„[Die] Struktur der Metapher an die Struktur der Produktion angleichen.** (Inwieweit ist die Struktur des Inhalts bzw. der Dienstleistung auf eine Metapher anwendbar oder die Metapher auf die Struktur? Welche Aspekte sind wichtig [bzw. relevant oder können verwirren] ? Welche Funktionen sind erklärungsbedürftig? Ist die Metapher einfach zu interpretieren, ohne trivial zu wirken?)“
- **„[Eine] gut darstellbare Metapher verwenden.** (Ist die Metapher angemessen? [...] Ist die Metapher gut darzustellen [(wie aufwändig ist es diese darzustellen)] ? Ist die Metapher implementierbar in die Gesamtgestaltung [und das Funktionsprinzip, dem Screen- und Informationdesign] ?)“
- **„[Eine] Metapher verwenden, die Anwender kennen.** (Inwieweit wird die Metapher von den Anwendern verstanden? Setzen die Metaphern beim Anwender Erfahrung voraus? Aus welchem Fachgebiet kommt die Metapher?)“
- **„Offene Metaphern benutzen.** (Kann die Metapher eine Eigendynamik entwickeln, die die Benutzung des Systems in positiver Weise verändert? Erschweren die Metaphern die Einführung von grundlegend neuen Ideen? Schränkt die Metapher die medialen Ausdrucksmöglichkeiten ein?)“

3.4 Vorgehen bei der Entwicklung

In der Entwicklung von E-Learning-Anwendungen spielen verschiedene Aspekte eine Rolle. Die wichtigsten wurden in den vorhergehenden Kapiteln beschrieben. Es sind die Grundlagen der Didaktik, um den Inhalt lerngerecht aufzubereiten (s. Kapitel 3.2 Didaktik im E-Learning) sowie die Grundlagen der Gestaltung, um die Anwendung ansprechend und nutzbar zu entwerfen (s. Kapitel 3.3 User-Interface-Design).

Auch eine Vorgehensweise zum Erstellen der Anwendung ist wichtig. In vielen Quellen findet man hierzu Vorgehensmodelle und Richtlinien die sich im grundlegenden Aufbau sehr ähneln. Für diese Arbeit habe ich verschiedene Modelle kombiniert um eine optimale Vorgehensweise zu erhalten (Verwendete Modelle und Vorgehensweisen: Gerhard Zimmer: „Aufgabenorientierte Entwicklung virtueller Lernmodule“ in ARNOLD (2011), Michael Kerres / Nadine Ojstersek / Jörg Stratmann: „Didaktische Konzeption von Angeboten des Online-Lernens“ in KLIMSA (2011) und Peter Schisler: „Planung und Entwicklung von Online-Lernangeboten in der Praxis“ in KLIMSA (2011)).

3.4.1 Vorbereitungen

Es gilt zunächst eine umfassende Analyse der Anforderungen durchzuführen, z.B. den Zweck der Lerneinheit, die technische Rahmenbedingungen oder die Analyse der Zielgruppe. Da diese Arbeiten im VIL bereits durchgeführt wurden soll hier nicht näher darauf eingegangen werden. (vgl. KLIMSA (2011) S. 256-262 und 265-268)

3.4.2 Didaktische Konzeption

Um ein professionelles didaktisches Design zu garantieren habe ich mich für das Vorgehensmodell nach Zimmer entschieden, da viele der enthaltenen Ansichten auch im VIL umgesetzt wurden und das Modell einen geschlossenen und plausiblen Rahmen bildet. Zimmer skizziert hier einen kompletten Ablaufplan, der in vier Phasen unterteilt ist. Da nicht alle Elemente des Vorgehensmodells für diese Arbeit relevant sind, sollen hier nur die für das VIL relevanten und in der Arbeit verwendeten Aspekte kurz vorgestellt werden.

Phase I – Konzeptphase (nach Gerhard Zimmer z.B. in ARNOLD (2011) S. 128-129, vgl. KLIMSA (2011) S. 267-268)

In dieser Phase werden die inhaltlichen Rahmenbedingungen festgelegt. Hier gilt es die Lernziele (s. Kapitel 3.2.2 Lernziele und Lernziel-Taxonomien) zu ermitteln und das erwartete Vorwissen der zukünftigen Lernenden festzuhalten (z.B. das Wissen aus vorhergehenden Lerneinheiten).

Phase II – Didaktische Struktur (nach Gerhard Zimmer z.B. in ARNOLD (2011) S. 130-131)

Zur Planung der didaktischen Struktur wird festgehalten welche Inhalte zur Verfügung gestellt werden, oder vom Lernenden selbst erarbeitet werden sollen. Die Entscheidung hängt hierbei vor allem von den

Kompetenzen und Lernaufgaben ab.

Zudem soll eine grobe Übersicht über die Lerneinheiten erstellt werden, eine sowohl zeitliche als auch organisatorische Struktur des Moduls. Es wird festgehalten welche Inhalte in welchen Lerneinheiten vermittelt werden und welche Aufgaben und Arbeitsformen dafür geeignet sind. Auch der gesamte Ablauf des Moduls wird geplant, die Reihenfolge der einzelnen Einheiten, das Zusammenspiel aus evtl. Präsenzteil und virtueller Phase.

Phase III – Formale Struktur (nach Gerhard Zimmer z.B. in ARNOLD (2011) S. 113, 132-133)

Auf Grundlage der vorangegangenen Überlegungen wird nun das Feinkonzept erstellt. Hierzu werden konkrete geeignete Lernaufgaben entworfen (s. Kapitel 3.2.3 Lernaufgaben und -szenarien) und entsprechende Lernressourcen aufbereitet die zur Verfügung gestellt werden. Auch Art und Inhalt der Rückmeldungen muss geplant und konzipiert werden. Hier gilt es eine schnelle, individuelle und differenzierte Rückmeldung zu entwickeln.

Es ist zudem zu ermitteln, ob Inhalte multimedial aufbereitet werden sollen sowie die Gestaltung dieser Elemente unter Berücksichtigung der Gestaltungsregeln (s. Kapitel 3.3 User-Centred-Design).

3.4.3 Erstellung eines Multimedia-Drehbuchs (Teilaspekt der Phase IV nach Zimmer)

Ein ebenso wichtiger Punkt ist abschließend das Festhalten der gewonnen Erkenntnisse und der Planungen der eigentlichen Umsetzung. Hier wird oft die Erstellung eines Multimedia-Drehbuchs empfohlen (z.B. Gerhard Zimmer: „Aufgabenorientierte Entwicklung virtueller Lernmodule – Phase IV: Operationale Struktur“ in ARNOLD (2011) S. 133, 159-160, Alexander Westphal: „Drehbuchs schreiben für Online-Lernangebote“ in KLIMSA (2011) S. 198-204 und „Feinkonzept (Drehbuch)“ in THESMANN (2010) S. 228ff). In diesem Drehbuch wird das Konzept und jede Bildschirmseite mit allen Inhalten, Elementen und Regieanweisungen festgehalten:

Hierzu gehören die Inhalte

- Verschiedene Bildschirmansichten, Aufteilung und Platzierung der enthaltenen Elemente
- Reihenfolge der einzelnen Sequenzen

eine detaillierte Beschreibung der Inhalte und Frames

- Video, Animation, Simulation, Foto und Illustration (mit Skizzen)
- Ton, Sprache, Aufgabentexte und Lehrtexte (Typografie, Atmosphäre etc.)
- Verwendete Interaktionstypen (Multiple Choice, Drag & Drop etc.), Navigationselemente
- Zur Verfügung stehende Funktionen und zusätzliche Inhalte/Hilfstexte etc.

und die genaue Beschreibung der Navigation und Interaktion

- Strukturbaum aller Bildschirmansichten
- Aktive/inaktive Navigationselemente und wo sie hinführen
- Tooltips (Text, Position und Größe)
- Interaktionsmöglichkeiten und Rückmeldung auf die Benutzereingabe (zu erwartende Benutzereingaben, Text, Position und Größe der Rückmeldung, Anzahl der Versuche etc.)

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

4.1 Änderungsvorschläge

Wie bereits in der Aufgabenstellung erwähnt, gab es eine Überprüfung des VIL bezüglich seiner Usability und des zu erreichenden Lernerfolgs (s. FEHRENBACH (2012)). Aus dieser Analyse ist eine Liste mit konkreten Änderungsvorschlägen hervorgegangen, die nun innerhalb der neuen Listen-Lektion umgesetzt werden sollen.

4.1.1 Rahmen und Navigation

Änderungen (vgl. FEHRENBACH (2012) S. 73-77)

Bezüglich der Navigation ergaben sich die dringendsten Änderungen. Hier soll vor allem eine klare Übersicht über die Inhalte der Lektionen geschaffen werden. Die „Lernwege“ können weggelassen werden, da sie die Probanden eher verunsicherten. Verwendete Farben sollen auf Highlights reduziert werden.

Das MIA-Feedback soll effektiver und aussagekräftiger gestaltet werden. Motivierendes Feedback wurde als eher störend empfunden und kann somit weggelassen werden.

Idee & Konzeption

Aufgrund dieser Vorgaben wurde der Rahmen inklusive der Navigation angepasst. Die einzelnen Navigationsmöglichkeiten sind nun in einer übersichtlichen Einheit (modifiziertes Wäscheschild) zusammengefasst.

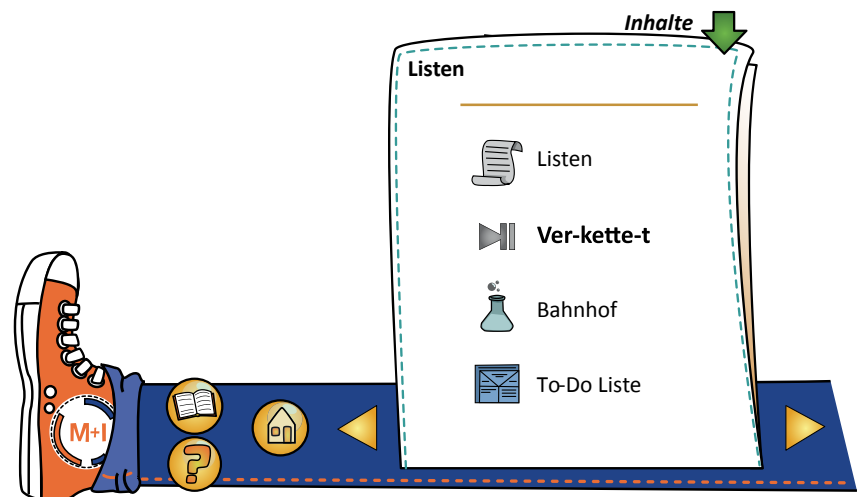


Abb. 14 modifizierte Navigation (ausgeklappt)
(eigener Entwurf auf Grundlage bestehender VIL-Elemente)

Die Inhalte dieser neuen Navigation sind ähnlich der früheren Lupen-Navigation aufgebaut: Unter dem Lektionsnamen finden sich die einzelnen Inhalte der Lektion (Instruktion, Animation, Experiment und Struktogrammentwicklung). Auch eine schrittweise-Navigation ist durch die Pfeile (weiter und zurück) gegeben; sie durchläuft alle Inhalte.

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

Der Home-Button (Button mit Haus; führt zurück in den Laborraum) wurde mit einer neuen Gestaltung separat neben der Lektions-Navigation platziert. Für die Animationsanwendung wurde ein neues Symbol erstellt (Play-/Pause-Button), welches auf die Möglichkeit der Interaktion mit einer Animation hindeutet.

Um die Farbvielfalt einzugrenzen sind die Rahmen der verschiedenen Inhaltsbereiche einfarbig gestaltet, die Navigationselemente wurden farblich reduziert.

Die detaillierte Layoutvorgabe zur Umgestaltung der Navigation findet sich im Anhang unter „Neues Navigationskonzept“ (S. 105).

4.1.2 Verschiedene Interaktionsmöglichkeiten

Änderungen (vgl. FEHRENBACH (2012) S. 73-77)

Auch bei der Interaktion mit den verschiedenen Anwendungen können einige Elemente geändert und erweitert werden.

So soll die Interaktion in einem Begleittext erklärt werden, um den Lernenden eine Möglichkeit zu geben, sich über deren Bedienung oder „Spielregeln“ zu informieren. Es ist möglich, durch die Änderung zu Gunsten der Usability, viele der festgestellten Probleme zu beheben. Eine zusätzliche Beschreibung gibt den Lernenden jedoch eine zusätzliche Sicherheit.

Bei der Animation und dem Experiment soll das Kontrollpanel näher an die eigentliche Interaktion platziert werden, um die Zugehörigkeit sichtbar zu machen.

Die Änderungen zur Struktogrammentwicklung umfassen kleinere Anpassungen. So soll das „bauen“ erleichtert werden, indem die Anfangselemente (Methodenaufruf und Ausgabe) mittig im Anwendungsfenster platziert werden, anstatt oben links. Die Eingabe im Struktogramm wird vereinfacht indem ein „Rückgängig“-Button eingeführt und der „Löschen“-Button sprechender benannt wird. Die „Funktionen“ werden direkt unter die „Variablen“ verschoben.

Zusätzlich wurde von den Probanden eine ausführliche Einzelschritterklärung bei vorgegebenen Struktogrammen gewünscht.

Idee & Konzeption

Zur Optimierung der Interaktion werden darum folgende Änderungen eingeführt: Die Interaktion wird über ein separates Fenster („Bedienung“) erklärt. Hier sollen auch evtl. „Spielregeln“ wie z.B. bei den „Türmen von Hanoi“ erklärt werden, falls Lernenden diese Probleme unbekannt sind. Die zusätzliche Beschreibung kann über einen Reiter im Feld der Aufgabenstellung („Aufgabe“) angezeigt werden. Durch die separate Darbietung können erfahrene Lernende die Beschreibung überspringen.

Das Kontrollpanel wird umpositioniert und direkt in das eigentliche Interaktionsfenster integriert. Dabei

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

bleiben die gestalterischen Elemente erhalten. Die zusätzlichen Erklärungen der Struktogramm-Befehle sind bereits als Tool-Tip im Struktogramm angedacht, um Lernende auf diese Möglichkeit hinzuweisen wird ein entsprechender Hinweis in die „Bedienung“ integriert.

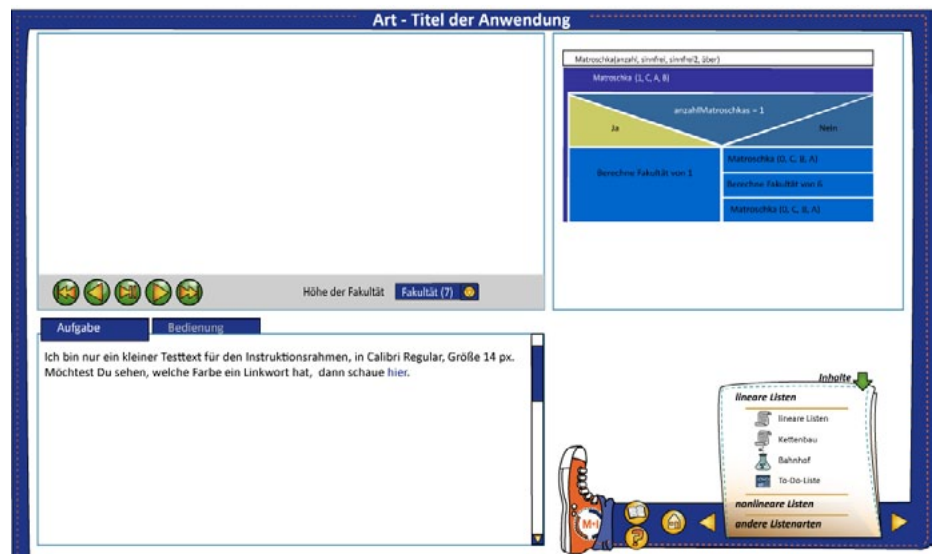


Abb. 15 modifizierte Seitenansicht für Animationen und Experimente (eigener Entwurf auf Grundlage bestehender VIL-Elemente)

Die Änderungen zur der Struktogrammentwicklung werden wie beschrieben umgesetzt. Zusätzlich soll eine weitere Änderung eingeführt werden, die auf Grundlage des Kapitels 3.3.3.2 „Kriterien zur Gestaltung intuitiver Interaktion“ erkannt wurde: Sobald ein Struktogrammblock ausgewählt wurde werden die Stellen, an die dieser Block platziert werden kann, optisch hervorgehoben. Hierdurch wird besser auf den Platzierungsort und die erwartete Interaktion hingewiesen.

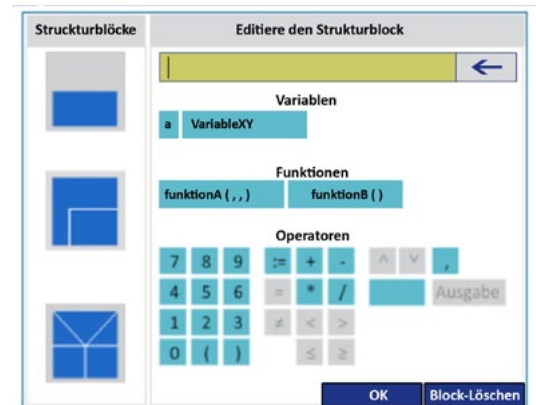


Abb. 16 modifizierte Tool-Box für Struktogrammentwicklung (eigener Entwurf auf Grundlage bestehender VIL-Elemente)

4.1.3 Texte und Textdesign

Änderungen die Texte oder das Textdesign (vgl. Fehrenbach (2012) S. 73-77) betreffen, werden nun kurz aufgeführt. Sie können jedoch erst in der Konzeption der eigentlichen Lektion umgesetzt werden (s. Kapitel 5.2 Inhalte der Lektion).

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

Die Einführungsseite gilt es übersichtlicher zu gestalten. Die optische Komplexität des Textes soll hier abgeschwächt werden. Bei den Tooltips soll darauf geachtet werden, dass wirklich nur zusätzliche Informationen dahinter verborgen sind. Auch wurde von einigen Probanden gewünscht die Textgröße zu erhöhen (in der Konzeption der Navigation wurde dieser Wunsch bereits integriert).

Die Erklärungen sollen sprachlich vereinfacht und gekürzt werden. Zudem fehlte vielen Probanden bei der Animation bzw. dem Experiment der Bezug zum eigentlichen Lernthema, auch dieser soll in der Aufgabenbeschreibung kurz vorgestellt werden.

Die Texte sollen generell verständlicher gestaltet werden. Hierzu gehören sinnvolle Absätze, Unterüberschriften und die Hervorhebung von wichtigen Elementen.

4.1.4 Prototypisches-Testen von Navigation und Interaktion

Um die vorgenommenen Änderungen zu testen wurde ein vergleichender Usability-Test (Thinking-Aloud) am Prototyp durchgeführt. Hierzu haben drei Probanden die Navigation getestet, indem sie mittels eines interaktiven PDFs die neuen Navigationsmöglichkeiten erprobten. Die Interaktion wurde überprüft indem die Probanden beschreiben sollten, wie sie mit den gegebenen Möglichkeiten interagieren würden.

Ablauf des Tests:

Vorgespräch/ Anmerkungen

- Test-Rahmen besprechen (Test im Rahmen meiner Thesis. Kurz die Zielgruppe des VIL beschreiben. Es kann vollkommen offen die eigene Meinung geäußert werden, da die Ideen zur Änderung im letzten Jahr von einer Projektgruppe entwickelt wurden.)
- Der Proband soll sich so verhalten als würde er/sie diese Lektion bearbeiten und dabei kurz erklären welchen Eindruck er/sie bekommt. (Ob irgendetwas verwirrend ist, die Darstellung zur Zielgruppe passt, etwas verbessert werden kann etc.)
- Bei Seiten mit Interaktion (Animation und Struktogrammentwicklung) kann nachgefragt werden wie die Interaktion nach Meinung des Probanden funktioniert. (Das Experiment kann nur anhand einer implementierten Lösung getestet werden, da die Interaktion zu komplex zum Erklären ist.)

Testaufgaben/ Fragestellung

- Animationsseite: Wie wählst du verschiedene Parameter (z.B. Anzahl der Schnecken) aus? Wie startest du die Animation?
- Struktogrammentwicklung: Wenn du dieses Struktogramm (vorgegebene Lösung) nachbauen solltest, mit was würdest du anfangen? Wenn Drag&Drop verstanden: Wie würdest du weiter vorgehen/ die Blöcke füllen? Was würdest du machen, wenn du nicht weißt wie du das Struktogramm nachbauen sollst (Testen der Möglichkeit sich die „Bedienung“ anzuzeigen)?

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

- Wenn zwischen allen Seiten nur mit einer Variante navigiert wurde, können hierzu weitere Fragen gestellt werden: Springe zurück auf die erste Seite. Findest du noch eine andere Möglichkeit zur Navigation?
- Zusätzlich: Wohin führt der Button mit dem Haus? („zurück in den Laborraum“)
- Zum Vergleich wurde allen Probanden, neben dem Prototyp, auch das aktuelle VIL (online-Version April 2012) gezeigt. Es wurden die vorgenommene Änderungen/Erweiterungen besprochen und beurteilt.

Ergebnisse und resultierende Änderungen:

Die Navigation war für alle drei Probanden auf Anhieb verständlich und intuitiv bedienbar. Die Interaktionsmöglichkeit der Animationsanwendung wurde schnell gefunden, deren Funktionsweise und Anwendung treffend beschrieben (da ähnlich einer Standard-Videointeraktion leicht zu verstehen).

Die Interaktion in der Struktogrammentwicklung wird wohl eine schwierige Interaktion bleiben, nach Aussage der Probanden sind hier aber sinnvolle Änderungen eingearbeitet worden. Weitere Änderungsmöglichkeiten wurden jedoch von keinem der Probanden angezeigt.

Durch eine zusätzliche Beschreibung der Interaktion („Bedienung“) fühlten sich alle Probanden sicher diese Aufgabe lösen zu können. Hervorgehoben wurde die Möglichkeit die Bedienungsempfehlung direkt verwenden zu können, da sie gleichzeitig zur Interaktion lesbar ist.

Im Vergleich zu dem gezeigten original VIL sind die Änderungen laut Aussage der Probanden alle positiv und sinnvoll. Sie tragen zur Verbesserung des Lernenergebnisses sowie der intuitiven Interaktion mit der Anwendung bei.

Als zusätzliche Ergänzung soll lediglich ein Hinweis auf die „Bedienung“ zu finden sein. Diese soll unter der Hilfe („?“) platziert werden, z.B. mit den Worten: „Wenn du Hilfe bei der Anwendung brachst helfen dir die Hinweise zur Bedienung im Fenster der Aufgabenbeschreibung“.

4.2 Umsetzung der neuen Navigation

Um das neue Navigationskonzept umzusetzen mussten einige vorhandene Elemente neu platziert, sowie die komplette Navigation neu erstellt und implementiert werden. Das separate Fenster für den Wechsel zwischen der Aufgabenstellung und der Beschreibung der Bedienung wurde in diesem Schritt ebenfalls implementiert, da es für jede Anwendung relevant ist.

Die Vorgehensweise bei der Umsetzung der Änderungen sollen nun erklärt werden, um weiteren Projekten am VIL einen guten Ausgangspunkt zu liefern:

4.2.1 Neuplatzierung der vorhandenen Elemente

Mit der Unterstützung des Entwicklers der aktuellen VIL-Komponenten (Khanh Do) konnte die neue Anordnung der vorhandenen Elemente schnell umgesetzt werden. Es war hierbei lediglich nötig herauszufinden wie die entsprechenden Interaktionselemente verschoben werden können.

Die Verschiebung der Elemente kann in der Flash-Datei „Structogram fla“ durchgeführt werden. Hier werden die meisten Interaktionselemente, z.B. die Kontrollbuttons oder die Tool-Box des Struktogrammentwurfs, verwaltet. Das Kontrollpanel konnte einfach auf der „Flash-Bühne“ umpositioniert werden, ebenso konnte hier die Farbe der Rahmen angepasst werden.

Um den grauen Balken im Hintergrund der neuen Interaktionsleiste (im eigentlichen Animations-/Visualisierungs-Rahmen) zu vergrößern, muss im Movie-Clip „fhoFormDiagram“ (ebenfalls in der Flash-Datei „Structogram fla“) der entsprechende Balken umpositioniert werden. Obwohl die dortige Ansicht einen anderen Eindruck vermittelt, wirkt sich diese Änderung auf alle Anwendungen die diesen Rahmen verwenden (viele Animationsanwendungen und alle Struktogrammentwicklungen) aus.



Abb. 17 Screen der neuen structogram fla (links) und der früheren (rechts)
(Quelle: Screenshot Flash-Entwicklungsansicht)

Einzig die Verschiebung des Drop-Down-Menüs und dessen Beschriftung sind nicht in dieser Datei zu finden. Diese müssen direkt für die Ausgabe der jeweiligen swf-Datei geändert werden (Ordner: build/virtlab/LEKTIONSNAME/fhoStruct/diagramFiles). In der Matroschka-Animation müsste z.B. für die Anpassung die Datei „matroschka05.dia“ in einem Editor geöffnet werden und die, hier türkis hervorgehobenen, Werte folgendermaßen verändert werden:

```
<param>  
    <v name="n" tip="Der Laufvariablen n wird das Argument aus dem Funktionsaufruf  
        zugewiesen" combo="1" min="1" max="6" default="5" x="350" y="-10" alias="Fakultät(n)"/>  
</param>
```

Die ursprünglichen Werte lagen hier bei $x="150"$ und $y="0"$. Die weiteren Angaben in dieser xml-Datei enthalten bei Animationen die Information wie das Struktogramm der Animation aufgebaut ist. Bei Struktogrammentwicklungen enthalten sie die Lösung dieser Entwicklung. Um die DropDown-Menü-Veränderung umzusetzen muss somit, in jeder neu erstellten Anwendung, dieser Wert individuell in der entsprechenden xml-Datei angepasst werden.

Für Anwendungen die nicht auf einem Struktogramm basieren, oder aus anderen Gründen nicht auf Grundlage der „Structogram.fla“ erstellt werden konnten, existiert eine weitere Vorlage („templateApplication.fla“). Auch hier wurden die Elemente angepasst. Im Gegenteil zur „Structogram.fla“ konnten jedoch einfach alle Elemente auf der „Flash-Bühne“ verschoben werden, auch das DropDown-Menü ist hier direkt in die Vorlage integriert.

4.2.2 Implementierung des Fensters für Aufgabe und Bedienung

Diese hier beschriebenen Änderungen führte ich selbstständig durch, als Hilfsmittel verwendete ich ein Flash-Handbuch (WESCHKALNIES (2010)). Durch die Überarbeitung der bereits implementierten Anwendungen des VIL konnte ich einige der dort verwendeten Ansätze, leicht abgeändert, weiter verwenden.

Für die Implementierung des Aufgaben/Bedienungs-Fensters konnte wieder die Vorlage der Flash-Datei „templateApplication.fla“ verwendet werden. Hier wurden die Buttons als MovieClip erstellt und deren Funktion im vorhandenen Code ergänzt. Auch die Möglichkeit zwei verschiedene Texte einzustellen wurde hier ergänzt, indem zusätzliche MovieClip-Textcontainer (übernommen vom bereits vorhandenen MovieClip „Description“) erstellt wurden. Dabei ist „Description2“ für den Aufgabentext und „Description3“ für den Text zur Bedienung vorgesehen.

Um die Veränderung zu veranschaulichen und einen Einblick in die jeweiligen Flash-Dateien zu gewährleisten, ist untenstehend (Abb. 18) eine Vergleichsübersicht der Screens der templateApplication.fla. Die vorhandenen Buttons sowie der „SPane“ (zum hereinladen der Texte) wurden verschoben und die beiden neuen Buttons Aufgabe (Instanzname: btn_aufgabe) und Bedienung (Instanzname: btn_bedienung) ergänzt.

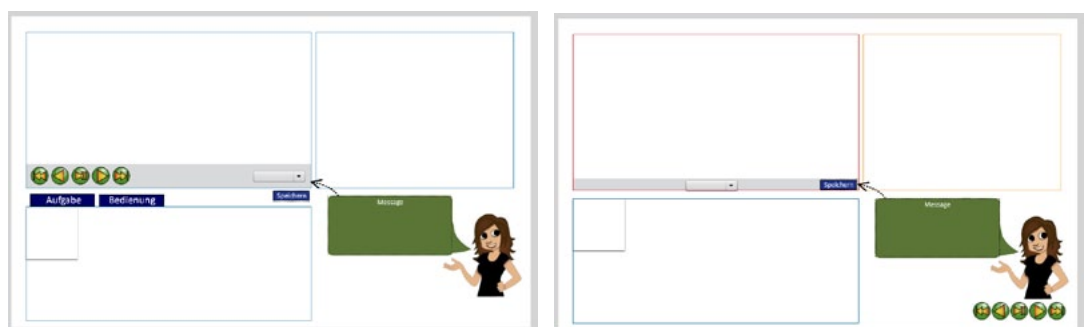


Abb. 18 Screen der neuen templateApplication.fla (links) und der früheren (rechts)
(Quelle: Screenshot Flash-Entwicklungsansicht)

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

Im zugehörigen Actionscript-Code mussten folgende Anweisungen erweitert bzw. verändert werden:

Auszug aus früherem Code (templateApplication.as - Entwickler: Khanh Do):

```
Zeile 11      public var description : MovieClip;
Zeile 32      SPane.setSize(550, 240);
```

Auszug aus neuem Code (templateApplication_new.as - eigene Ergänzungen):

```
                // Anlegen der MovieClip-Container für die verschiedenen Texte
Zeile 11      public var description, description2, description3 : MovieClip;
                // Anpassen der Länge des Scroll-Balkens für das nun etwas kleinere Text-Fenster
Zeile 32      SPane.setSize(550, 220);
```

Alle weiteren Ergänzungen sind neu hinzugekommen (s. Anhang, Seite 111 - „Auszug aus neuem Code (templateApplication_new.as - eigene Ergänzungen)“).

Für Anwendungen die auf Grundlage der Structogram.flas erstellt werden, musste wiederum ein anderes Vorgehen gewählt werden. Die Texte in der entsprechenden Anwendung werden hier direkt, und nicht über die übergeordnete Struktogramm-Anwendung aufgerufen. Somit muss diese zusätzliche Funktion neu in der jeweiligen Anwendung implementiert werden.

Als Beispiel für diese Ergänzung habe ich die Rekursions-Anwendung „Matroschkas.flas“ entsprechend angepasst (s. Anhang „Screenshots der geänderten Matroschka-Anwendung“, Seite 109 und praktische Arbeit „virtlab-Juli2012“: `fla/lineareListen/fhoStruct/matroschkas/Matroschkas.flas` und `src/fho/lineareListen/structogram/Matroschkas/Matroschka_new.as`).

Bei dieser Anpassung mussten nur die bereits oben beschriebenen Schritte (das Verschieben der „Spane“, das Anlegen der neuen Buttons und die Ergänzungen im Actionscript-Code) realisiert werden, um die Anwendung in das neue Layout einzupassen. Mit einem Aufwand von ca. 5 Minuten pro Anwendung (die über die „Structogram.flas“ erstellt wurde) können so die früheren Lektionen problemlos in das neue Layout übernommen werden.

4.2.3 Implementierung der neuen Navigation

Die Implementierung der neuen Navigation bildete den größten Entwicklungs-Aufwand der „Änderungen und Erweiterungen der Lektionen“. Hierbei konnte nicht auf der bereits existierenden Navigation aufgebaut werden, da das frühere Navigationskonzept zu wenige Gemeinsamkeiten mit dem neuen hatte. Nur einige der im Hintergrund der Anwendung ablaufenden Funktionen konnten, leicht verändert, wieder verwendet werden. Diese hier beschriebenen Änderungen führte ich selbstständig durch, als Hilfsmittel verwendete ich ein Flash-Handbuch (WESCHKALNIES (2010)).

Im Folgenden soll nun eine erklärende Zusammenfassung der einzelnen Funktionen der neuen Implementierung, und die Kommunikation mit den geladenen Anwendungsseiten gegeben werden (für die komplette Umsetzung s. praktische Arbeit „virtlab-Juli2012“: `fla/lineareListen/` und `src/fho/lineareListen/`).

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

Die „LineareListen.flä“, oder besser der darin befindliche MovieClip „LineareListen“, beinhaltet die komplette neue Navigation. Der Aufbau der verschiedenen Elemente lässt sich am besten grafisch darstellen (s. Anhang, Seite 110 „Übersicht über den Aufbau der LineareListen.flä (eigene Skizzierung)“).

Durch diese Darstellung ist ersichtlich, wie jedes Element über den ActionScript-Code aufgerufen wird. Die doppelt umrandeten Elemente sind hierbei direkt aufrufbar und somit auch in der Hauptklasse (LineareListen_new) direkt deklariert. Die einfach umrandeten Elemente sind hingegen nur über deren Eltern-Elemente (doppelt umrandete) zu erreichen. Zusätzlich zu diesem physischen Aufbau der Flash-Datei werden die Elemente der aufgeklappten Navigation, da diese durch die „Theme.xml“ veränderbar sind, erst zur Laufzeit erstellt.

Nach diesem Überblick über den Aufbau der Flash-Elemente, soll nun auch der implementierte Code vorgestellt werden. Der ActionScript-Code ist zwar kommentiert, doch bietet diese Übersicht über die verschiedenen Funktionen einen besseren Überblick über das Zusammenspiel und die Funktionen die implementiert wurden:

Die Klasse „beginLineareListen“ (beginLineareListen.as)

Wird die kompilierte „LineareListen.flä“ ausgeführt, die „beginLineareListen.swf“, wird zunächst die übergeordnete Actionscript-Datei „beginLineareListen.as“ aufgerufen. Diese recht überschaubare Klasse lädt die eigentliche Hauptklasse (LineareListen_new) und verarbeitet die vom Laborraum übergebenen Funktionen.

Mit Hilfe eines Timers wird ein Objekt der Hauptklasse „LineareListen_new“ angelegt und deren Inhalte dadurch initialisiert.

Den Funktionen FSetCallBackToHome() und FHandleLexicon() wird deren eigentliche Funktionalität durch den Aufruf der Lektion übergeben (s. Kapitel 6.1 Einbinden der neuen Lektion). **FSetCallBackToHome()** wird die Funktion übergeben, das aus der aktuellen Lektion wieder in den Laborraum zurückgekehrt werden kann. **FHandleLexicon()** erhält die Funktionalität des Lexikon-Aufrufes.

Um auf die hier angelegten Variablen und Funktionen auch aus anderen Klassen zugreifen zu können, wurde zudem eine statische Variable („instance“) angelegt. Diese beinhaltet das Objekt dieser Klasse und stellt somit alle Inhalte zur Verfügung.

Das Vorgehen, die beiden Funktionen sowie eine Instanz der Klasse anzulegen, wurde auch bei bereits vorhandenen Lektionen gewählt. Es konnte somit für diese Lektion übernommen werden.

Die Klasse „LineareListen_new“ (LineareListen_new.as)

In der eigentlichen Hauptklasse der Anwendung „LineareListen_new“ werden alle Komponenten der Navigation initialisiert sowie die xml-Datei zur Lektions-Struktur (theme.xml), und damit auch die einzelnen Anwendungsseiten geladen.

Zunächst werden hier alle Variablen die sich direkt auf der Flash-Bühne befinden (vgl. Anhang „Übersicht über den Aufbau der LineareListen.flä (eigene Skizzierung)“ Seite 110) angelegt. Hinzu kommen noch

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

einige Hilfsvariablen, die sich z.B. das aktuelle Kapitel merken oder auch Loader, Requests und Text-Formate.

Die Funktion **LineareListen_new()** wird durch das Anlegen eines neuen Objekts dieser Klasse (in der beginLineareListen.as) aufgerufen. Hier werden Security-Klassen gestartet, eine Instanz der Klasse angelegt und die Funktion init() aufgerufen. Die Instanz (Variable: „**instance**“) dient auch hier wieder dem möglichen Zugriff anderer Klassen auf die hier vorhandenen Elemente und Funktionen.

Auch die Funktionen **handleLexicon()** und **setCallBackToHome()** werden durch die „beginLineareListen.as“ aufgerufen. Hier werden die entsprechenden Buttons (Lexikon und Home) mit den ursprünglich, vom Laborraum übergebenen, Funktionen versehen und als funktionierende Buttons angelegt.

In der **init()** Methode werden nun alle Eigenschaften der neuen Navigationsstruktur initialisiert und die Anweisung zum Laden der Inhalte aufgerufen. Zunächst muss der URL-Pfad ermittelt werden. Dies dient der Ermittlung des Pfades zu den Unterseiten bzw. den verschiedenen Inhalten, die später aufgerufen bzw. geladen werden sollen. Mittels dieses Pfades („**basePath**“, z.B. „build/LineareListen/“) wird im nächsten Schritt der Befehl zum Laden der theme.xml abgegeben, indem die Funktion loadThemeXML() aufgerufen wird. Dieser Funktion wird der Pfad zur theme.xml mitgegeben (basePath+ „theme.xml“ ergibt z.B. „build/LineareListen/theme.xml“).

Im weiteren Verlauf der init()-Methode wird festgelegt, dass die erste Seite (Inhaltsseite „0“) zuerst angezeigt werden soll. Auch verschiedene Positionierungen, z.B. des Seiten-Titels oder der statischen Navigationselemente werden angelegt. Als letzte Funktion werden noch alle vorhandenen Buttons über die Funktion „**buttonFunction()**“ initialisiert. Diese Funktion legt die entsprechenden Mouse-Events für alle auf der Flash-Bühne existierende Buttons an. Sie initialisiert darüber auch den Wechsel zwischen den Inhalten. Auch die Wechselmöglichkeit zwischen ein- und ausgeklappter Zettel-Navigation, oder dem Ein- und Ausblenden der Hilfe-Blase wird hier angelegt.

Die von der init()-Methode aufgerufene Funktion **loadThemeXML()** lädt die Struktur des Lektionsinhaltes aus der jeweiligen „theme.xml“. Nachdem die Datei geladen wurde (complete-Handler) leitet die Funktion diese zur Verarbeitung der darin enthaltenen Informationen weiter, zur **loadedXmlHandler()**. In dieser Funktion werden die Daten der xml-Datei ausgelesen. Hier wird zunächst erfragt, ob es sich um eine Lektion mit oder ohne Unterkapitel handelt. Erst dann können die Inhalte entsprechend ausgelesen und zu der Struktur der Seite „übersetzt“ werden. In dieser „Übersetzung“ werden alle Inhaltsseiten mit deren spezifischen Informationen als Objekt der Klasse „newPageElement“ angelegt. Diese Elemente werden dann nacheinander (in der richtigen Reihenfolge) in einen Array gespeichert. Somit wird schon in diesem Array die Grundlage für das „schrittweise-durchlaufen“ der Lektionen gelegt. Nachdem die Struktur und auch die Inhalte der jeweiligen Lektion angelegt wurden, kann nun die erste Seite (mittels der Methode „newPages()“) geladen werden.

Die Methode **newPages()** wird immer dann aufgerufen, wenn eine neue Inhaltsseite geladen wird. Das geschieht am Anfang nach dem Anlegen der Inhalte und bei jedem Seitenwechsel innerhalb der Lektion. Entsprechend ihrer Aufgabe, die neuen Inhalte zu laden und alle nötigen Funktionen der Seite entsprechend des geladenen Inhaltes zur Verfügung zu stellen, beinhaltet diese Methode die meisten Verzweigungen zu anderen Funktionen der Klasse.

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

Zunächst muss hier die entsprechende neue Lektion geladen werden. Zu diesem Zweck wird die Funktion `loadSWF()` aufgerufen. Nachdem diese geladen ist, wird der Titel der Seite und die Beschriftung der eingeklappten Navigation erneuert. Hierzu muss ermittelt werden, ob es sich um eine Einleitungsseite handelt, oder um eine der Anwendungsseiten. Bei Anwendungsseiten kann einfach der Name der Anwendung und deren Kapitel/Unterkapitel übernommen werden. Bei Einleitungsseiten muss explizit „Einleitung“ zu dem jeweiligen Kapitelnamen ergänzt werden, da dies nicht in der xml steht (hier wird nur der Kapitelname angegeben).

Auch die verschiedenen statischen Buttons müssen überprüft werden. Sollte die angezeigte Inhaltsseite die erste bzw. letzte der Lektion sein, muss der entsprechende Button (zurück bzw. weiter) auf anwählbar oder nicht-anwählbar gesetzt werden.

Auch die nicht statischen Buttons (Lektionsseiten-Buttons in der ausgeklappten Navigation) müssen an die neue Inhaltsseite angepasst werden. Ist es eine Lektion ohne Unterkapitel kann einfach der Button der aktuellen Seite auf „nicht-anwählbar“, und alle anderen auf „anwählbar“ gesetzt werden. Wenn eine Lektion Unterkapitel besitzt wird dieses Anpassen auf die aktuelle Inhaltsseite etwas komplexer, diese Anpassung übernimmt die Methode „`chapterMode()`“. In beiden Fällen werden die Buttons „anwählbar“ bzw. „nicht-anwählbar“ gesetzt über Textformate, `buttonMode` und das hinzufügen bzw. entfernen von Mouse-Events erreicht.

Die Methode **`chapterMode()`** arbeitet recht mathematisch um die Positionierung und Anwählbarkeit der Buttons in der ausgeklappten Navigation zu ermitteln. Über Abfragen wird ermittelt, welcher Button zu der gerade angezeigten Inhaltsseite gehört, und welche anderen Buttons dementsprechend sichtbar und anwählbar sind. Da bei diesem Navigationsmenü die Anordnung aller Buttons davon abhängig ist, welches Untermenü gerade aufgeklappt ist, muss die Positionierung jedes Buttons immer neu ermittelt werden. So hängt die Positionierung der Unterkapitel-Punkte davon ab, ob das vorherige Kapitel ausgeklappt ist. Ist ein Unterkapitel ausgeklappt, hängt es auch von der Anzahl der Unterpunkte dieses Kapitels ab. Um diese äußerst flexible Positionierung zu ermöglichen, müssen alle Eventualitäten einbezogen und jeweils entsprechende Konstanten zur Positionierung hinzugerechnet werden.

`loadSWF()` stellt alle Methoden zum Laden einer neuen Inhaltsseite zur Verfügung. Zunächst wird hier die aktuell sichtbare Inhaltsseite entfernt und der Preloader, der den Fortschritt des Ladens der neuen Seite anzeigen soll, zurückgesetzt und sichtbar gemacht. Sobald das Laden der neuen Inhaltsseite beginnt, zeigt der Preloader mittels eines Prozess-Events und der zugehörigen Funktion „**`progressHandler()`**“ an, wie viel Prozent des Inhaltes bereits geladen sind. Dieser Prozess wird mittels eines Complete-Handlers überwacht. Sobald der Inhalt geladen ist beendet die Funktion „**`completeHandler()`**“ den Preloader und zeigt die entsprechende Lektion an.

Im letzten Schritt des Ladens der neuen swf-Datei wird die Hilfe-Blase der jeweiligen Inhaltsseite neu erstellt. Dies geschieht über die Klasse „`fhoHelp_new`“, dieser Seite wird der entsprechende Hilfe-Text der xml-Datei (`theme.xml`), mittels deren Funktion `setHelp()` übergeben.

Nun bleiben noch vier kleinere Funktionen zum Steuern und Verarbeiten der zur Laufzeit erstellten Buttons (im ausgeklappten Navigationsmenü). Der **`mc_iconHandler()`** ermittelt hierbei welcher Button geklickt wurde. Die entsprechende Zuweisung dieser Funktion wird bei der Erstellung der Buttons in der Klasse „`newPageElement`“ vorgenommen. Die Funktion ermittelt den Anchor des Buttons (im `theme.xml`

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

festgelegte, eindeutige Erkennung der einzelnen Inhalte) und gibt diesen Anchor-Wert an die Funktion `loadAppByAnchor()` weiter. **`loadAppByAnchor()`** nimmt nun diesen Anchor-Wert und sucht die entsprechende Seite, die nun geladen werden soll und ruft auch die Funktion `newPages()` auf, um diese zu laden.

Die Funktionen **`mouseOverHandler()`** und **`mouseOutHandler()`** übernehmen, wie die Namen schon vermuten lassen, den hover-Effekt der zur Laufzeit erstellten Buttons. Wie der `mc_iconHandler()` werden auch diese Funktionen den Buttons bei deren Erstellung in der Klasse „newPageElement“ zugewiesen.

Die Klasse „newPageElement“ (`newPageElement.as`)

Diese überschaubare Klasse beinhaltet das Anlegen der verschiedenen Inhaltsseiten als Objekte dieser Klasse. Im Konstruktor werden alle Inhalte und Daten der „theme.xml“ als String bzw. Zahl und die Position des neu anzulegenden Elements in der Inhaltsseiten-Hirarchie erwartet:

```
public function newPageElement(positionS:uint, chapterS:uint, typeS:String, titleS:String,  
swfS:String, helpS:String, diagramFileS:String, solutionS:String, iconS:String, lexiconS:String,  
anchorS:String)
```

Sobald dieser Konstruktor mit den entsprechenden Werten aufgerufen wird erstellt er ein Objekt indem alle übergebenen Informationen gespeichert, bzw. verarbeitet werden.

Die Übergabe des Button-Typs wird mittels der Funktion **`newMcButton()`** weiter verarbeitet. Hier wird der Button-Typ zu einem MovieClip des entsprechenden Buttons umgewandelt, und dieser der Bühne bzw. dem MovieClip-Container „mc_icon_container“ hinzugefügt.

Die nächste und auch letzte Weiterverarbeitung betrifft die Positionierung und Beschriftung des Objekts, bzw. der entsprechenden Buttons. Diese Punkte werden in der **`init()`**-Methode realisiert. Hier wird nach dem Typ des Objekts bzw. der Lektionsseite sortiert, der Button benannt und initialisiert. So müssen z.B. Intro- oder Chapter-Inhaltsseiten mit dem Zusatz „Einleitung“ benannt werden. Dies betrifft sowohl den eigentlichen Inhalt- bzw. Chapter-Punkt, oder auch das entsprechende Einleitungs-Icon mit dessen Text. Bei Anwendungsseiten muss allerdings nur das entsprechende Icon und dessen Beschriftung übernommen werden.

Wenn diese Zuordnung und Initialisierung abgeschlossen ist werden die Buttons/Icons, wenn keine Unterkapitel vorhanden sind, mittels eines errechneten Abstandes positioniert. Wenn Unterkapitel vorhanden sind übernimmt diese Anordnung die Funktion „chapterMode()“ in der Hauptklasse, sobald eine Lektion geladen wird.

Die Klasse „fhoHelp_new“ (`fhoHelp_new.as`)

Diese Klasse wurde größtenteils aus der Vorlage der Klasse „fhoHelp“ der bereits implementierten Rekursions-Lektion übernommen. Sie übersetzt in der Funktion **`setHelp()`** die im Hilfetext der theme.xml enthaltenen css-Befehle, und wendet sie auf den entsprechenden Hilfetext an. Auch die darin enthaltenen Verlinkungen mittels Anchors werden über die Funktion **`xmlLinkHandler()`** hier initialisiert.

4.3 Erstellen einer neuen Lektionsstruktur (theme.xml)

Mit der neuen Navigation haben sich auch die Möglichkeiten des Aufbaus der theme.xml geändert. Mit der früheren Navigation konnten nur Lektionen mit Unterkapiteln angelegt werden. Im neuen Navigationsrahmen ist es jetzt möglich Lektionen ohne Unterkapitel (wie z.B. die Listen-Lektion) mit bis zu fünf Anwendungsseiten (zusätzlich zur Einleitungsseite) zu erstellen.

Wie in der früheren Implementierung können auch Lektionen mit Unterkapiteln weiterhin erstellt werden. Hier können z.B. drei Unterkapitel geschaffen werden, die jeweils bis zu drei Anwendungsseiten (zusätzlich zu je einer Einleitungsseite) beinhalten können.

Da die theme.xml für diese verschiedenen Varianten auch leicht anders aufgebaut ist, stelle ich nun die einzelnen Teile dieser Seitenstruktur-xml dar. Mit dieser Erklärung können neue Lektionen einfach nach den individuellen Erfordernissen angelegt werden. Dies kann geschehen ohne, wie das bisher der Fall war, dass in der zugehörigen Flash-Datei noch Anpassungen vorgenommen werden müssen.

Allgemeines

Je nachdem ob eine Lektion mit oder ohne Unterkapitel angelegt werden soll, ergeben sich unterschiedliche Strukturen der xml. Beiden gemeinsam ist jedoch das Anlegen der Anwendungsseiten, diese werden jeweils als <page>-Tag angelegt.

Eine Page muss hierbei Werte für type, title, swf, icon, lexicon und anchor besitzen. Zusätzlich können der diagramFile und eine solution angegeben werden:

- type: Art der Anwendung - INSTRUCTION, EXPERIMENT oder STRUCTOGRAM
- title: Hier kann der Name der Anwendung eingetragen werden, dieser wird dann als Navigationspunkt und Seitenüberschrift angezeigt.
- swf: Pfad zu der swf-Datei der Anwendung. Bei Anwendungen die über ein Struktogramm gesteuert werden (Experimente und einige Instruktionsanwendungen) wird der Pfad zur structogram.swf angegeben.
- icon: Hier wird angegeben, welches Icon verwendet werden soll (für Instruktionsanwendungen „sInstruction“, für Experimente „sExperiment“ und für Struktogramme „sStructogram“).
- lexicon: Hier wird der Anchorwert der zur Lektion zugehörigen Lexikon-Seite mitgegeben, diese kann dann durch klick auf das Lexikon-Symbol aufgerufen werden.
- anchor: Hier wird der Anchor der Anwendung festgelegt. Dieser Anchorwert muss eindeutig sein, da über diesen die verschiedenen Anwendungen aufgerufen werden.
- diagramFile: Wenn die Anwendung über ein Struktogramm gesteuert wird, muss hier die zugehörige dia-Datei, die den Inhalt des Struktogramms enthält, mit ihrem Pfad angegeben werden.
- solution: Für Struktogrammentwicklungen wird hier die dia-Datei mit der Lösung des Struktogramms übergeben, ebenfalls durch Angabe des Pfades

Um das Struktogramm „Freunde besuchen“ einzubinden muss z.B. folgende Page angelegt werden:

```
<page type="STRUCTOGRAM" title="Freunde besuchen" swf="fhoStruct/Structogram.swf"
      diagramFile="fhoStruct/diagramFiles/besuche00-recursion.dia"
      solution="fhoStruct/diagramFiles/besuche00-recursion_solution.dia">
```

4 ÄNDERUNGEN & ERWEITERUNGEN FÜR DIE LEKTIONEN

```
icon="sStructogram" lexicon="R001" anchor="A210">
...
</page>
```

Innerhalb des jeweiligen Page-Tags wird der Tag für den Hilfetext der Anwendung (<help>) aufgenommen. Hierbei kann mittels CDATA ein Text mit css-Befehlen und Links eingefügt werden:

```
<![CDATA[
    • Möchtest Du mehr über Rekursion erfahren?
    <a href='event:lexicon, R005'>Ja</a> <br>
    • Oder nochmals nachschauen wie die Fakultätsfunktion berechnet wird?
    <a href='event:loadApp, A110'>Ja</a>
]]>
```

Für das Link-Event sollte hier jeweils, wie im oberen Beispiel sichtbar, dass entsprechende Event

- showSolution - zum Anzeigen der Lösung eines Struktogramms
 - loadApp - zum Laden einer anderen Inhaltsseite
 - lexicon - zum Sprung in eine angegebenen Lexikonseite
- sowie der referenzierende Anchorpunkt angegeben werden.

Lektionen ohne Unterkapitel

Um eine Lektion ohne Unterkapitel zu erstellen muss ein <theme>-Tag angelegt werden, dessen Form auf „pages“ gesetzt wird. Alle anderen mitzugebenden Werte sind ähnlich der Pages. Der title enthält den Namen der Lektion, der auch in der Navigation angezeigt wird. In der swf-Datei wird der entsprechende Pfad zur Einleitungsseite mitgegeben. lexicon enthält den passenden Anchor zum Lexikoneintrag, und der Anchor wiederum eine eindeutige Referenz auf diese Seite. Im Punkt icon ist als Lektions-Startseite immer „sIntro“ eingetragen. Die oben beschriebenen angelegten „Pages“ können innerhalb dieser „Theme“ nach belieben angeordnet werden.

```
<theme title="Listen" form="pages" swf="Introduction/introListen.swf"
icon="sIntro" lexicon="R006" anchor="A200">
    <page ...> ...</page>
    <page ...> ...</page>
    ...
</theme>
```

Als Beispiel kann hier in der Listen-Lektion die theme.xml (s. praktische Arbeit „virtlab-Juli2012“: build/virtlab/LineareListen) mit dem xml-Beispiel „theme1Kapitel.xml“ im selben Ordner ersetzt und getestet werden. Diese xml zeigt alle möglichen Formen der Anwendungen (Instruktion, Experiment, Struktogrammentwicklung). Die Inhalte wurden mit den bereits in der Rekursions-Lektion vorhandenen Inhalten ergänzt. Der Inhalt der Seiten ist somit nicht aufeinander abgestimmt und dient nur der Veranschaulichung.

Lektionen mit Unterkapitel

Wenn eine Lektion mit Unterkapiteln angelegt werden soll, weicht die Vorgehensweise leicht ab. Auch hier muss zunächst ein `<theme>`-Tag erstellt werden. Dieser Tag hat die selben Eigenschaften wie das `Theme-Tag` in einer Lektion ohne Unterkapitel. Nur die `form` muss hier auf „chapter“ umgestellt werden (`form="chapter"`). Auch hier kann die Einstiegsseite der Lektion wieder als Pfad zur `swf`-Datei angegeben werden.

Alle weiteren Einleitungsseiten sind Einstiegsseiten zum jeweiligen Kapitel und werden mittels eines `<chapter>`-Tags angelegt. Die Attribute `title`, `swf`, `lexicon` und `anchor` sind, wie in den vorherigen Elementen beschrieben, auch hier anzugeben. Der `type` wird in einem `chapter` stets auf „CHAPTER“, das `icon` stets auf „sIntroC“ gesetzt. Die zugehörigen Anwendungsseiten (`<page>`) können auch hier innerhalb der Kapitel (`<chapter>`) nach belieben angeordnet werden:

```
<chapter type="CHAPTER" title="Listen" swf="Introduction/introListen.swf"
        icon="sIntroC" lexicon="R001" anchor="A301">
    <page ...> ....</page>
    <page ...> ....</page>
    ...
</chapter>
```

Als Beispiel kann auch hier in der Listen-Lektion die `theme.xml` (s. praktische Arbeit „virtlab-Juli2012“: `build/virtlab/LineareListen`) mit dem `xml`-Beispiel „`theme3Kapitel.xml`“ im selben Ordner ersetzt und getestet werden. Diese `xml` zeigt, wie die Navigation bei einer Lektion mit drei Unterkapiteln aussehen würde. Die Inhalte wurden mit den bereits in der Rekursionslektion vorhandenen Inhalten ergänzt. Der Inhalt der Seiten ist somit nicht aufeinander abgestimmt und dient nur der Veranschaulichung.

5.1 Überblick über die Lektion

Die neue Lektion zum Thema Listen soll Lernenden wichtige Begriffe, Konzepte und den generellen Umgang mit linearen Listen näher bringen. Um dies zu erreichen wurden vier Lektionsseiten sowie ein zusätzlicher Eintrag in das Lexikon des VILs konzipiert.

Die Einstiegsseite der Lektion stellt die Einführung in das Thema Listen dar. Hier werden Anwendungsbeispiele und wichtige Begriffe aufgezeigt.

Auf die Einstiegsseite folgen drei verschiedene Anwendungsseiten. In der Visualisierung werden die drei grundlegenden Eingriffsmöglichkeiten in die Listen - Suchen, Einfügen und Löschen - erklärt und anschaulich dargestellt. Im Experiment können die Lernenden diese Konzepte selbst erproben und verschiedene Vorgehensweisen testen. Der Struktogrammentwurf ermöglicht den Entwurf eines eigenen Listen-Algorithmus.

Im Lexikon-Eintrag werden wichtige Begriffe noch einmal genauer dargestellt und Anwendungen aus der Praxis vorgestellt.

Aus diesen Inhalten ergibt sich die folgende Struktur für die neue Lektion:

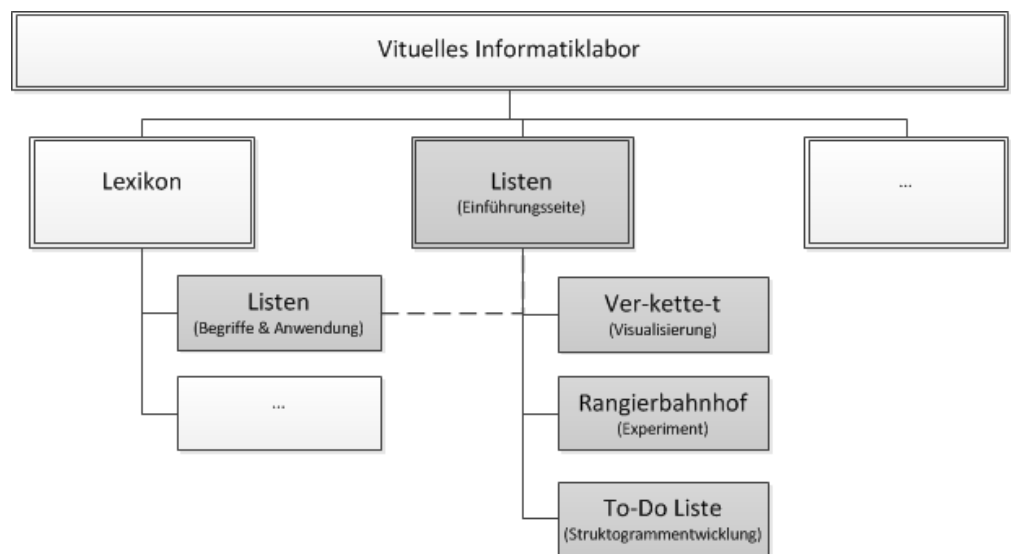


Abb. 19 Struktur der Listen-Lektion (eigene Skizzierung)

5.2 Inhalte der Lektion

Im Folgenden werden die neu konzipierten Inhalte vorgestellt und genau beschrieben. Um eine bessere Übersicht über die Konzeption zu erhalten wurden die „Texte“ der einzelnen Inhalte (jeweils letzter Unterpunkt) folgendermaßen gekennzeichnet:

- ***Bold Italic***: Informationen und Anmerkungen zur Umsetzung
- Unterstrichen: Links zu anderen Inhalten
- **Farbig unterstrichen**: Tool-Tipps (die zugehörigen Texte finden sich im Kapitel 5.2.6 Tool-Tip Texte)

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Die Positionierung und Darstellung der Textelemente wurde aus den bereits erstellten Lektionen übernommen. Alle eingeführten Änderungen in Bezug auf die früheren Lektionen wurden im Kapitel 4 (Änderungen und Erweiterungen für die Lektionen) beschrieben.

Auch in der Wortwahl und dem Schreibstil der Texte wurde sich an der bereits implementierten Lektionen orientiert. Als inhaltliche Grundlage der Texte dienten die im Kapitel 2.2 „Listen in der Informatik“ aufgezeigten Informationen, sowie für den Lexikoneintrag eine zusätzliche Quelle zur Praxis-Anwendung (WIRTH (1968) S. 183-199).

5.2.1 Einführungsseite Listen

Für die Einführungsseite wurde das Layout der bereits vorhandenen Rekursions-Einführungsseiten übernommen. Somit mussten nur die Texte und eine passende Visualisierung entworfen werden.

5.2.1.1 Einführungsseite – Aufbau & Interaktion

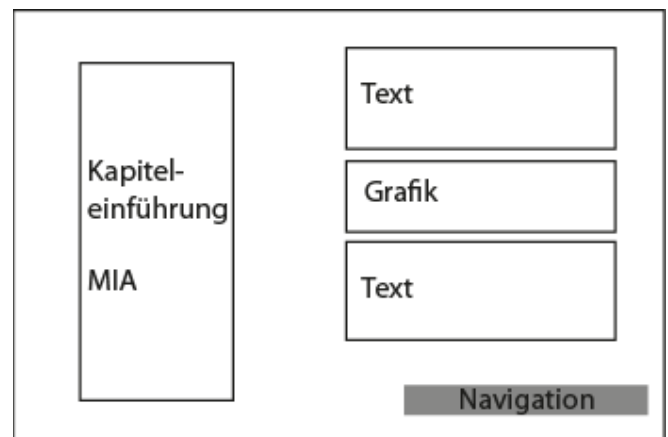


Abb. 20 grobe Aufteilung der Elemente der Einstiegsseite (eigene Skizzierung)

aktive Navigationselemente (im Bein bzw. auf dem Wäschezettel):

Lexikon-Button	führt zu R006 (Lexikon Listen-Eintrag)
Home-Button	Link in den Laborraum
M+I-Button	Link zum MI-eLearning Portal
Weiter-Pfeil	führt zu A201 (Ver-kette-t, Visualisierung)
Menü-Punkt Ver-kette-t	führt zu A201 (Ver-kette-t, Visualisierung)
Menü-Punkt Rangierbahnhof	führt zu A202 (Rangierbahnhof, Experiment)
Menü-Punkt To-Do Liste	führt zu A203 (To-Do Liste, Struktogrammentwicklung)

aktive Interaktionselemente:

Es sind keine Interaktionselemente vorhanden.

5.2.1.2 Einführungsseite – Rohrpost Visualisierung

Die Idee ein Rohrpost-System zur Veranschaulichung einzusetzen entwickelte sich aus dem Beispiel des „Partybesuchs“ (s. Kapitel 2.2.2 Verschiedene Anwendungen aus dem Alltag). Den dort vermittelten Konzepten wurden, durch die Veranschaulichung als Rohrpost, weitere Eigenschaften der linearen Listen hinzugefügt, z.B. dass ein System nur in eine Richtung funktioniert.

Diese Grafik veranschaulicht den im Text erklärten Aufbau einer Liste mittels eines Rohrpostsystems (head → Mark → Susi → Dani → NULL).

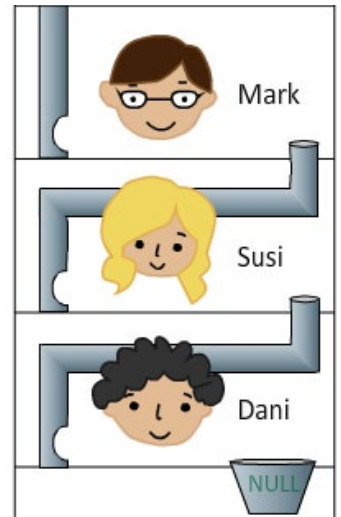


Abb. 21 Grafik des Rohrpost-Systems (eigener Entwurf)

5.2.1.3 Einführungsseite – Texte

Kapiteleinführung und MIA-Sprechblase:

Hast du schon mal was von Listen gehört?

[genauer: verkettete lineare Liste]

Lass uns zusammen herausfinden wie Listen funktionieren!

Seitentext:

Listen kommen zum Einsatz, wenn Daten dynamisch gespeichert werden sollen. Dynamisch bedeutet hierbei, dass die Anzahl der Elemente frei variierbar ist, auch während das Programm läuft. Im Gegenteil dazu ist z.B. bei einem Array bzw. Feld die Anzahl der Elemente festgelegt und kann, wenn das Programm läuft, nicht geändert werden.

Rohrpost-Grafik (head → Mark → Susi → Dani → NULL)

Dieses Rohrpost-System zeigt dir den Aufbau einer typischen Liste.

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Das Rohr zu Mark ist sozusagen der Startpunkt der Liste, der sogenannte „head“. Er besteht aus einem „Verweis“ (Rohr) auf den ersten „Knoten“ (Mark).

Susi ist nur zu erreichen, wenn Mark die Post weiterleiten kann. Sobald Mark nicht mehr an seinem Platz ist können Susi und damit auch Dani nicht mehr erreicht werden.

Dani, die letzte Person der Liste, kann die Post nicht weiterleiten weil von ihr kein Rohr mehr weg führt (Verweis auf „NULL“).

Das Rohr zu Susi bzw. Mark ist zu steil um die Post zurückzuschicken.

Nachdem du jetzt weißt was eine Liste ist, kannst du auf den nächsten Seiten dieser Lektion herausfinden wie eine Liste funktioniert und selbst einige Aufgaben lösen.

5.2.2 Ver-kette-t (Visualisierung/Instruktionsanwendung)

Für diese Anwendung wurde das Layout der bereits vorhandenen Lektionen zum Großteil übernommen und nur durch kleine Änderungen ergänzt. Hierzu zählten das Verschieben des Kontrollbuttons und die Erweiterung um eine Beschreibung der Interaktion mit der Anwendung (s. Kapitel 4.1.2 Änderungsvorschläge - Verschiedene Interaktionsmöglichkeiten).

5.2.2.1 Visualisierung – Aufbau & Interaktion

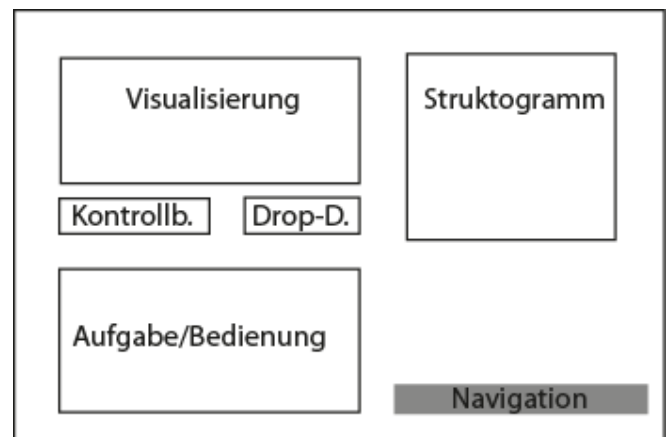


Abb. 22 grobe Aufteilung der Elemente der Instruktionsanwendung (eigene Skizzierung)

aktive Navigationselemente (im Bein bzw. auf dem Wäschezettel):

Lexikon-Button	führt zu R006 (Lexikon Listen-Eintrag)
Home-Button	Link in den Laborraum
M+I-Button	Link zum MI-eLearning Portal
Weiter-Pfeil	führt zu A202 (Rangierbahnhof, Experiment)
Zurück-Pfeil	führt zu A200 (Listen, Einstiegsseite)
Menü-Punkt Listen	führt zu A200 (Listen, Einstiegsseite)
Menü-Punkt Rangierbahnhof	führt zu A202 (Rangierbahnhof, Experiment)
Menü-Punkt To-Do Liste	führt zu A203 (To-Do Liste, Struktogrammentwicklung)

aktive Interaktionselemente:

- Kontrollbuttons: Durchlaufen bzw. Starten der Visualisierung (s. Unterpunkt 5.2.2.2 Visualisierung – Ver-Kette-t Animation)
- Drop-Down: Verschiedene Inhalte der Visualisierung; fünf Unterpunkte: Suchen, Einfügen am Anfang, Einfügen in der Mitte, Löschen am Anfang und Löschen in der Mitte (s. Unterpunkt 5.2.2.2 Visualisierung – Ver-Kette-t Animation)
- Struktogramm: Tool-Tipps zu jedem Punkt (s. Unterpunkt 5.2.2.3 Visualisierung – Texte / Step-By-Step)
- Aufgabe/Bedienung: Möglichkeit zwischen der Aufgabenbeschreibung und der Beschreibung der Bedienung zu wechseln

5.2.2.2 Visualisierung – Ver-Kette-t Animation

Die Idee, anhand einer Kette die verschiedenen Interaktionen mit einer lineare Liste aufzuzeigen, entstammte dem Begriff „verkettete Liste“, der in der Literatur oft verwendet wird. Anhand dieser Veranschaulichung können das Suchen, Einfügen und Löschen in einer Liste gut dargestellt werden. Indem man die Kette „aufhängt“ kann z.B. sehr gut aufgezeigt werden, wie Elemente gelöscht werden. Wenn zu einem Element keine Verbindung mehr besteht „fällt“ es aus dem Bild und ist somit gelöscht.

Suchen in einer Liste:

Der Algorithmus, der das Durchlaufen und somit auch das Suchen in einer Liste koordiniert, wurde für das Durchlaufen der Kette angepasst (Abb. 23).

Die im Folgenden aufgezeigten Ansichten stellen die Visualisierung dieses Algorithmus dar. Hierbei wurden Punkte dargestellt, bei denen ein Struktogrammbefehl eine darstellbare Reaktion hervorruft.

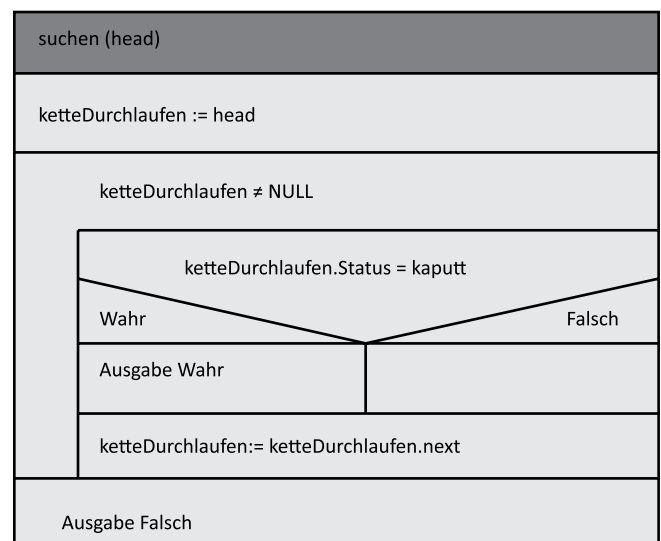


Abb. 23 Struktogrammentwurf zum Suchen in einer Liste/Kette (eigener Entwurf)

1. Ansicht - Suchen

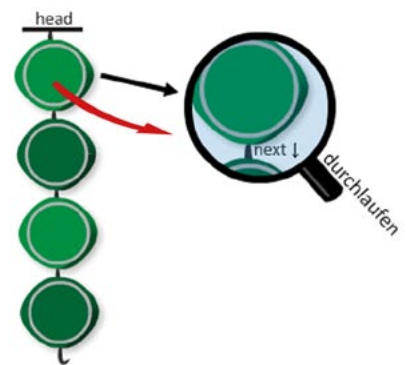
Die Visualisierung startet in diesem Fall als Kette mit vier Elementen. Da keine Variable außer dem „head“ mitgegeben wird, ist zunächst auch nur die Kette selbst sichtbar.

Der zweite Schritt des Struktogramms wird aufgerufen und der „head“ wird der Variable „elementeDurchlaufen“ zugewiesen. Während dieser Zuweisung wechselt die Visualisierung von der ersten zur zweiten Ansicht. Dies geschieht indem die Lupe „durchlaufen“, wie der rote Pfeil andeutet, über dem ersten Element erscheint und dann zur Seite fährt (zweite Ansicht).



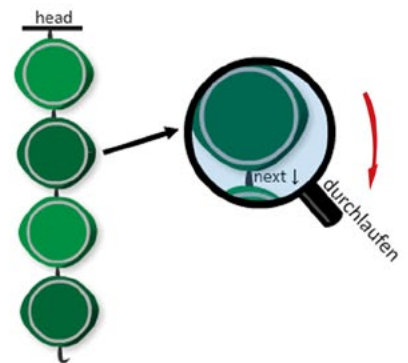
2. Ansicht - Suchen

In dieser Ansicht wird die if-Abfrage das erste Mal durchlaufen. Da das Element nicht „kaputt“ ist wird der Abzweig „Falsch“ gewählt. Der nächste Befehl setzt die Lupe auf das nächste Element (Ansicht drei).



3. Ansicht - Suchen

Dieser Abschnitt läuft wie der vorherige ab, die if-Abfrage wird im „Falsch“-Abzweig durchlaufen. Die Lupe wird im nächsten Befehl auf das nächste Element (vierte Ansicht) gesetzt.



4. Ansicht - Suchen

Hier wird das kaputte Stück gefunden. Das Struktogramm läuft in der if-Abfrage den „Wahr“-Weg. Die Anwendung stoppt mit der Lupe auf dem „kaputten“ Element und dem Struktogramm-Befehl „Ausgabe Wahr“.

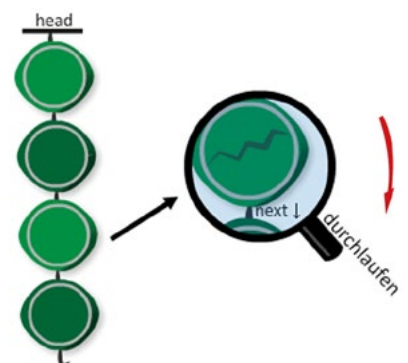


Abb. 24 Ansichten zum Suchen in einer Liste/Kette (eigene Skizzierung)

Einfügen in eine Liste:

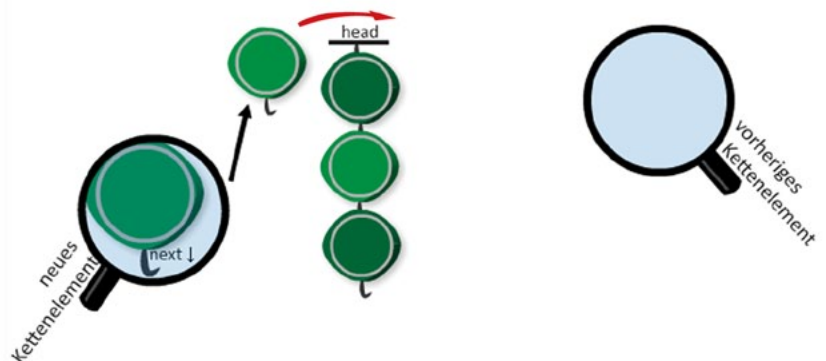
einfügen (vorherigesKettenstück, neuesKettenstück, head)	
vorherigesKettenstück = NULL	
Wahr	Falsch
neuesKettenstück.next := head	merkVerweis := vorherigesKettenstück.next
head := neuesKettenstück	vorherigesKettenstück.next := neuesElement
	neuesKettenstück.next := merkVerweis

Abb. 25 Struktogrammentwurf zum Einfügen in eine Liste/Kette (eigener Entwurf)

Dieser Algorithmus beschreibt das Einfügen eines neuen Elements in eine bestehende Kette. Hierzu wurden zwei Ansichten der Visualisierung entworfen. Die erste zeigt den „Wahr“-Weg, wenn ein Element am Anfang der Liste eingefügt wird. Die zweite veranschaulicht den Fall, dass ein Element in der Mitte der Kette eingefügt werden soll.

1. Ansicht - Einfügen am Anfang

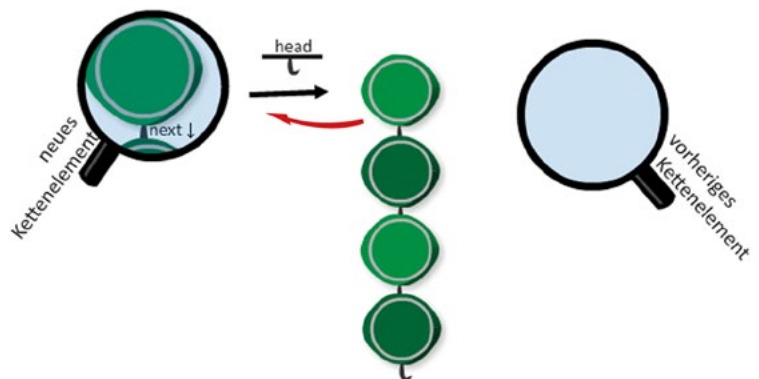
Die ersten Befehle des Struktogramms laufen mit dieser Ansicht ab. Da das „vorherige Element“ nicht gegeben ist wird hierfür zur Veranschaulichung eine leere Lupe angezeigt.



Erst nachdem die if-Anweisung den Zweig „Wahr“ gewählt hat, wird hier zur zweiten Ansicht übergegangen. Wie der rote Pfeil andeutet muss hierzu dem „neuen“ Element das erste Element der Kette/Liste angehängt werden.

2. Ansicht - Einfügen am Anfang

Diese Ansicht stellt den ersten Punkt der if-Anweisung für „Wahr“ dar. Für den Übergang zur nächsten Ansicht wird das neue Element, wie hier angedeutet, unter den „head“ platziert.



3. Ansicht - Einfügen am Anfang

Sobald der letzte Punkt der if-Anweisung bearbeitet wurde ist das „neue Kettenelement“ in die Liste/Kette integriert, das Struktogramm somit komplett durchlaufen.

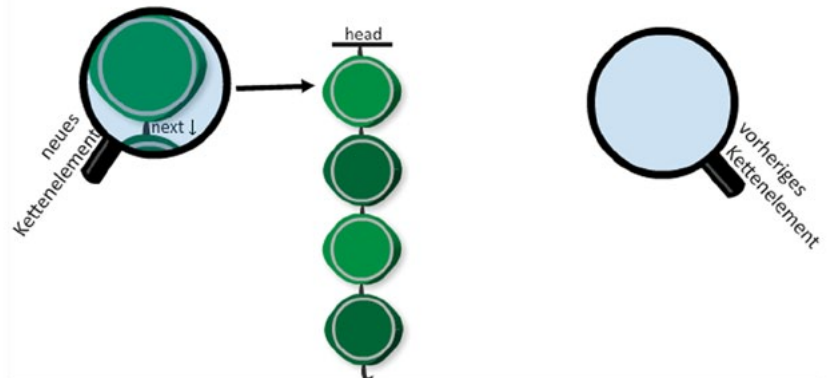
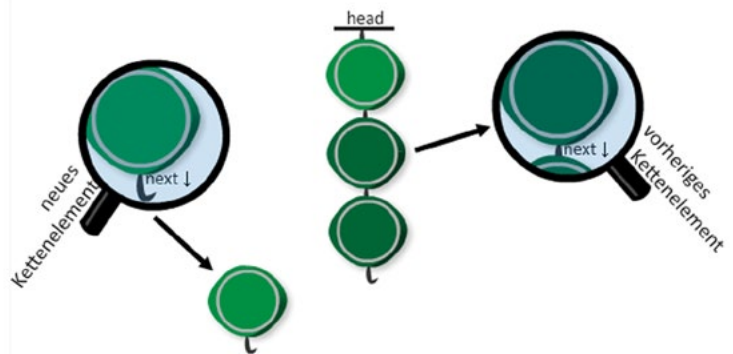


Abb. 26 Ansichten zum Einfügen am Anfang einer Liste/Kette (eigene Skizzierung)

1. Ansicht - Einfügen in der Mitte

Die ersten Befehle des Struktogramms laufen mit dieser Ansicht ab. Erst nachdem die if-Anweisung den Zweig „Falsch“ gewählt hat, wird zur zweiten Ansicht übergegangen.

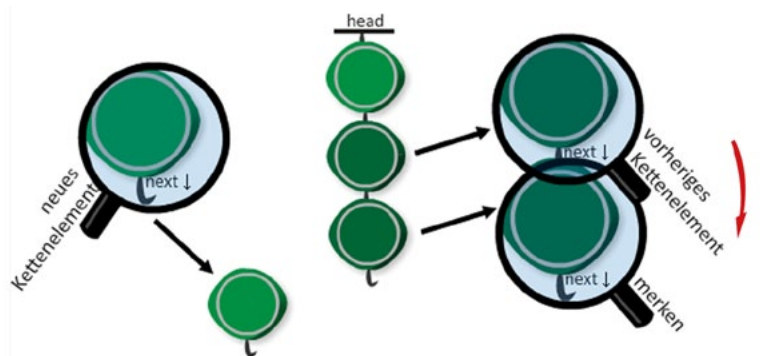


Im ersten Schritt der if-Anweisung wird eine neue Variable zugewiesen. Hierzu erscheint zum Übergang in die zweite Ansicht die zweite Lupe „merken“ über der Lupe des „vorherigen“ Elements. Diese enthält, wie in der zweiten Ansicht sichtbar, das Folgeelement des „vorherigen“ Elements.

2. Ansicht - Einfügen in der Mitte

Diese Ansicht stellt den ersten Punkt der if-Anweisung für „Falsch“ dar.

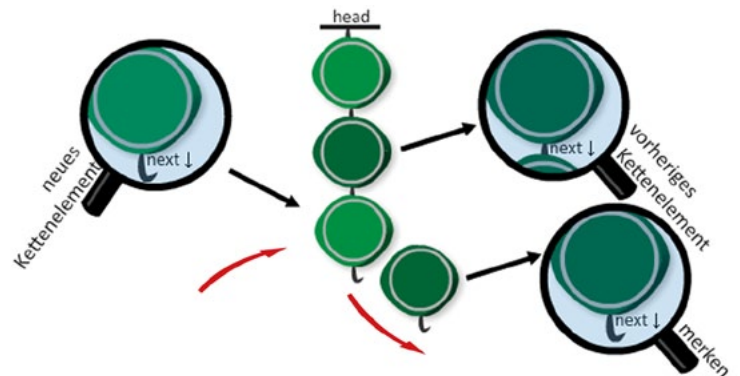
Für den Übergang zur nächsten Ansicht werden die beiden Lupen mit ihren Elementen, wie in der dritten Ansicht angedeutet, vertauscht.



3. Ansicht - Einfügen in der Mitte

Diese Ansicht stellt den zweiten Punkt der if-Anweisung dar.

Zum Übergang in die letzte Darstellung wird die „merken“-Lupe und deren Element an das „neue Kettenelement“ angebracht.



4. Ansicht - Einfügen in der Mitte

Sobald die „merken“-Lupe und deren Element an die des „neuen Kettenstücks“ angehängt wurde, ist das Struktogramm komplett durchlaufen. Die „merken“-Lupe wird ausgeblendet. Das neue Kettenelement erscheint nun auch in der Komplettansicht der Kette, wurde also in diese integriert.

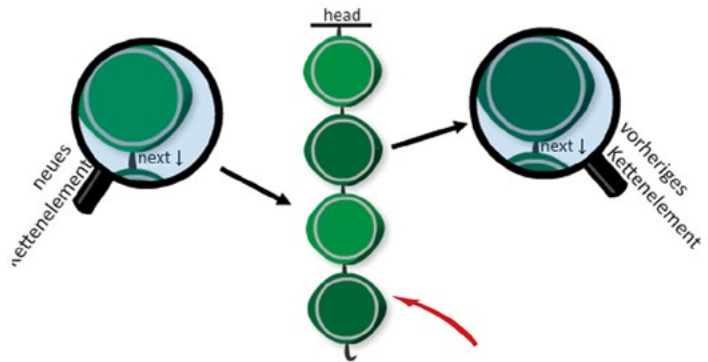


Abb. 27 Ansichten zum Einfügen in der Mitte einer Liste/Kette (eigene Skizzierung)

Löschen aus einer Liste:

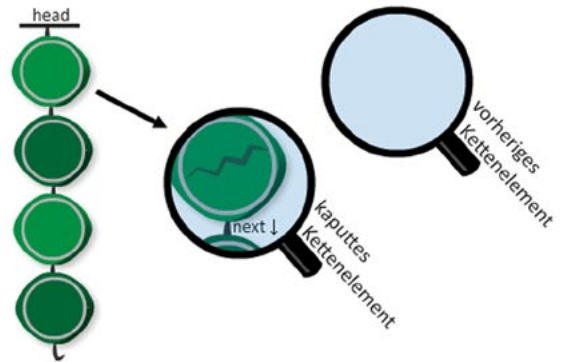
löschen (vorherigesKettenstück, löscheKettenstück, head)	
vorherigesKettenstück = NULL	
Wahr	Falsch
head := löscheKettenstück.next	vorherigesKettenstück.next := löscheKettenstück.next

Abb. 28 Struktogrammentwurf zum Löschen aus einer Liste/Kette (eigener Entwurf)

Das letzte Struktogramm der Instruktionsanwendung erklärt das Löschen aus einer Kette. Auch hierzu wurden wieder zwei Visualisierung entworfen. Im „Wahr“-Zweig wird ein Element am Anfang der Liste/Kette gelöscht, im „Falsch“-Zweig eines in der Mitte der Liste/Kette.

1. Ansicht - Löschen am Anfang

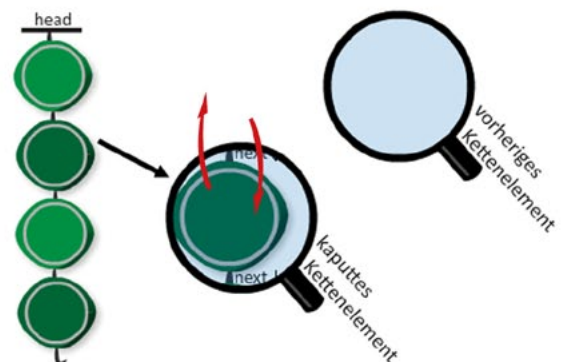
Die ersten Schritte des Struktogramms laufen ohne Veränderung in der Visualisierung ab. Da es nur eine Anweisung zum Löschen gibt laufen während diesem Punkt alle folgenden Ansichten ab.



2. Ansicht - Löschen am Anfang

Um zur zweiten Ansicht zu gelangen wurde lediglich die Lupe des kaputten Elements (erste Ansicht) auf ihr Folgeelement verschoben.

Um zur letzten Ansicht zu gelangen wird diese Lupe, diesmal mitsamt dem gezeigten Element nach oben geschoben.



3. Ansicht - Löschen am Anfang

Sobald das letzte Element nach oben geschoben wurde, fehlt dem zu löschenden Element die Verbindung zur Kette/Liste und es fällt nach unten weg.

Die dritte Ansicht bildet das Endresultat dieser Listen-Operationen.

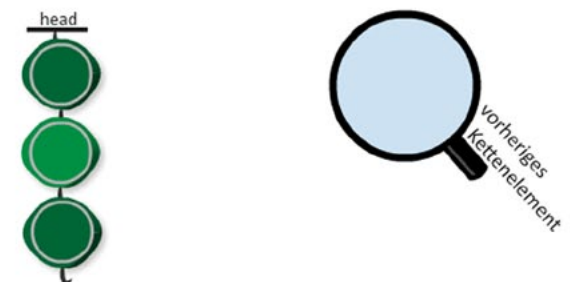


Abb. 29 Ansichten zum Löschen des ersten Elements einer Liste/Kette (eigene Skizzierung)

Löschen in der Mitte

Das Löschen in der Mitte der Kette verläuft wie oben beschrieben. Nur die Anfangs-Position der angezeigten Lupen ändert sich. In dieser Visualisierung wird das vorletzte Element der Kette gelöscht:

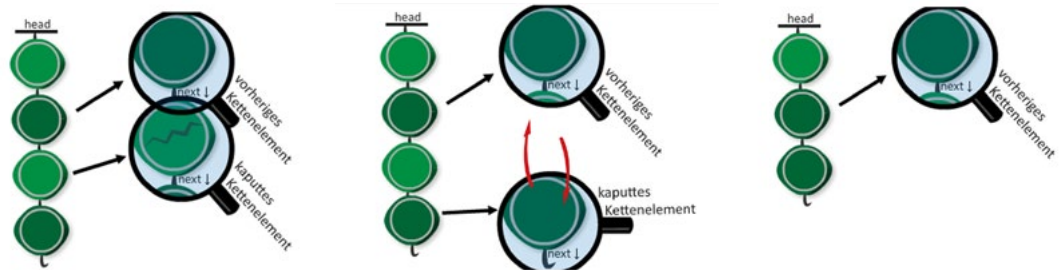


Abb. 30 Ansichten zum Löschen aus der Mitte einer Liste/Kette (eigene Skizzierung)

Interaktionsmöglichkeiten

Mittels der Kontrollbuttons kann entweder die Visualisierung schrittweise durchlaufen, oder die durchlaufenen Schritte rückgängig gemacht werden. Die Schrittabfolge stimmt mit den Struktogrammpunkten und der damit verbundenen geschilderten Visualisierung überein. Für den animierten Durchlauf (Play/Pause-Button) werden die einzelnen Schritte nacheinander mit einer kleinen Verzögerung abgespielt. Die Animation stoppt wenn das Ende der Animation erreicht ist, oder der Play/Pause-Button nochmals geklickt wird.

5.2.2.3 Visualisierung – Texte

Aufgabenbeschreibung:

Wie das Suchen in einer Liste und das Einfügen und Löschen eines Knotens funktionieren, kannst du mit dieser Visualisierung genau beobachten. Jedes Kettenelement hat hier einen „Status“ (kaputt oder ganz) und einen Platz, um eine Referenz auf ein anderes Kettenelement aufzunehmen („next“).

Im Struktogramm siehst du parallel zur Animation den Algorithmus, der für die ausgewählte Aufgabe verwendet wird. Du kannst also genau nachvollziehen was der jeweilige Struktogramm-Befehl bewirkt. Der Befehl „Ausgabe“ beendet hierbei den Durchlauf und gibt den entsprechenden Wert zurück.

Beschreibung der Bedienung:

Für die Animation kannst du zwischen den drei Aufgaben – Suchen, Einfügen und Löschen – wählen (Drop-Down unter der Animation).

Gestartet wird die Anwendung über die Kontrollbuttons (Buttons unter der Animation). Sie bieten folgende Möglichkeiten, von links nach rechts:

- zum Beginn der Animation springen
- einen Schritt in der Animation zurück
- Play/Pause
- einen Schritt in der Animation weiter
- zum Ende der Animation springen

Wenn du eine genaue Erklärung der Struktogramm-Befehle suchst, kannst du die Tool-Tipps zu den einzelnen Befehlen aufrufen, indem du über den jeweiligen Struktogramm-Punkt fährst.

Step-By-Step (Tool-Tipps für jeden Struktogramm-Befehl) - 1. Suchen

- **suchen(head):** Die Funktion „suchen“ wird aufgerufen, ihr wird der „head“ der zu durchsuchenden Liste mitgegeben.
- Eine Variable, die die Kette durchlaufen soll bekommt das erste Kettenelement (head) zugewiesen.
- Die Schleife läuft so lange, bis das letzte Element der Kette erreicht ist, also bis zum Verweis auf NULL.
- Ist das aktuelle Kettenelement kaputt ...
- Ja: Das Kettenelement wurde gefunden, die Schleife kann beendet werden.
- Nein: Es passiert nichts, da das kaputte Kettenelement noch nicht gefunden wurde.
- Das nächste Kettenelement wird in der Variable „elementeDurchlaufen“ gespeichert.
- Wenn das Ende der Kette erreicht ist wird zurückgegeben, dass keines der Kettenelemente kaputt ist.

Step-By-Step (Tool-Tipps für jeden Struktogramm-Befehl) - 2. Einfügen

- **einfügen(vorherigesKettenelement, neuesKettenelement, head):** Die Funktion „einfügen“ wird aufgerufen. Ihr wird das neue Kettenelement mitgegeben, das Element an das dieses angehängt wird und der Kettenanfang.
- Soll das neue Element an den Anfang gesetzt werden (es gibt also kein vorheriges Element) ...
- Ja - 1. Schritt : Der aktuelle Kettenanfang wird an das neue Kettenelement gehängt.
- Ja - 2. Schritt : Das neue Kettenelement ist nun der Kettenanfang.
- Nein - 1. Schritt: Das Kettenelement, das hinter das neue Element soll wird gespeichert, damit es nicht verloren geht.
- Nein - 2. Schritt: Das neue Kettenelement wird an das „vorherigeKettenelement“ gehängt.
- Nein - 3. Schritt : Das Kettenelement das gespeichert wurde, wird an das neu eingefügte Element gehängt.
- Der (evtl. neue) Kettenanfang wird zurückgegeben.

Step-By-Step (Tool-Tipps für jeden Struktogramm-Befehl) - 3. Löschen

- **löschen(vorherigesKettenelement, kaputtesKettenelement, head):** Die Funktion „löschen“ wird aufgerufen. Ihr wird das Kettenelement mitgegeben, das gelöscht werden soll. Das Element an dem dieses angehängt ist und der Kettenanfang.
- Falls das zu löschende Kettenelement der Kettenanfang ist, also das „vorherigeKettenelement“ nicht existiert
- Ja: Das Kettenelement nach dem kaputten Element wird zum Kettenanfang.
- Nein: Das Kettenelement nach dem kaputten Element wird dem „vorherigenKettenelement“ angehängt, das kaputte wird somit nicht mehr „festgehalten“ und ist damit gelöscht.
- Der (evtl. neue) Kettenanfang wird zurückgegeben.

Hilfe-Button:

- Möchtest du mehr über Listen erfahren? [Ja](#) (**Link in das Lexikon Thema Listen: R006**)
- Oder nochmals nachlesen was Listen sind? [Ja](#) (**Link zur Einstiegsseite „Lineare Listen“: A200**)
- Wenn du wissen willst, wie die Anwendung bedient wird, kannst du unter dem Register „Bedienung“ im Aufgabenfenster alles nachlesen.

5.2.3 Rangierbahnhof (Experiment)

Für dieses Experiment mussten die benötigten Interaktionsmöglichkeiten neu erschaffen werden. Der grundlegende Aufbau der Seite bleibt auch hier erhalten. Die einzigen Änderungen sind wiederum die verschobenen Kontrollbuttons, und die zusätzliche Beschreibung des Interaktionskonzepts.

5.2.3.1 Experiment – Aufbau & Interaktion

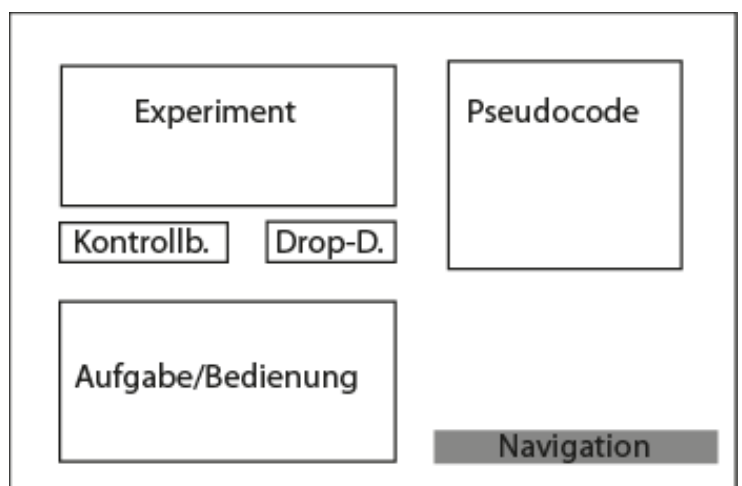


Abb. 31 grobe Aufteilung der Elemente des Experiments (eigene Skizzierung)

aktive Navigationselemente (im Bein bzw. auf dem Wäschezettel):

Lexikon-Button	führt zu R006 (Lexikon Listen-Eintrag)
Home-Button	Link in den Laborraum
M+I-Button	Link zum MI-eLearning Portal
Weiter-Pfeil	führt zu A203 (To-Do Liste, Struktogrammentwicklung)
Zurück-Pfeil	führt zu A201 (Ver-kette-t, Visualisierung)
Menü-Punkt Listen	führt zu A200 (Listen, Einstiegsseite)
Menü-Punkt Ver-kette-t	führt zu A201 (Ver-kette-t, Visualisierung)
Menü-Punkt To-Do Liste	führt zu A203 (To-Do Liste, Struktogrammentwicklung)

aktive Interaktionselemente:

- Kontrollbuttons: Ermöglicht das Zurücksetzen des Experiments auf den Start-/Ausgangspunkt
- Drop-Down: Wahl der verschiedenen Aufgaben; drei Unterpunkte: Löschen, Einfügen, Kombination (s. Unterpunkt 5.2.3.2 Experiment – Interaktionskonzept)
- Interaktion mit dem Experiment: verschiedene Drag&Drop Möglichkeiten (s. Unterpunkt 5.2.3.2 Experiment – Interaktionskonzept)
- Aufgabe/Bedienung: Möglichkeit zwischen der Aufgabenbeschreibung und der Beschreibung der Bedienung zu wechseln

5.2.3.2 Experiment – Interaktionskonzept

Die Idee einen Rangierbahnhof als Experiment zu entwerfen entstammte einem Vorschlag aus der Literatur (s. Kapitel 2.2.2 Verschiedene Anwendungen aus dem Alltag). Das Interaktionskonzept musste jedoch komplett neu entwickelt werden. Die „Beschreibung der Bedienung“ im Unterpunkt 5.2.3.4 erklärt bereits die geplante Umsetzung der Interaktion. Zum besseren Verständnis sollen hier diese Punkte, anhand visueller Beispiele, nochmals genauer erläutert werden.

Aufbau & Gestaltung

Der generelle Aufbau des Experiments lässt sich am besten grafisch darstellen (s. Abb. 32). In jeder Aufgabenstellung sind zwei Gleise (oben und unten bzw. ICE-Gleis und Abstellgleis) vorhanden, sowie zwei Hilfskräne (Hilfsvariablen, jeweils eine oben und eine unten). Einzig die Anzahl und Position der Wagons ist bei jeder Aufgabe unterschiedlich.

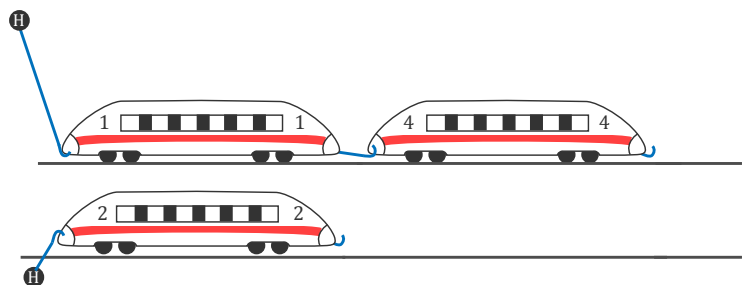


Abb. 32 Ansicht des Experiments (eigener Entwurf)

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Zusätzlich gibt es zwei Wagon-Arten. Kaputte Wagons werden grau und mit Dellen dargestellt. „Funktionierende“ Wagons hingegen gleichen einem realen ICE-Wagon. Alle Wagons besitzen eine identifizierende Nummer.

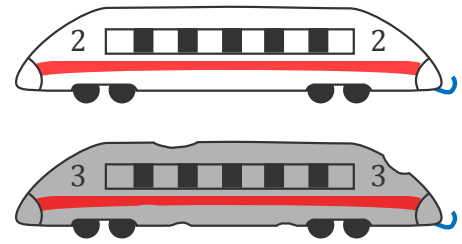


Abb. 33 Wagon-Arten (eigener Entwurf)

Interaktionsmöglichkeiten

Wie in der „Beschreibung der Bedienung“ (s. Kapitel 5.2.3.4 Experiment - Texte) erklärt, können die Verweise (Haken) am Hilfskran bzw. am Ende eines Wagons herausgezogen, und am Anfang eines anderen Wagons befestigt werden.

Wird ein Verweis (Haken) ergriffen, werden die möglichen „Ankerpunkte“ (am Anfang der anderen Wagons) visuell hervorgehoben (s. Abb. 34). Sobald ein Verweis einen Ankerpunkt erreicht, in dem er eingehängt werden kann, erfolgt ebenfalls eine visuelle Rückmeldung (der Kreis wird dunkler).

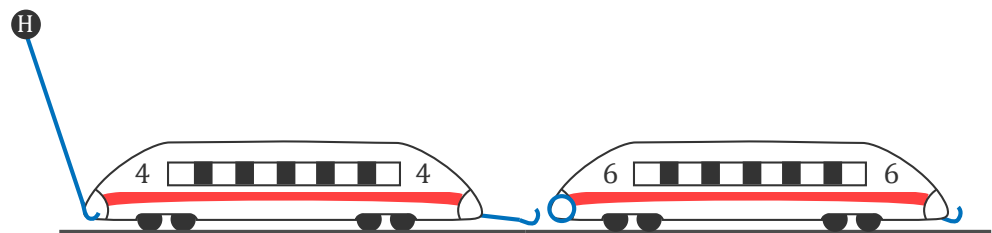


Abb. 34 Visuelle Rückmeldung, wenn ein Haken ergriffen wird (eigener Entwurf)

Die Reaktionen, die bei dem Umsetzen eines Hakens/Verweises hervorgerufen werden, sind an die Funktionen einer realen Liste angepasst:

Ein Wagon bleibt demnach nur sichtbar, wenn er an einem Haken an einer Hilfsvariable oder einem anderen Wagon festgemacht wurde. Sobald diese Verbindung nicht mehr existiert sind die entsprechenden Wagons gelöscht. In Abb. 35 z.B. verschwinden somit, wenn die Verbindung vom oberen Hilfskran (H) zum ersten Wagon gelöscht wird, der erste und vierte Wagon.

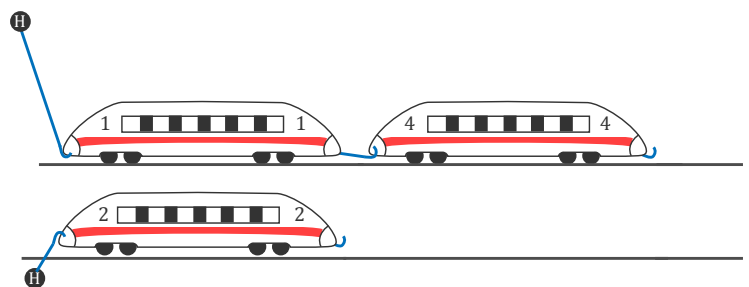


Abb. 35 Ansicht einer Experiment-Aufgabe (eigener Entwurf)

Um eine Verbindung zu löschen muss nur der entsprechende Verweis von dessen Ankerpunkt entfernt werden. Wenn z.B. der verbindende Haken von Wagon eins zu Wagon vier nach oben gezogen wird (außerhalb des vierten Wagons), so wird der Wagon vier gelöscht.

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Einen Wagon auf das andere Gleis zu stellen ist auch recht einfach. Hierzu muss nur die Verbindung zu einem Wagon des anderen Gleises, oder zum entsprechenden Hilfskran gezogen werden. Im Falle der Abb. 35 könnte z.B. der vierte Wagon an den zweiten oder an den unteren Hilfskran gehängt werden. Im ersten Fall wird der vierte Wagon hinter dem zweiten Wagon platziert werden. Im zweiten Fall hingegen wird der vierte Wagon die Position des zweiten einnehmen. Der zweite Wagon wird somit gelöscht.

5.2.3.3 Experiment – Aufgabenstellungen

Da drei verschiedene Aufgabenstellungen bearbeitet werden können, müssen verschiedene Anfangsszenarios möglich sein:

1. Aufgabe: Löschen

In der ersten Aufgabe muss ein Wagon (Nr. 3) gelöscht werden. Diese Aufgabe soll die Einfachheit des Löschens veranschaulichen und erfahrbar machen.

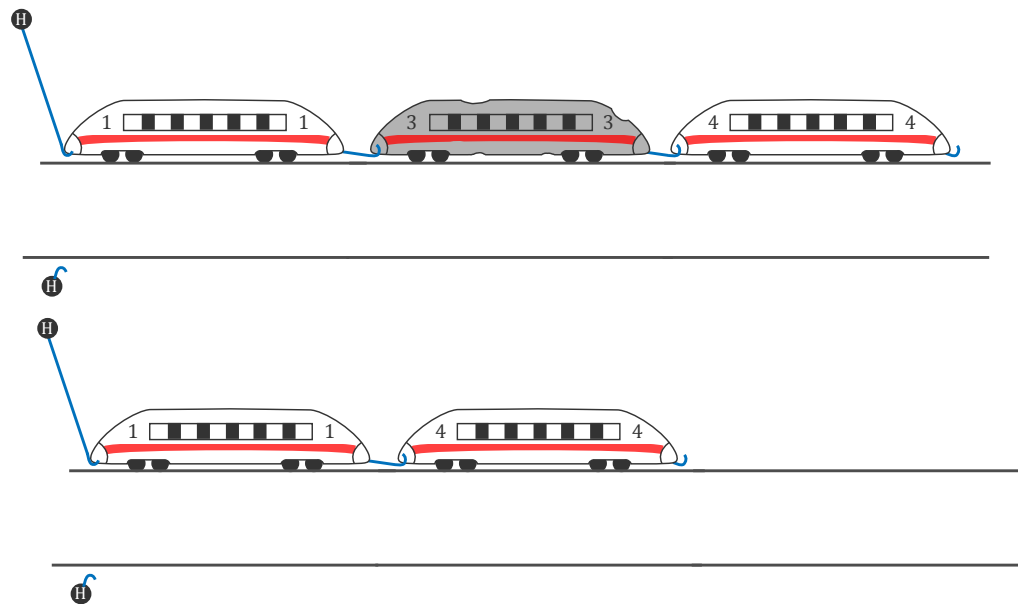


Abb. 36 Ansicht der Löschen-Aufgabe, Start (oben) & Lösung (eigener Entwurf)

Beim optimalen Vorgehen benötigt man nur einen Klick um einen Wagon zu löschen (Haken des 1. Wagens an den 4. Wagon hängen):

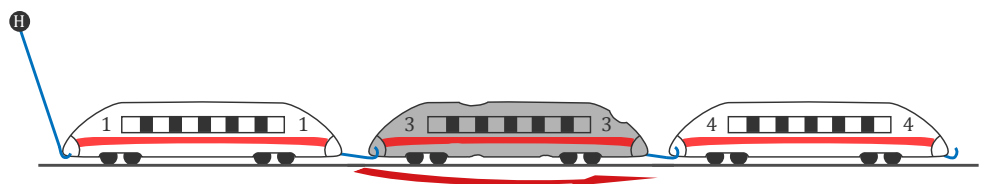


Abb. 37 Ansicht der Löschen-Aufgabe, Lösungsweg (eigener Entwurf)

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

2. Aufgabe: Einfügen

Die zweite Aufgabe ist etwas anspruchsvoller als die erste, es muss hier ein Wagon (Nr. 2) richtig in den ICE (Wagon 1 und 4) eingebaut werden.

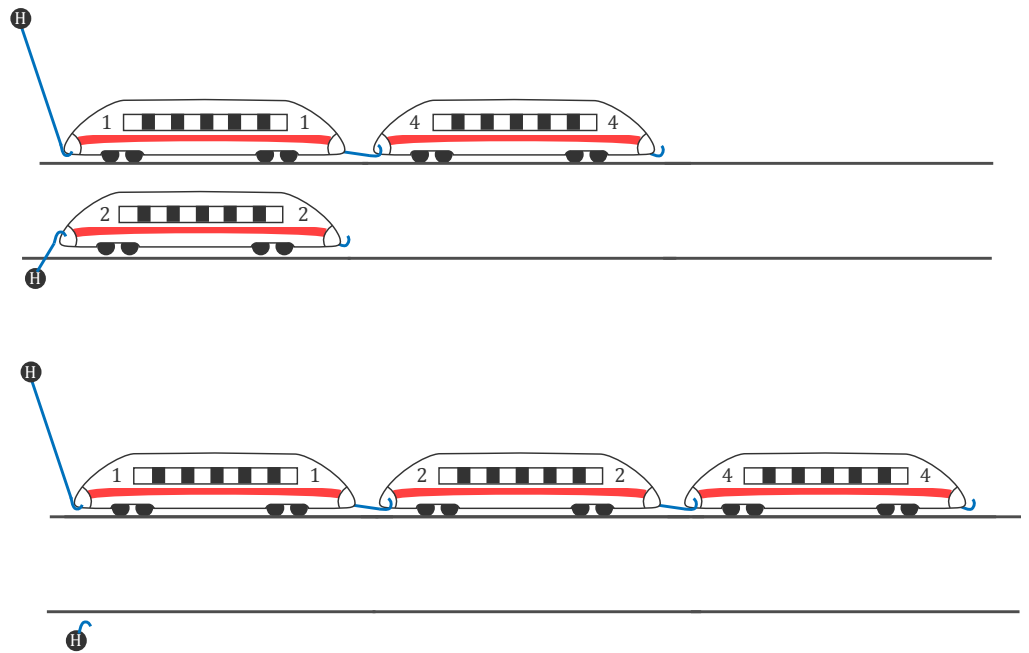


Abb. 38 Ansicht der Einfügen-Aufgabe, Start (oben) & Lösung (eigener Entwurf)

Wenn man das optimale Vorgehen verwendet benötigt man zwei Klicks, um einen Wagon einzubauen (Haken des 2. Wagens an den 4. und Haken des 1. Wagens an den 2.):

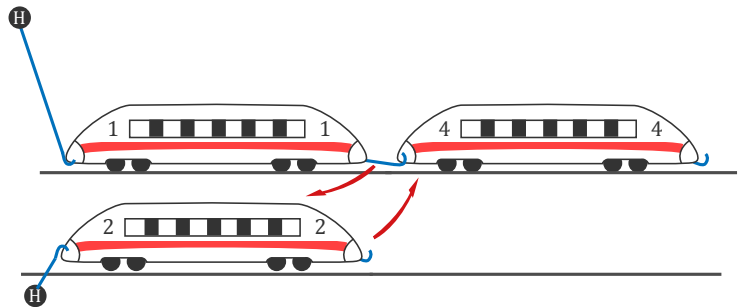


Abb. 39 Ansicht der Einfügen-Aufgabe, Lösungsweg (eigener Entwurf)

3. Aufgabe: Kombination

Die letzte Aufgabe verbindet und kombiniert die vorherigen Aufgaben. Hier muss ein Wagon (Nr. 3) gelöscht und ein anderer (Nr. 1) neu eingefügt werden.

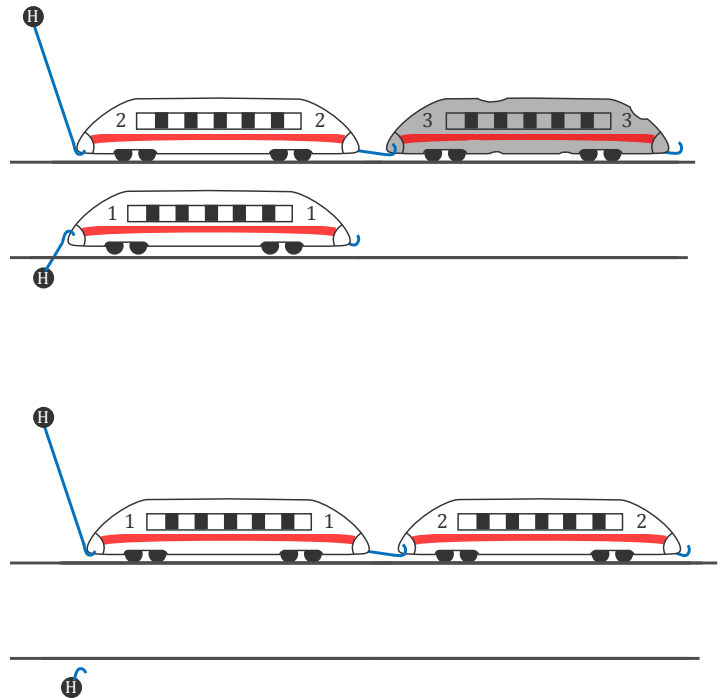


Abb. 40 Ansicht der Kombinations-Aufgabe, Start & Ende (eigener Entwurf)

Im optimalen Fall benötigt man hier drei Klicks um einen Wagon einzubauen und einen anderen zu löschen (Haken des 2. Wagens vom 3. lösen, Haken des 1. Wagens an den 2. Wagon und den Haken der oberen Hilfsvariable an den 1. Wagon).

5.2.3.4 Experiment – Texte

Aufgabenbeschreibung:

Die Züge, die in diesem Rangierbahnhof stehen, funktionieren wie eine Liste. Deine Aufgabe ist es die Ersatzwagen (unten) in den ICE (oben) einzubauen, oder einen defekten Wagon (grau und mit Dellen) auszubauen.

Die Zugteile müssen dabei in möglichst wenigen Schritten und in der richtigen Reihenfolge (aufsteigend nach den Wagen-Nummern sortiert) eingebaut werden, damit die Fahrgäste sich schnell zurechtfinden.

In drei Aufgabenstellungen kannst du selbst ausprobieren wie Listen funktionieren. Wenn du dabei den Algorithmus auf der rechten Seite anwendest, findest du immer die optimale Lösung.

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Der „vorherige“ Wagon ist dabei immer der mit der kleineren Zugnummer (als der neue oder zu löschende Wagon). Der „nächste“ Wagon der mit der größeren Zugnummer (als der neue oder zu löschende Wagon).

Beschreibung der Bedienung:

In diesem Experiment stehen Dir folgende Möglichkeiten zur Verfügung um die gestellten Aufgaben zu lösen:

Alle Haken (**[Bild Haken]**- jeweils am **[Bild Hilfsvariable]** oder dem Wagon-Ende befestigt) können per Drag&Drop und zu einem Wagon-Ankerpunkt (**[Bild Wagon vorne]**) gezogen werden. Die verfügbaren Ankerpunkte werden dir mittels eines blauen Kreises (**[Bild Wagon vorne mit Kreis]**) angezeigt, sobald du einen Haken ergreifst.

Diese Haken funktionieren wie Verweise in einer Liste. Sobald ein Wagon also von keinem Haken (am Anfang des Wagons) mehr gehalten wird, wird dieser gelöscht.

Wie in einer normalen linearen Liste kannst du auch hier die Verweise/Haken nach belieben umsetzen. Um einem Wagon an einen anderen Wagon oder **[Bild Hilfsvariable]** zu hängen muss der Haken des „vorherigen“ Wagons einfach in den Kreis des gewünschten „nächsten“ Wagons gezogen werden. Sobald der Haken eingehängt werden kann, wird der entsprechende Kreis dunkler.

Wenn du einen Fehler gemacht hast kannst du einfach mit dem ersten der fünf Buttons („Zum Anfang“) das Experiment von neuem starten.

Texte zur Erklärung (Pseudocode):

Lösungsidee:

Durchlaufe die obere Wagonkette solange, bis

entweder

„vorherige“Wagonnummer < Wagonnummer Abstellgleis
und Wagonnummer Abstellgleis < „nächste“Wagonnummer
dann den Wagon einfügen:

1. „neuer“Wagon.next := „nächster“Wagon
2. „vorheriger“Wagon.next := „neuer“Wagon

oder

der nächste Wagon kaputt ist
dann den Wagon löschen:

1. „vorheriger“Wagon.next := „kaputter“Wagon.next:

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

MIA, wenn die beste Lösung gefunden wurde:

Super, das ist die richtige Lösung!

MIA, wenn nicht die beste Lösung gefunden wurde:

Das war schon gut, aber es geht auch mit weniger Schritten. Probier`s doch nochmal!

MIA, wenn ein Wagon zuviel gelöscht wurde und die Aufgabe nicht mehr gelöst werden kann:

Jetzt hast du einen Zug gelöscht, den du noch brauchst. Probier`s am besten gleich nochmal von vorn!

MIA, wenn die Wagons richtig angeordnet sind, aber noch auf dem unteren Gleis stehen:

Du hast es fast geschafft, der ICE muss nur noch aufs andere Gleis.

Hilfe-Button:

- Möchtest du mehr über Listen erfahren? [Ja](#) (**Link in das Lexikon – Thema Listen: R006**)
- Oder nochmal nachschauen wie man Elemente löscht und einfügt? [Ja](#) (**Link zu „Ver-Kette-t“: A201**)
- Wenn du wissen willst, wie die Anwendung bedient wird, kannst du unter dem Register „Bedienung“ im Aufgabenfenster alles nachlesen.

5.2.4 To-Do Liste (Struktogrammentwicklung)

In der Struktogrammentwicklung wurden einige Änderungen und Erweiterungen entworfen, die im Kapitel 4.1.2 „Verschiedene Interaktionsmöglichkeiten“ nachgelesen werden können. Der Grundgedanke bleibt auch hier bestehen; ein Struktogramm soll entwickelt und in einer passenden Animation veranschaulicht werden. Diese Animation ermöglicht es Fehler in einem noch nicht funktionierenden Struktogramm zu erkennen.

5.2.4.1 Struktogrammentwicklung – Aufbau & Interaktion

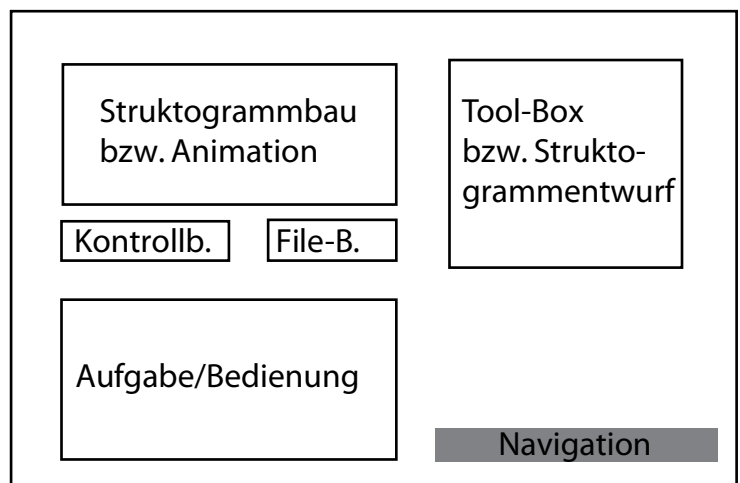


Abb. 41 grobe Aufteilung der Elemente im Struktogrammentwurf (eigene Skizzierung)

aktive Navigationselemente (im Bein bzw. auf dem Wäschezettel):

Lexikon-Button	führt zu R006 (Listen-Eintrag)
Home-Button	ausblenden der Lektion / Laborraum anzeigen
M+I-Button	führt zu MI-eLearning Portal
Zurück-Pfeil	führt zu A202 (Rangierbahnhof, Experiment)
Menü-Punkt Listen	führt zu A200 (Listen, Einstiegsseite)
Menü-Punkt Ver-kette-t	führt zu A201 (Ver-kette-t, Visualisierung)
Menü-Punkt Rangierbahnhof	führt zu A202 (Rangierbahnhof, Experiment)

aktive Interaktionselemente:

- File-Buttons: Speichern, Öffnen, Löschen und Prüfen eines entworfenen Struktogramms (Funktion wie in den bereits implementierten Lektionen)
- Tool-Box: Bau des Struktogramms (Funktion wie in den bereits implementierten Lektionen unter Berücksichtigung der Änderungen aus Kapitel 4.1.2). Die Struktur-Blöcke sind hierbei alle auswählbar (die Schleife ist nur einmal verwendbar). Die Operatoren werden wie in den bestehenden Lektionen übernommen und sind alle verwendbar. Funktionen sind keine vorhanden. Als Variablen werden „fälligkeit“, „HEAD“, „durchlaufen“, „merken“, „next“ und „neuesToDo“ zur Verfügung gestellt.
- Methodenaufruf im oberen/linken Fenster: kann verschoben werden, befindet sich zu Beginn in der Mitte der Anwendung. Das Struktogramm weitet sich bei Bedarf zentriert aus, es bleibt also immer mittig, wenn es nicht verschoben wurde.
- Animation - Drop-Down: Zwei Auswahlpunkte mit denen das Szenario der Animation bestimmt werden: „ein neues To-Do“ oder „zwei neue To-DOS“.
- Aufgabe/Bedienung: Möglichkeit zwischen der Aufgabenbeschreibung und der Beschreibung der Bedienung zu wechseln

5.2.4.2 Struktogrammentwicklung - Aufgabenstellung & Interaktion

Wie in der Aufgabenstellung (s. 5.2.4.4 Struktogrammentwicklung - Texte) erklärt, soll hier ein Struktogramm zum Füllen einer To-Do Liste erstellt werden. In einer bereits bestehenden Liste sind das erste Element (Fälligkeit = 1) sowie einige weitere Elemente (aufsteigend nach der Fälligkeit sortiert) bereits eingefügt. Das Struktogramm soll so entworfen werden, dass neue Elemente mit einer Fälligkeit größer 1, richtig in die Liste eingefügt werden.

Ein fehlerhaftes Struktogramm, das die Elemente nach dem neu einzufügendem versehentlich löscht, kann ebenfalls getestet werden.

Die Interaktion um das Struktogramm zu entwerfen entspricht den bereits in den bisherigen Lektionen umgesetzten Konzepten. Hinzu kommen nur die bereits im Kapitel 4.1.2 vorgestellten Änderungen. Da das eigentliche Konzept der Interaktion nicht verändert wurde, soll dieses hier nicht nochmals erklärt werden.

5.2.4.3 Struktogrammentwicklung – Struktogramm-Lösung

Folgende Struktogramme können in der Anwendung entworfen und anschließend getestet werden:

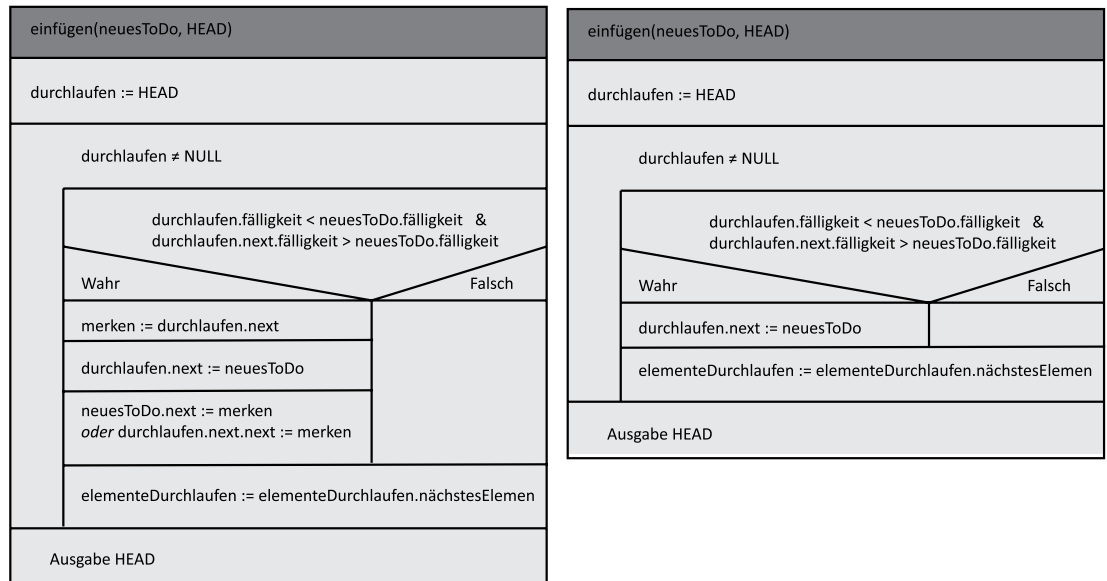


Abb. 42 Lösungs-Struktogramm, Struktogramm rechts mit testbarem Fehler (eigener Entwurf)

5.2.4.4 Struktogrammentwicklung – Visualisierung des Struktogramms

Das entwickelte Struktogramm kann in einer Visualisierung „getestet“ werden. Die Visualisierung zeigt den korrekten Ablauf des Struktogramms sobald das korrekte Struktogramm entwickelt wurde. Die Visualisierung des testbaren Fehlers wird gezeigt, wenn ein fehlerhaftes aber testbares Struktogramm (s. Unterpunkt 5.2.4.3 Struktogrammentwicklung – Struktogramm-Lösung) entworfen wurde.

Interaktionskonzept

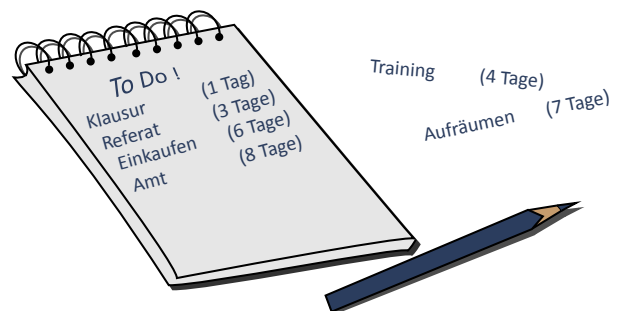
Die Interaktion funktioniert wie in einer Visualisierung (s. z.B. 5.2.2.2 Visualisierung – Ver-Kette-t Animation). Im Drop-Down können verschiedene Szenarien eingestellt werden. Die Animation wird auch hier mittels der Kontrollbuttons gesteuert, und das entworfene Struktogramm entsprechend durchlaufen.

Animationen

To-Do mit Fälligkeit 4 einfügen

Die ersten Schritte des Struktogramms finden mit dieser Ansicht statt (Ansicht 1). Sobald der HEAD zugewiesen wird, bekommt der erste Punkt „Klausur“ die Farbe rot. Die if-Abfrage läuft zunächst über „Falsch“.

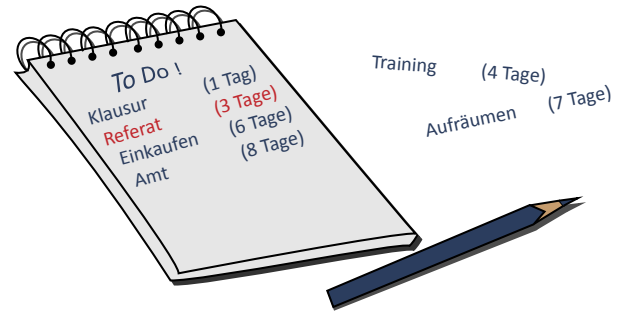
Das nächste Element wird ausgewählt und bekommt somit eine rote Schrift, der erste



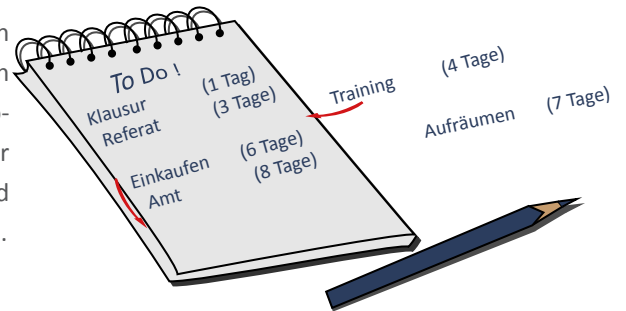
5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

Punkt ist nicht mehr aktuell und erhält wieder seine alte Farbe (Ansicht 2).

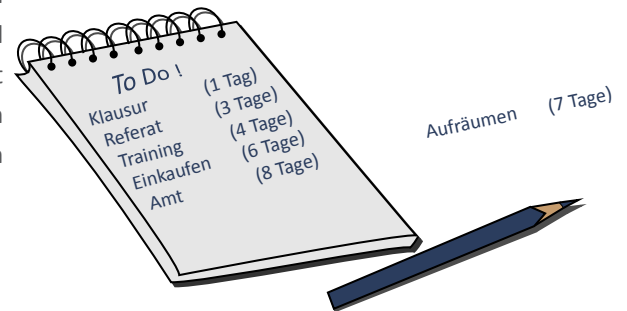
Am Ende jedes Durchlaufens der if-Abfrage wird das nächste Element rot gefärbt, und das vorherige wieder auf blau zurückgesetzt. Nur beim letzten Aufruf, sobald NULL zugewiesen wird, sind alle Punkte wieder blau.



Wenn ein Element eingefügt wird, z.B. nach dem Punkt „Referat“ (Ansicht 3) werden beim Durchlaufen des ersten „Wahr“-Struktogramm-Befehls die unteren Elemente weiter nach unten geschoben. Im zweiten Punkt wird das neue Element dann in die Liste integriert.



Die Ansicht 4 zeigt die To-Do Liste, nachdem der erste neue Punkt eingefügt wurde. Und somit auch die Endansicht, falls „To-Do mit Fälligkeit 4 einfügen“ gewählt wurde. Wenn das richtige Struktogramm entwickelt worden ist, sind alle Punkte vorhanden (Ansicht 4).



Wurde das „falsche“ Struktogramm entwickelt schiebt sich der neue Punkt, sobald „Wahr“ eingetreten ist, an die richtige Stelle und verdrängt das Element, das vorher an dieser Stelle war. Dieses Element und alle weiteren (im hier gezeigten Fall Einkaufen und Amt) fallen dann nach unten aus dem Bild und sind damit gelöscht. Die Schleife ist beendet, denn es gibt keine nächsten Elemente mehr.

To-Do mit Fälligkeit 4 und 7 einfügen

Beim Einfügen eines weiteren To-Do werden die Schritte des ersten Ablaufs wiederholt. Dann wird das Struktogramm ein zweites Mal mit dem nächsten To-Do gestartet.

Wenn das „falsche“ Struktogramm entwickelt wurde, wird „Aufräumen“ direkt an das „Training“ gehängt und dann der Durchlauf der Liste beendet. Wurde es korrekt entwickelt wird es, wie hier (Ansicht 5) gezeigt, an der vorletzten Position eingefügt.



Abb. 43 Ansichten der Visualisierung der To-Do Liste (eigener Entwurf)

5.2.4.5 Struktogrammentwicklung – Texte

Aufgabenbeschreibung Struktogramm:

Paul hat sich in der nächsten Woche ziemlich viel vorgenommen. Um einen Überblick zu bekommen, was er an welchem Tag fertig haben muss, will er eine To-Do Liste schreiben. In dieser sollen die Fälligkeiten aufsteigend sortiert werden. Ein paar wichtige Punkte hat er schon in die **Liste** geschrieben. Doch wie bekommt er die anderen Punkte noch zwischen die, die er bereits aufgeschrieben hat?

Helfe Paul, indem du ihm ein Struktogramm entwirfst, das einen neuen Listeneintrag („**neuesToDo**“) an die richtige Stelle der bestehenden Liste (gegeben durch „**head**“) einfügt. Die richtige Stelle ist hierbei, wenn die Fälligkeit des vorhergehenden Elements z.B. 1 Tag beträgt, die des nächsten Elements 4 Tage und du ein Element mit 3 Tagen Fälligkeit („**fälligkeit=3**“) einfügen möchtest .

Beachte hierbei, dass du nicht weißt, an welche Stelle das neue Element kommen wird, nur das erste Element steht schon fest da es die Fälligkeit 1 hat. Entwerfe das Struktogramm so, dass alle möglichen neuen Einträge richtig eingefügt werden. Alles was du dazu brauchst findest du in der Tool-Box oben rechts. Wenn du nicht weißt wie du die Tool-Box verwenden sollst, schaue unter „Bedienung“ nach.

Beschreibung der Struktogramm-Bedienung:

Um das Struktogramm zu bauen, musst du zuerst einen Struktogramm-Block in den bereits vorhandenen Methodenaufruf (oben im großen Fenster) einfügen. Sobald du einen Struktogramm-Block per Drag&Drop zum Methodenaufruf ziehst, werden dir die Möglichkeiten angezeigt, wo dieser platziert werden kann.

Der zweite Schritt besteht darin, die Struktogramm-Blöcke mit den entsprechenden Befehlen zu füllen. Klicke hierzu einen Struktogramm-Block im großen Fenster an, dann kannst du oben rechts mit Hilfe der dort sichtbaren Elemente deine Anweisung zusammenstellen.

Die Buttons für den Struktogramm-Entwurf:

- | | |
|----------------|---|
| OK: | Übernimmt die Änderungen in das Struktogramm. |
| Block-Löschen: | Die Änderungen in dem Block werden verworfen.
Wenn der Block leer ist wird dieser ganz gelöscht. |
| Speichern: | Speichert dein Struktogramm. |
| Öffnen: | Öffnet ein gespeichertes Struktogramm. |
| Check: | Prüft das Struktogramm auf Fehler. |
| Ausführen: | Dein Struktogramm kann von dir hier animiert werden. |

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

MIA-Rückmeldungen für die Struktogrammentwicklung - Syntaxfehler (wie in bereits vorhandenen Lektionen umgesetzt):

SYNTAXFEHLER	REAKTION
Unerwartete Zahl	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Zahlen.“ Stufe 2: „Im markierten Block ist eine Zahl an einer nicht erlaubten Position.“
Unerwarteter Operator	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Operatoren.“ Stufe 2: „Im markierten Block ist ein Operator an einer nicht erlaubten Position.“
Unerwartete Funktion	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Funktionen.“ Stufe 2: „Im markierten Block ist eine Funktion an einer nicht erlaubten Position“
Unerwartete (-Klammer	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Klammersetzung.“ Stufe 2: „Im markierten Block taucht eine unerwartete (auf.“
Unerwartete)-Klammer	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Klammersetzung.“ Stufe 2: „Im markierten Block taucht eine unerwartete) auf.“
Unerwartetes Komma	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Kommasetzung.“ Stufe 2: „Im markierten Block taucht ein unerwartetes Komma auf.“
Unerwartete Variable	Stufe 1: „Syntaxfehler: Prüfe im markierten Block deine Variablen.“ Stufe 2: „Im markierten Block hast du eine falsche Variable verwendet.“
Unbekannte Zeichen	Stufe 1: „Syntaxfehler: Prüfe die Zeichen im markierten Block.“ Stufe2: „Syntaxfehler: Im markierten Block existieren unbekannte Zeichen“
Variablen und Operatoren passen nicht zueinander	Stufe 1: „Syntaxfehler: Prüfe im markierten Block die Variablen und Operatoren“ Stufe 2: „Syntaxfehler: Im markierten Block passen die Variablen mit den Operatoren nicht zusammen.“

Die Prüfung auf Fehler findet jeweils statt, wenn der „Check“-Button geklickt wurde.

5 LEKTION „LISTEN“ - IDEE UND KONZEPTION

andere MIA-Rückmeldungen für die Struktogrammentwicklung:

FEHLER	REAKTION
Genau ein if-Block	Stufe 1: „Du brauchst genau eine Unterscheidung, ob das Element hier eingefügt wird oder nicht.“ Stufe 2: „Es darf nur dann das Element eingefügt werden, wenn eine Fälligkeit kleiner und die nächste größer ist als die des neuen To-Dos. Nutze also eine Verzweigung!“ Stufe 3: MIA gibt den korrekten Aufruf vor.
Inhalt der if-Abfrage	Stufe1: „Es darf nur dann das Element eingefügt werden, wenn eine Fälligkeit kleiner und die nächste größer ist als die des neuen To-Dos.“ Stufe2: „Die Verzweigung muss prüfen, ob die Fälligkeit des neuen To-Dos zwischen die von „durchlaufen“ und „durchlaufen.next“ passt.“ Stufe 3: MIA gibt den korrekten Aufruf vor.
Genau eine Schleife bzw. -Abfrage	Stufe 1: „Du musst die Liste genau einmal ganz durchlaufen.“ Stufe 2: „Die Liste muss einmal durchlaufen werden, also bis der Verweis auf das nächste Element „NULL“ ist. Nutze also eine Schleife!“ Stufe 3: MIA gibt den korrekten Aufruf vor.
Fehlendes „Hochzählen“ mit .next	Stufe 1: „Du musst die Liste durchlaufen, also immer von einem To-Do zum nächsten gehen.“ Stufe2: „Gebe dem „durchlaufen“ ein neues To-Do-Element indem du ihm das nächste To-Do zuweist.“ Stufe 3: MIA gibt den korrekten Aufruf vor.
Der HEAD wird keinem anderen Element zugewiesen	Stufe 1: „Um die To-Do-Liste zu durchlaufen solltest du den HEAD nicht direkt verwenden.“ Stufe 2: „Weise „durchlaufen“ den HEAD zu um damit die Liste zu durchlaufen.“ Stufe 3: MIA gibt den korrekten Aufruf vor.
Fehlende „Wahr“-Anweisungen	Stufe 1: „Wenn du die richtige Stelle für das neue To-Do gefunden hast muss es eingefügt werden.“ Stufe 2: „Füge das To-Do ein, indem du den jetzigen Nachfolger merkst, das neue To-Do einfügst und dann das gemerkte an das neue Element fügen.“ Stufe 3: MIA gibt den korrekten Aufruf vor.
testbarer Fehler	„Du wirst falsche Lösungen erhalten, kannst das Programm aber trotzdem ausführen um den Fehler zu erkennen.“

Die Prüfung auf Fehler findet jeweils satt, wenn der „Check“-Button geklickt wurde.

Aufgabenbeschreibung Animation des Struktogramms:

Paul hat sich in der nächsten Woche ziemlich viel vorgenommen. Um einen Überblick zu bekommen, was er an welchem Tag fertig haben muss, will er eine To-Do Liste schreiben. Ein paar wichtige Punkte hat er schon in die Liste eingetragen. Doch wie bekommt er die anderen Punkte noch dazwischen?

Für diese Aufgabe hast du ein Struktogramm entworfen. Nun kannst du es animieren und sehen ob es funktioniert, oder was du noch verbessern kannst.

Beschreibung der Animation-Bedienung:

Im Drop-Down Menü kannst du einstellen, welches neue To-Do du einfügen möchtest. Entweder ein To-Do mit Fälligkeit 4 oder 7 ... probiere einfach aus ob dein Struktogramm alle To-Dos richtig einfügt.

Gestartet wird die Anwendung über die Kontrollbuttons (Buttons unter der Animation). Sie bieten folgende Möglichkeiten, von links nach rechts:

- um Beginn der Animation springen
- einen Schritt in der Animation zurück
- Play/Pause
- einen Schritt in der Animation weiter
- zum Ende der Animation springen

Über „Editieren“ kannst du dein Struktogramm nochmal ändern.

Animation - MIA, wenn die beste Lösung gefunden wurde:

Wow, du hast den Algorithmus prima entwickelt!

Animation - MIA sonst:

Hast du deinen Fehler gefunden? Du kannst über den „Editieren“-Button dein Struktogramm nochmal ändern.

Hilfe-Button:

- Möchtest du mehr über Listen erfahren? [Ja](#) (**Link in das Lexikon – Thema Listen: R006**)
- Oder nochmal nachschauen wie man Elemente löscht und einfügt? [Ja](#) (**Link zu „Ver-Kette-t“: A201**)
- Wenn du wissen willst, wie die Anwendung bedient wird, kannst du unter dem Register „Bedienung“ im Aufgabenfenster alles nachlesen.

5.2.5 Lexikoneintrag

Im Lexikon werden verschiedene Anwendungsbeispiele und der Aufbau, bzw. wichtige Begriffe, einer Liste genauer erklärt und veranschaulicht. Das Layout und die Interaktionsmöglichkeiten wurden hierbei exakt der Vorlage bereits implementierter Seitenansichten übernommen.

5.2.5.1 Lexikoneintrag – Aufbau & Interaktion

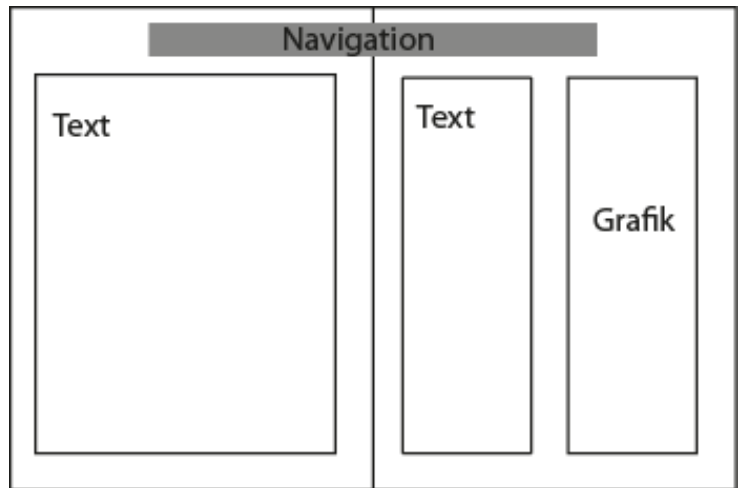


Abb. 44 grobe Aufteilung der Elemente im Lexikon-Eintrag (eigene Skizzierung)

aktive Navigationselemente:

Lexikon-spezifische Navigation durch den gesamten Inhalt des Lexikons und zurück zur vorher besuchten Seitenansicht (außerhalb des Lexikons).

aktive Interaktionselemente:

Es sind keine Interaktionselemente vorhanden.

5.2.5.2 Lexikoneintrag – Visualisierung

Um die Erklärung der wichtigen Elemente einer Liste zu vereinfachen und anschaulich darzustellen, werden diese anhand der in der Visualisierungs-Anwendung verwendeten Kette aufgezeigt.

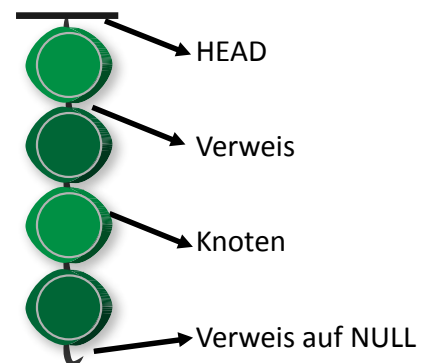


Abb. 45 Darstellung zur Erklärung der Listen-Elemente (eigener Entwurf)

5.2.5.3 Lexikoneintrag – Texte

Listen

Die erste Datenstruktur, die Lernenden in der Informatik vorgestellt wird ist der „Array“ bzw. das „Feld“. In einem Array kann eine zuvor festgelegte begrenzte Anzahl an Elementen gespeichert, und über den Index des Speicherortes wieder erreicht werden.

„Listen“ dagegen sind eine dynamische Datenstruktur. Sie können also zur Laufzeit des Programms beliebig vergrößert (neue Elemente speichern), oder verkleinert (Elemente löschen) werden und passen sich so dem jeweiligen Speicherbedarf optimal an.

Dem entsprechend werden Listen in der Praxis vor allem angewendet, wenn eine hohe Flexibilität benötigt wird. Also wenn nur wenige Elemente gespeichert, gelöscht und oft neue eingefügt werden sollen. Auch bei einer topologischen Sortierung, also wenn Dinge aufeinander aufbauen, werden Listen gerne verwendet. Z.B: wenn eine Aufgabe abgeschlossen sein muss bevor die nächste gestartet werden kann und man dann nicht mehr zur ersten Aufgabe zurückkehren kann.

Anhand der verketteten linearen Liste kann das Grundverständnis zur Funktion und Umgangsweise mit allen Listenarten erklärt werden. Auch die Grundlagen anderer Datenstrukturen wie den Bäumen oder der Schlange bauen auf diesen Prinzipien auf. ([Link zum jeweiligen Kapitel](#))

Aufbau und Funktion einer Liste:

Die verkettete Liste besteht aus mehreren Elementen („Knoten“), die miteinander verkettet sind. Jeder Knoten besteht aus seinen spezifischen Daten und einer Referenz bzw. einem Verweis auf ein Folgeelement. Das letzte Element der Liste wird durch den Verweis auf „NULL“ (kein Element) gekennzeichnet.

[Bild der Kette (Animation) mit Beschriftung der einzelnen Elemente]

Um auf die komplette Liste zugreifen zu können, muss demnach nur das erste Element der Liste durch einem Anker-Verweis („Head“ oder „Kopf“) referenziert werden. Alle anderen Elemente können über das Navigieren über die Verweise, also das von vorne durchlaufen der Liste, erreicht werden.

Wenn z.B. das zweite Element der Liste aufgerufen werden soll, lautet die Anweisung für die Listennavigation „Gehe an den Anfang (Head) der Liste, dann gehe zum nächsten Element“. Ähnlich einer Einbahnstraße kann bei einer linearen Liste jedoch nicht zum vorherigen Element zurückgekehrt werden.

5.2.6 Tool-Tip Texte

Liste

Eine **Liste** startet immer mit dem „Head“ der auf das erste Element („Knoten“) verweist. Das erste Element verweist dann auf das zweite, das zweite auf das dritte und so weiter. Nur das letzte Element verweist auf kein Element („NULL“).

Knoten

Ein **Knoten** ist ein Objekt bzw. Element in einer Liste. Er besitzt immer die Daten des Elements und einen Verweis auf den nächsten Knoten.

Verweis

Ein **Verweis** verknüpft zwei Elemente bzw. Objekte einer Liste miteinander. Um von einem Element zum nächsten zu kommen, muss man also über den Verweis zu dem nächsten Element gehen. Wenn der Verweis auf „NULL“ zeigt bist du am letzten Element der Liste angekommen.

6 LEKTION „LISTEN“ - UMSETZUNG

Nachdem die Konzeption der Listen Lektion abgeschlossen war, konnte die eigentliche Implementierung der Inhalte beginnen. Diese Lektion wurde, wie die bereits vorhandenen Elemente des VIL, mittels Flash (CS5) und Actionscript 3 umgesetzt.

Die hier beschriebene Umsetzung der neuen Lektionsinhalte sowie das Einbinden in das bestehende VIL führte ich selbstständig durch. Beim Online-stellen der Inhalte unterstützte mich Carolina Bernal (Assistentin der Hochschule), wie im entsprechenden Kapitel (6.7 Online-stellen der neuen Inhalte) beschrieben. Als Hilfsmittel verwendete ich wiederum ein Flash-Handbuch (WESCHKALNIES (2010)).

Im Folgenden soll nun eine Zusammenfassung der einzelnen Funktionen und das Zusammenspiel der verschiedenen Elemente der neuen Lektion gegeben werden (komplette Umsetzung s. praktische Arbeit „virtlab-Juli2012“: `fla/lineareListen/` und `src/fho/lineareListen/`). Für das bisher implementierte VIL ist keine Dokumentation vorhanden. Aus diesem Grund habe ich die im bisherigen VIL durchgeführten Schritte genau beschrieben.

6.1 Einbinden der neuen Lektion

Vorgehen

Das Einbinden einer neuen Lektion konnte auf den vorhandenen bereits implementierten Lektionen aufgebaut werden. Um eine Lektion einzubinden sind nur wenige Ergänzungen bzw. Anpassungen nötig:

Die eigentliche Flash-Datei (s. praktische Arbeit „virtlab-Juli2012“: `fla/virtlab/virtlab.fla`) musste nur geringfügig angepasst werden. Hier genügt es den zur Lektion gehörenden Button so umzugestalten dass er ein aktiver Button ist.

Hierzu geht man zunächst in den MovieClip „mcRoom“ und wählt den gewünschten Button an. Dieser Button hat, wenn die Lektion noch nicht implementiert ist, eine Standardbezeichnung, z.B. „mclcon02“. Dieser Instanzname sollte zunächst in den gewünschten Namen umgeändert werden, in diesem Fall „mclListen“.

Im zweiten Schritt sollte die Beschriftung des Buttons (Tool-Tipp) angepasst werden. Im entsprechenden MovieClip, hier der „mclListen“, kann im zweiten Bild dieser Text von „Diese Anwendung wird noch entwickelt“ in den eigentlichen Namen der Lektion, hier „Listen“, umbenannt werden. Weitere grafische Änderungen müssen im Raum nicht vorgenommen werden.

Zu der Flash-Datei müssen noch folgenden ActionScript-Dateien (s. praktische Arbeit „virtlab-Juli2012“: `src/fho/virtlab`) angepasst werden um eine Lektion einzubinden:

In der „FRoom.as“:

```
Hinzufügen des neu angelegte Icon (mclListen) in Zeile 12:  
public var mcMI:Flcon, mcLexikon:Flcon, mcMatroschkas:Flcon, mclListen:Flcon;
```

6 LEKTION „LISTEN“ - UMSETZUNG

Unter „public function FRoom()“ muss folgender Befehl eingefügt werden:

```
mclisten.addEventListener(MouseEvent.CLICK, function(e:MouseEvent):void{
    FVirtlab_.appLineareListen.showApp();
});
```

Die alte Bezeichnung des Icons (aus der Flash-Datei, hier „mclcon02“) muss jetzt noch aus der Liste der inaktiven Icons entfernt werden (jeweils ein Eintrag in Zeile 14ff und 44ff).

In der „FExtra.as“:

Hier müssen nach dem Vorbild der Rekursion folgende Zeilen (jeweils an die Lektion angepasst) eingefügt werden. Die URL beinhaltet hierbei die Startseite der Lektion (mit der Navigation) im entsprechenden Unterordner:

```
public static const LINEARELISTEN_URL : String = „LineareListen/beginLineareListen.swf“;
public static const LINEARELISTEN_X : int = 0;
public static const LINEARELISTEN_Y : int = 0;
```

In der „Fvirtlab.as“:

In dieser Klasse der Laborraum-Funktionen muss eine neue Variable für die jeweilige Lektion in der Klassen-Definition angelegt werden:

```
public var appLineareListen:FAppLineareListen;
```

Diese Variable wird dann der init() Methode der Bühne hinzugefügt:

```
appLineareListen = new FAppLineareListen();
pAppHolder.addChild(appLineareListen);
```

Im letzten Schritt der Lektionseinbindung muss noch eine eigene Klasse für die Lektion angelegt werden. Dies ist, wie in den vorangegangenen Code-Fragmenten bereits gezeigt, im Fall der Listen Lektion die Klasse „FAppLineareListen“. Diese Klasse kann von den bereits implementierten Inhalten übernommen werden. Lediglich die Funktions- und Variablennamen werden entsprechend der gewünschten Lektion geändert:

```
...
public class FAppLineareListen extends FApp{
    public function FAppLineareListen() {
        url = FExtra.LINEARELISTEN_URL;
        var basePath:String = FVirtlab.getInstance().basePath;
        url = FCommon.getAbsPath(basePath, url);
        trace(„LineareListen url = „ + url);
        X = FExtra.LINEARELISTEN_X;
        Y = FExtra.LINEARELISTEN_Y;
    }
    override public function loaded(event : Event) : void {
        super.loaded(event);
    }
}
```

```
        with(MovieClip(_l.content)){
            setCallBackToHome(hideApp_);
            handleLexicon(FVirtlab.getInstance().appLexicon.showPage);
        }
    }
    ...
}
```

Ein besonderes Augenmerk soll hier noch kurz auf die zweite Funktion dieser Klasse gelegt werden. Durch die Aufrufe der Funktionen „setCallBackToHome()“ und „handleLexicon()“ werden diese Funktionen innerhalb der jeweiligen Lektionen gesetzt. Erst diese Aufrufe ermöglichen die Funktion des „Home-Button“ (zurück in den Laborraum) und den Aufruf des Lexikons, aus den Lektionsseiten heraus.

Probleme

Die Vorgehensweise bei der Einbindung einer neuen Lektion ist an für sich denkbar einfach. Doch können auch hier unerwartete Schwierigkeiten auftreten. In meiner Implementierung wurde die entsprechende Lektion (hier die Listen Lektion) aufgerufen und konnte auch bedient werden. Lediglich die Home-Funktion und die Links ins Lexikon funktionierten nicht.

Mir war schnell klar, dass dies aus der fehlenden Übergabe der Funktionen „SetCallBackToHome“ und „HandleLexicon“ herrühren musste. Auch die Flash-Fehlermeldung, dass die Datei (Listen-Lektion) nicht richtig geladen wurde, sprach dafür. Trotz dieses Hinweises von Flash wurde die Lektion jedoch geladen.

Nach langer Fehler-Suche im Actionscript-Code wurde ich schließlich an einer unerwarteten Stelle fündig: Beim Laden der Lektion wurde die Fehlermeldung erzeugt, da in den Exporteinstellungen (Datei → ActionScript Einstellungen) die Standardverknüpfung für die „Runtime Shared Library“ nicht „Im Code zusammengeführt“ wurde.

Durch diese fehlerhafte Einstellung wird eine externe Datei („textLayout_swz“) neben der eigentlichen SWF-Datei erstellt. Diese Datei wird nun, wenn die SWF-Datei aufgerufen wird, noch vor dieser geladen. Hierdurch kann die aufrufende Anwendung (hier der Laborraum) die Variablen (z.B. die Funktionen für Home und das Lexikon) nicht an die dafür vorgesehene Stelle in der eigentlichen SWF-Datei setzen. Diese SWF-Datei wird erst zu einem späteren Zeitpunkt geladen, somit entsteht die Fehlermeldung, obwohl die Datei geladen wird.

6.2 Einstiegsseite Listen

Vorgehen

Um eine neue Einstiegsseite anzulegen verwendet man am Besten eine bereits vorhandene Einstiegsseite einer anderen Lektion. In dieser Einstiegsseiten-Flash-Datei sind alle benötigten Grafiken (stehende MIA und ihre Sprechblase) sowie Layouteinstellungen bereits vorhanden. Somit muss man hier lediglich die Texte verändern.

6 LEKTION „LISTEN“ - UMSETZUNG

Soll eine externe Grafik eingefügt werden, kann dies über einfache ActionScript-Befehle geschehen. Auch diese Befehle kann man aus früher entworfenen Einstiegsseite übernehmen:

Direkt auf der Bühnen-Zeitleiste findet sich hierzu die Ebene „ActionS“. In diese kann mittels dieser Code-Zeilen eine andere SWF-Datei eingelesen und angezeigt werden:

```
import fho.FCommon;

var Xpos:Number = 705; var Ypos:Number = 120;
var swf:MovieClip; var loader:Loader = new Loader();
var basePath:String = FCommon.getBasePath(loaderInfo.url);
var urlTheme:String = „rohrpost.swf“;
urlTheme = FCommon.getAbsPath(basePath, urlTheme);
var rohrSWF:URLRequest = new URLRequest(urlTheme);

loader.load(rohrSWF);
loader.x = Xpos; loader.y = Ypos;
addChild(loader);
```

Einzig der Name bzw. Speicherort der SWF-Datei, die zusätzlich geladen werden soll, und die Positionierung dieser Datei müssen hierzu geändert werden. Die Positionierung per x und y Werte kann gewisse Zeit in Anspruch nehmen, da man meist verschiedene Kombinationen austesten muss um die richtigen Werte zu finden.

Rohrpost-Grafik

Die Rohrpost-Grafik wurde zunächst als eine Animation entworfen. Da im Usability-Test (s. Kap 6.6 Usability-Test) jedoch schnell klar wurde, dass diese Animation überflüssig ist, wurde sie auf eine Grafik reduziert.

6.3 Lexikoneintrag

Vorgehen

Auch bei der Erstellung eines neuen Lexikon-Eintrags kann auf bereits erstellte Lexikon-Seiten (s. praktische Arbeit „virtlab-Juli2012“: z.B. fla/lexikon/pages/LineareListen/Listen2 fla) zurückgegriffen werden. In diese Dateien kann der neue Lexikon-Text eingestellt und evtl. Grafiken und Veranschaulichungen direkt platziert, bzw. ausgetauscht werden.

Diese neue/veränderte Flash-Datei wird dann mit einem kurzen ActionScript Code versehen, hier „fho.lexikon.FLineareListen.as“. Diese Klasse besteht aus den untenstehenden Zeilen, wieder mit verändertem Klassennamen je nach Lexikoneintrag. Der Code kann für mehrere Seiten des Lexikons eingesetzt werden, so verweisen auch beide Listen-Lexikoneinträge auf die Klasse „fho.lexikon.FLineareListen“:

6 LEKTION „LISTEN“ - UMSETZUNG

```
package fho.lexikon {
    import flash.display.MovieClip;
    import flash.events.MouseEvent;

    public class FLineareListen extends FPage
    {
        public function FLineareListen(){
            init();
        }
        public function init():void {
        }
    }
}
```

Mit entsprechenden Namen versehen, kann die swf-Datei nun bereits in das Lexikon-Verzeichnis (praktische Arbeit „virtlab-Juli2012“: build/virtlab/Lexikon/Data) exportiert werden. Die neue Lexikon-Seite steht jetzt bereit. Sie muss nun in die Struktur des bestehenden Lexikons eingepflegt werden:

Als erstem Schritt sollte man hier die Inhaltsseite des Lexikons anpassen (praktische Arbeit „virtlab-Juli2012“: fla/lexikon/pages/Inhalt.fla). Hier gilt es einen neuen Link mit dem Namen des neuen Lexikoneintrags zu erstellen. Am schnellsten geht dies, wenn man einen bereits vorhandenen Link in der Bibliothek dupliziert und entsprechend fortlaufend benennt. Im Fall des Listen-Eintrags bekam der Link zum Listenthema den Namen „link08“. Im nächsten Schritt kann der neu erstellte Link auf der Bühne an gewünschter Stelle platziert werden. Der Link muss dann der entsprechenden Instanznamen (hier „link08“) erhalten. Der MovieClip-Text muss in dem Namen der neuen Inhalte umbenannt werden, hier „Listen“.

In der Actionsript-Datei die zum Inhaltsverzeichnis gehört (praktische Arbeit „virtlab-Juli2012“: fho/lexikon/FInhalt.as) muss dieser neue Link noch initialisiert werden. In Zeile 7 dieses Codes kann der Link als MovieClip, wie die bereits vorhanden Links, definiert werden:

```
public var link01:MovieClip, link02:MovieClip, link03:MovieClip, link04:MovieClip,
link05:MovieClip, link06:MovieClip, link07:MovieClip, link08:MovieClip;
```

Dann wird dem entsprechenden Link, der bis jetzt nur ein einfacher MovieClip war, in Zeile 14 die Button-Funktionen zuzuweisen:

```
link01.buttonMode = link02.buttonMode = link03.buttonMode = link04.buttonMode =
link05.buttonMode = link06.buttonMode = link07.buttonMode = link08.buttonMode = true;
```

Nun muss dem Link noch dessen „Aufgabe“ zugewiesen werden. Er bekommt hierzu ein Click-Event-Listener dem die entsprechende Lexikon-Seite, im Listen-Eintrag die Referenz „R006“, übergeben wird.

```
link08.addEventListener(MouseEvent.CLICK, function():void{
    anchor.gotoAnchor(„R006“);
});
```

Diese Referenz (hier „R006“) wird in der letzten zu erweiternden Datei festgelegt. Diese Datei (prakti-

6 LEKTION „LISTEN“ - UMSETZUNG

sche Arbeit „virtlab-Juli2012“: Lexikon/megazine.mz3) ist eine xml-Datei in der die eigentliche Logik des Lexikons hinterlegt ist:

Hier wird die Lexikon-Referenz im Tag <bookmarks> festgelegt, im Lexikoneintrag zum Thema Listen z.B. die Referenz „R006“. Um die neue Referenz zu erstellen wird in diesem Tag, nach Vorlage der bereits vorhandenen Einträge, ein neuer Eintrag mit den spezifischen Anpassungen angelegt:

```
<bookmark title="Listen" page="R006" color="#3366EE"/>
```

Die nächste Änderung innerhalb dieses Dokuments wird im zweiten <nav> Tag („<nav rasterize“ ...) vorgenommen. Auch hier kann die Form des Eintrags entsprechend der bereits vorhandenen Einträge übernommen werden. In diesem Fall wird er jedoch nicht fortlaufend, sondern nach dem Alphabet (hier „Listen“) sortiert in die bereits vorhandenen Einträge eingefügt:

```
<lnk url="anchor:R006"><![CDATA[<font size="14" face="calibri, arial">Listen </font>]]></lnk>
```

Dieser Listeneintrag kommt, beim heutigen Stand des VIL, an dritte Stelle (unter Fakultät und Fibonacci-Zahlen und über Rekursion, Struktogramm und Türme von Hanoi).

Die letzte Ergänzung sorgt dafür, dass die entsprechenden Lexikonseiten angezeigt werden. Auch dieser Eintrag muss wieder nach dem Alphabet sortiert, in die bereits vorhandenen Einträge eingefügt werden. Innerhalb aller <page> Tags muss man hier die korrekte Stelle zum Einfügen der neuen Seiten finden.

Um eine Seite mit entsprechendem Anchor (z.B. „R006“) einzufügen muss dieser als Zusatz im <page> Tag angegeben werden. Ist die Seite eine Folgeseite (wie hier die 2. Inhaltsseite zu Listen) kann dieses Attribut weggelassen werden:

```
<page anchor="R006"></page>  
<page></page>
```

Mit dieser letzten Ergänzung sind die neuen Einträge komplett ins Lexikon, und damit auch ins VIL integriert.

Probleme

Eine neue Lexikon-Seite einzufügen erschien, ganz ohne Anleitung und Hinweise, zunächst recht komplex. Demnach musste ich viele Teile der hier beschriebenen Schritte oft wiederholen bzw. Schritte per Trial&Error selbst erkennen. Die Beschreibung des Vorgehens stellt somit nur die funktionierende Endversion einer langen Reihe von Versuchen dar.

6.4 Ver-Kette-t (Visualisierung/Instruktionsanwendung)

Eigentlich kann eine Visualisierung dieser Art recht einfach mit der Vorlage „Structogram fla“ entwickelt werden. In der Anwendung „Ver-Kette-t“ könnten alle Elemente über die entsprechenden Struktogramme (Suchen, Einfügen, Löschen) erstellt und gesteuert werden. Auch übernimmt diese Vorlage das eigentliche Erstellen der Struktogramme und deren Tool-Tipps.

Leider konnte in diesem Fall die Vorlage nicht verwendet werden: Die Vorlage „Structogram fla“ kann nur ein einziges Struktogramm erstellen, auf dessen Grundlage alle Visualisierungen angezeigt werden. Die Möglichkeit zwischen mehreren Struktogrammen zu wechseln wurde nicht implementiert.

Die Anwendung „Ver-Kette-t“ wurde somit auf Grundlage der bereits optimierten Vorlage „templateApplication fla“ erstellt. Die Datei wurde umbenannt und eine zugehörige Klasse (Vorlage „templateApplication.as“, „verkettet_new.as“) erstellt. Nun konnte ein neuer MovieClip (mc_container_animation) hinzugefügt werden, der die verschiedenen Animationen und Struktogramme aufnehmen kann. Eine separate ActionScript-Klasse („animationVerkettet.as“) wurde erstellt, um die Logik der eigentlichen Animationsschritte aus den allgemeinen Seitenfunktionalitäten auszulagern.

Flash-Datei

Auf der oberen Ebene sind die Elemente, die aus der Vorlage der „templateApplication fla“ stammen, angeordnet. Die verschiedenen Animationen und Struktogramme werden zur Laufzeit in die Bühne geladen, in den Container „mc_container_animation“.

Jeder der eigentlichen Animations-Inhalte-MovieClips (MovieClips: mc_anIE - Einfügen, mc_anIS - Suchen, mc_anIL - Löschen) ist nach dem gleichen System abgebaut:

Alle Elemente, die animiert werden, finden sich auf der Bühne wieder. Die Animation selbst wurde über Tweens direkt in der Flash-Datei erstellt. Gesteuert wird diese Animation über die hier hinterlegten „Steps“. Diese „Steps“ sind Zeitslots, die exakt auf den Durchlauf des jeweiligen Struktogramms angepasst sind, und über Actionscript aufgerufen und abgespielt werden können.

Wie bereits erwähnt, mussten auch die Struktogramme „nachgebaut“ werden. Als Design-Vorlage verwendete ich die bereits implementierten Struktogramme der Rekursions-Lektion. Ein Struktogramm (MovieClips: einf_struk - Einfügen, - Suchen, - Löschen) besitzt weitere MovieClips, in denen die einzelnen Schritte des gesamten Struktogramms enthalten sind.

Diese MovieClips sind nach ihrer Position benannt: das erste Element (der Funktionsaufruf) hat die Endung „_funk“, alle weiteren sind durchnummeriert mit step1, step2 etc.. Wenn mehrere Elemente sich eine Position teilen, also eine if-Anweisung mit einem Wahr- und Falsch-Zweig vorhanden ist, werden die Elemente zusätzlich gekennzeichnet: z.B. step3w bzw. step3f. Die Tool-Tipps jedes Elements befinden sich wiederum in einem Movieclip innerhalb dieses Elements. Benannt sind sie nach dem Elternelement mit dem Zusatz „_tip“ (z.B. step3w_tip).

Actionscript

Nachdem die zugrunde liegenden Elemente der Flash-Datei erklärt wurden, kann nun der implementierte Code genauer betrachtet werden. Der Code selbst ist kommentiert, somit soll hier nur ein Einblick in die verschiedenen Abläufe gegeben werden.

Die Klasse „verkettet_new“

Hier finden sich viele der aus der Vorlage „templateApplication.as“ stammenden Elemente. Ergänzt wurde lediglich die Funktionen zur Kommunikation mit der „animationVerkettet“-Klasse (Logik der Animation):

Hierzu wurde der Container für die Animationen als Variable, und eine Instanz der animationVerkettet-Klasse angelegt.

```
public var mc_container_animation:MovieClip;
public var animationV:animationVerkettet = new animationVerkettet;
```

Auch die Select-Box musste noch auf die entsprechenden Möglichkeiten dieser Anwendung angepasst werden. Hier wird mittels eines change-Events überwacht, wenn ein anderes Element ausgewählt wird. Geschieht dies, ruft diese Funktion die Methode „setAnimation“ der animationVerkettet-Klasse auf und übergibt dieser den Namen des Auswahlpunktes zur Weiterverarbeitung.

```
cbTest.setSize(142, 20);
with(cbTest){
    addItem({label:„Suchen“, data:1});
    addItem({label:„Löschen in der Mitte“, data:2});
    addItem({label:„Löschen am Anfang“, data:3});
    addItem({label:„Einfügen in der Mitte“, data:4});
    addItem({label:„Einfügen am Anfang“, data:5});
}

cbTest.addEventListener(Event.CHANGE, function(event:Event):void{
    animationV.setAnimation(cbTest.selectedItem.label);
});
```

Eine weitere Ergänzung betraf die Buttons des Kontrollpanels. Auch hier muss die Möglichkeit bestehen von der „animationVerkettet“-Klasse aus die Buttons auf „aktiv“ oder „nicht aktiv“ zu setzen. Um dies zu erreichen wurde die Funktion **buttonChange()** ergänzt. Dieser Funktion wird der Name eines Buttons und ein Boolean-Wert mitgegeben. Sie bewirkt dass mit diesen Daten der entsprechende Button (btnBegin, btnBack, etc.) auf „aktiv“ (true) oder „nicht aktiv“ (false) gesetzt wird.

Die Klasse „animationVerkettet“

Erst in dieser Klasse werden die eigentlichen Animations-Inhalte initialisiert. Hierzu wird jede Ansicht als MovieClip angelegt (aniS -Suchen, aniE -Einfügen, aniL - Löschen).

In der Konstruktormethode **animationVerkettet()** werden zunächst alle Struktogramm-Punkte und deren Tool-Tipp-Funktionen initialisiert. Der Tool-Tip wird hier jeweils auf „nicht sichtbar“ gesetzt. Dann wird ein Mouse-over und Mouse-out Event zugewiesen, der die Tool-Tips je nach Mausposition ein- oder ausblendet.

Gestartet wird die Animation durch den Aufruf der **setAnimation()** Funktion aus der verkettet_new-Klasse. Je nach erhaltenen Namen (Suchen, Einfügen in der Mitte, etc.) wird die entsprechende Animation, über die Methode „showAnimationStep()“ aufgerufen und sichtbar. Dazu wird die maximale Anzahl an Schritten, die diese Animation besitzt, gespeichert (int „stepMax“) und die entsprechende Ansicht aktiviert (MovieClip „active“). Da hier der Anfang der Animation angezeigt wird, werden alle Variablen und Buttons auf deren Anfangsposition gesetzt.

showAnimationStep() koordiniert die Schritte aller Animationen. Hier wird zunächst erfragt, welche Animation gerade aktiv ist, dann welcher Schritt dieser Animation angezeigt werden soll. Je nach Schritt wird der entsprechende Struktogramm-Befehl hervorgehoben und alle anderen wieder auf ihren normalen Wert gesetzt. Über den Befehl „gotoAndPlay()“ wird der aktuelle Schritt der Ketten-Animation aufgerufen und abgespielt.

Wenn der „Play“-Button geklickt wurde übernimmt ein Timer, mit einer zum Animationsschritt passenden Verzögerung, das „weiterklicken“ zum nächsten Schritt. Solange die Animation nicht am Ende angekommen ist, oder der „Play“-Button erneut geklickt wurde, zählt diese Abfrage die Animationsschritte immer hoch und startet das Abspielen über die Funktion „replay()“ neu.

Die weiteren implementierten Inhalte sind die Funktionen der Buttons im Kontrollpanel. Je nach geklicktem Button muss hier entweder die Animation zurück springen, einen Schritt zurück gehen, in den Abspielmodus wechseln, einen Schritt vor gehen, oder ganz ans Ende der Animation springen. Hierzu muss jeweils die Klickbarkeit der Buttons überprüft und gegebenenfalls geändert, sowie der entsprechende Animationsschritt wieder über die Methode „showAnimationStep()“ angezeigt werden.

6.5 ICE-Bahnhof (Experiment)

Auch in der zweiten Anwendung (Experiment „ICE-Bahnhof“) verwendete ich die Vorlagen der „templateApplication.fla“ und „templateApplication.as“ und entwickelte auf dieser Grundlage die weiteren Konzepte dieser Anwendung. Da hier die Anwendung nicht über ein Struktogramm gesteuert werden kann, ist dies auch die vorgesehene Methode zur Entwicklung.

Flash-Datei

Auch hier wurden die Elemente, die aus der Vorlage der „templateApplication.fla“ stammen, wiederverwendet. Der Text der Lösungsidee, das Textfeld zum Zählen der Schritte, die Gleise und „Hilfsvariablen“

(H) konnten direkt platziert werden, da sie zur Laufzeit nicht umplatziert werden müssen. Die verschiedenen Züge, Haken, deren Verbindungen und die Kreise werden, jeweils auf die gewählte Aufgabe angepasst, erst zur Laufzeit erzeugt und platziert. Auch hier spielen sich alle Veränderungen in dem Fenster der Animation ab (MovieClip „mc_animation_content“).

Die Wagons wurden als MovieClip erstellt und nach ihrer Nummer benannt (Instanzname: „wagon1“, „wagon2“, „wagon3“ und „wagon4“). Ebenfalls als MovieClip hinterlegt sind die Grafik des Kreises (Instanzname: „kreis“) und die verschiedenen ausgerichteten Haken (Instanzname: „hakenW“ - Haken der Wagons, „hakenO“ - Haken der oberen Hilfsvariable, „hakenU“ - Haken der unteren Hilfsvariable). Einzig die Verbindungslinien zwischen dem Haken und dessen Ursprungsposition werden komplett per Actionscript erzeugt.

Actionscript

In dieser Anwendung wurden wieder zwei Klassen angelegt um die verschiedenen Funktionen zu implementieren. Die „bahnhof.as“ beinhaltet hierbei die angepassten Funktionen der Vorlage „templateApplication.as“. Die eigentliche Logik des Experiments wurde in die Klasse „zugAnimation“ ausgelagert. Beide Klassen sind kommentiert; um einen Einblick zu erhalten sollen ihre Funktionalitäten und Abläufe hier kurz vorgestellt werden.

Die Klasse „bahnhof“

Ergänzt wurden in dieser Klasse, wie in der Ver-Kette-t-Anwendung, die Variablen des MovieClips der die Wagons enthält sowie die Instanz der „zugAnimation“-Klasse:

```
public var mc_animation_content:MovieClip;
public var zugA:zugAnimation = new zugAnimation();
```

Auch die Select-Box und deren Change-Event wurden angepasst. Hier wird bei jeder Wahl einer neuen Aufgabe die Funktion newSzene() der „zugAnimation“-Klasse aufgerufen. Dieser Funktion wird der Name der aktuellen Auswahl übergeben:

```
with(cbTest){
    addItem({label:„Löschen“, data:1});
    addItem({label:„Einfügen“, data:2});
    addItem({label:„Kombination“, data:3});
}
cbTest.addEventListener(Event.CHANGE, function(event:Event):void{
    zugA.newSzene(cbTest.selectedItem.label);
});
```

Auch wurde wieder eine Methode zur Schaltung der Buttons implementiert („changeBtn()“). Da bei einem Experiment nur der „zurück zum Anfang“-Button aktiv werden kann, wird dieser Funktion nur ein Boolean-Wert übergeben. Auch hier setzt der Wert „true“ den Button „aktiv“ und „false“ den Button auf „nicht aktiv“.

6 LEKTION „LISTEN“ - UMSETZUNG

Im Gegenteil zur Ver-Kette-t Anwendung werden im Experiment auch MIA-Rückmeldungen eingesetzt. Je nach Textmenge kann hier die Form der Sprechblase auf „Bubble“ (wenig Text, ca. 1 Satz) oder „Rect“ (ca. 3 Sätze) umgestellt werden. Diese Angabe wird, zusammen mit dem Text der ausgegeben werden soll, der Funktion „miaSay()“ übergeben. Diese Funktion greift wiederum auf die bereits in der Rekursions-Lektion implementierten Eigenschaften der „fhoMia“ zurück:

```
public function miaSay (s:String, shape:String):void{
    if (s == „“)MIA.hide();
    else if (shape == „Bubble“) MIA.say(s, fhoMIA.HideArrow, fhoMIA.BubbleShape);
    else if (shape == „Rect“) MIA.say(s, fhoMIA.HideArrow, fhoMIA.RectangleShape);
}
```

Die Klasse „zugAnimation“

Die Logik des Experiments wird von dieser großen Klasse aus gesteuert und überwacht. Zunächst werden hier die verschiedenen Variablen angelegt. Hilfsvariablen die verschiedenen Zwischenschritte speichern, sowie die verschiedenen Wagons, Haken, Verbindungslinien und Kreise der Anwendung.

Die **init()** Funktion wird aus der „bahnhof.as“ aus aufgerufen. Hier werden alle Haken, Verbindungen, Kreise und Wagons erstellt und der Bühne bzw. dem MovieClip „mc_animation_content“ hinzugefügt, und entsprechend ihrer Position und Zugehörigkeit benannt.

Die Funktion **newSzene()** wird ebenfalls aus der bahnhof-Klasse aufgerufen. Über diese werden dann die jeweiligen Aufgaben des Experiments (szeneEinf(), szeneLoesch(), szeneKombi()) aktiviert. Auch die Haken werden hier mit einem Event-Listener (Drag&Drop) versehen und klickbar gemacht.

Die einzelnen Aufgaben werden über deren zugehörige Funktionen (szeneEinf() - Szene der Aufgabenstellung „Einfügen“, szeneLoesch() - Szene der Aufgabenstellung „Löschen“, szeneKombi() - Szene der Aufgabenstellung „Kombination“) erstellt. Jede dieser Funktionen ist hierbei vom Ablauf identisch aufgebaut:

Zunächst wird die Startposition der Wagons angelegt. Hierzu wird über Hilfsvariablen festgelegt, welcher Zug auf welchem Gleis („1“ - oberes Gleis oder „2“ - unteres Gleis) und auf welcher Position („1“ - vorne, „2“ - mitte oder „3“ - ende) steht. Wenn das Gleis den Wert „0“ erhält wird der Wagon und alle mit ihm verbundenen Elemente nicht platziert.

Im zweiten Schritt werden die Wagons wirklich positioniert, über die Funktion „setWKH()“. Nun muss nur noch der Schritte-Zähler zurückgesetzt, und die Kreise der Wagons unsichtbar gemacht werden.

setWKH() ist die Funktion die alle Wagons, Kreise und Haken sichtbar macht und richtig positioniert. Auch die Verbindungslinien zwischen den gezogenen Haken und deren Ursprungspositionen werden hier erstellt. Die Positionierung geschieht über hinterlegte Werte zur Positionierung. Für jede Wagon-, Kreis- und Haken-Position wurde, ebenfalls in dieser Klasse, ein x- und y-Wert hinterlegt. Dieser wird hier aufgerufen und den richtigen Elementen zugewiesen.

6 LEKTION „LISTEN“ - UMSETZUNG

Die Verbindungslinien wurden als shape-Objekt erstellt. Diesen wird, ebenfalls mittels der hinterlegten Positionierungswerte, deren Start- und Endpunkt zugewiesen.

Durch diese Hinterlegung der verschiedenen Positionen wurde eine bessere Übersicht über die Positionierung gegeben. Ein erster Versuch alle Wagons und deren Elemente mathematisch durch Abstandsvariablen zu positionieren, wurde durch die Komplexität dieser Anwendung so unüberschaubar, dass eine andere Methode gewählt werden musste.

Die oben beschriebenen Funktionen stellen die Vorbereitungen der eigentlichen Interaktion mit dem Experiment dar. Das Experiment kann über die Funktionen `drag()` und `drop()` vom Lernenden gesteuert werden. Diese Funktionen starten bzw. enden immer dann, wenn ein Haken gezogen bzw. wieder losgelassen wird (Mouse-Events DOWN und UP).

Wird auf einen Haken geklickt (Mouse-Event DOWN) startet die Funktion **`drag()`**. In dieser Funktion wird der geklickte Haken gespeichert und alle Kreise, in die der Haken eingehängt werden kann, sichtbar gemacht. Auch die Funktion, die Kreise hervorzuheben wenn der Haken an der richtigen Position zum einhaken ist, wird von hier aus gesteuert. Über das Mouse-Event „MOVE“ wird, immer wenn die Maus bewegt wird, die Funktion `drawLine()` aufgerufen. Diese eine Linie vom ergriffenen Haken zu dessen Ursprungsort zieht.

Sobald ein Haken wieder losgelassen wird (Mouse-Event UP), wird über die Funktion **`drop()`** die Position des Hakens überprüft und alle sich daraus ergebenden Veränderungen initialisiert. Hierzu wird zunächst die Funktion „`drawLine()`“ entfernt und alle Kreise wieder ausgeblendet. Die Anzahl der benötigten Schritte wird hochgezählt und der Button „Zurück zum Anfang“ aktiv geschaltet. Sobald dies geschehen ist kann die Funktion „`checkAnchor()`“ aufgerufen werden.

`checkAnchor()` überprüft wo der Haken abgesetzt wurde. Wenn der Haken von einem Wagon gelöst wurde wird die Funktion „`wagonsLoeschen()`“ aufgerufen. Wurde der Haken an einen neuen Wagon gehängt übernimmt die Funktion „`wagonsNeuAnordnen()`“ die weiteren Schritte. Beide Funktionen ordnen die Wagons neu an und speichern die neuen Positionen aller Wagons in Hilfsvariablen ab. Diese können dann wieder von der Funktion „`setWKH()`“ umgesetzt werden.

Nachdem die Wagons neu positioniert und auch an der neuen Position angezeigt werden wird noch überprüft, ob MIA eine Nachricht ausgeben soll. Wenn ein Wagon versehentlich gelöscht wurde, oder die Wagons auf dem falschen Gleis richtig angeordnet sind, gibt es eine entsprechende Rückmeldung. Wurde die Lösung gefunden wird die Anwendung beendet indem die Funktion „`end()`“ aufgerufen wird.

Der Methode **`end()`** wird ein Boolean-Wert mitgegeben. „`true`“ bedeutet hierbei, dass die Lösung mit der minimalen Anzahl an Schritten gefunden wurden. „`false`“ dementsprechend, dass mehr Schritte benötigt wurden um die Aufgabe zu lösen. Je nach Wert ändert sich die Rückmeldung die MIA gibt. Unabhängig davon wird die Interaktion mit dem Experiment hier beendet, indem die Haken nicht mehr geklickt werden können und alle Event-Listener entfernt werden.

wagonsLoeschen() ist eine der Methoden, welche die Neuordnung der Wagons realisiert. Hier wird über viele Abfragen ermittelt, ob der umgesetzte Haken an einem Wagon befestigt war und welche Wagons dementsprechend gelöscht werden müssen.

wagonsNeuAnordnen() übernimmt eine ähnliche Funktion, ist jedoch um einiges komplexer. Hier wird ein Haken an einen anderen Wagon gehängt. Dementsprechend muss überprüft werden, ob der Haken zuvor einen anderen Wagon gehalten hat. Dieser Wagon, sowie die nach ihm folgenden, müssen dann zum Löschen vorgemerkt werden. Im nächsten Schritt wird überprüft welcher Wagon verschoben werden soll und ob dadurch Wagons, die zum Löschen vorgemerkt wurden, evtl. nicht gelöscht werden müssen. Auch hier ist das Ergebnis ein im Hilfsvariablen gespeicherter „Konstruktionsplan“ für die neue Positionierung der Wagons, der von der `setWKH()`-Funktion umgesetzt werden kann.

6.6 Usability-Test

Der zweite Usability-Test im Rahmen dieser Thesis wurde durchgeführt, als alle Inhalte der neuen Lektion (Lexikoneintrag und Einstiegsseite zum Thema Listen sowie die Anwendungen Ver-Kette-t und ICE-Bahnhof) bereits implementiert waren.

Der Aufbau dieses Tests ähnelt dem ersten. Einziger Unterschied war, dass die Probanden an einer komplett funktionierenden Version des VILs deren Inhalte testen konnten. Besonderes Augenmerk wurde hier natürlich auf die Inhalte und Funktionen der neu eingebundene Listen-Lektion und die Änderungen zu früheren Inhalten gelegt.

Der Usability-Test wurde wieder als Thinking-Aloud Test durchgeführt. Unter den vier teilnehmenden Probanden waren zwei, die keine Vorkenntnisse in der Informatik hatten und zwei mit (Grund-)Kenntnissen in der Informatik und somit auch des Listen-Konzepts.

Ablauf des Tests

- Test-Rahmen besprechen (Text im Rahmen meiner Thesis. Kurz die Zielgruppe des VIL umreißen. Es kann vollkommen offen die eigene Meinung geäußert werden, da die Ideen zur Änderung im letzten Jahr von einer Projektgruppe entwickelt wurden. Nur kritische Rückmeldungen bringen das Projekt weiter.)
- Der Proband soll sich so verhalten, als würde er diese Lektion bearbeiten und dabei kurz erklären welchen Eindruck er bekommt. (Ob irgendetwas verwirrend ist, ob die Darstellung zur Zielgruppe passt, ob etwas verbessert werden kann etc.)
- Navigation: Als „warm-up“ wird noch einmal kurz das neue Navigationskonzept mit dem früheren verglichen. Hierzu sollen die Probanden eine der Lektionen (Listen oder Rekursion) aufrufen, zur dritten Unterseite dieser Lektion navigieren, um dann auf einem anderen Weg (mit einer anderen Navigationsmöglichkeit) wieder auf die Einstiegsseite der Lektion zurückzukommen. Sobald diese Aufgabe in einer Lektion erfüllt wurde soll der Proband in den Laborraum zurückkehren, und die gleiche Aufgabe in der anderen Lektion lösen. (Die Inhalte sind für diese Aufgaben zunächst egal, es geht nur um die Navigation)
- Augenmerk beim Testen der Navigation: Welche Navigationsmöglichkeit wurde zuerst verwendet?

6 LEKTION „LISTEN“ - UMSETZUNG

Wie schnell wurden die beiden Navigationsmöglichkeiten gefunden? Wie bewerten die Probanden selbst die Navigationsmöglichkeiten - was war gut/ was schlecht?

- Einstiegsseite Listen: Der Proband wird gebeten diese Lektion so zu verwenden als ob er/sie damit lernen würde. Zusätzlich sollen alle Unklarheiten, Probleme im Verständnis, Anmerkungen zur Gestaltung geäußert werden. Wenn der Proband zu erkennen gibt, dass er/sie zur nächsten Seite weitergehen würde wird noch einmal spezifisch nachgefragt, wie die Einstiegsseite und deren Inhalte wahrgenommen wurden.
- Ver-Kette-t: Auch hier soll der Proband so tun, als würde er mit dieser Anwendung lernen und wieder seine Anmerkungen hierzu äußern.
- Augenmerk lag hier vor allem auf der zusätzlichen Beschreibung der „Bedienung“ und ob der Proband den Inhalt der Anwendung und deren Möglichkeiten zur Interaktion versteht. Auch hier wird im Anschluss noch einmal genau nachgefragt wie diese Anwendung beurteilt wird.
- ICE-Bahnhof: Bei dieser Anwendung konnte der Ehrgeiz der Probanden angesprochen werden. Hier bestand die Aufgabe darin die drei Aufgabenstellungen zu lösen - je nach Vorwissen/Motivation des Probanden mit dem Zusatz die beste Lösung zu finden.
- Das Augenmerk lag hier wieder in dem verwenden der zusätzlichen Beschreibung der „Bedienung“ und generell der intuitiven Bedienbarkeit des Experiments. Auch hier wurde im Anschluss nachgefragt wie diese Anwendung beurteilt wird.
- Zum Abschluss wurden den Probanden noch Fragen zur gesamten Wahrnehmung des VILs und der Listen-Lektion gestellt: Sind die Anwendungen und Texte alle verständlich? Kann man mit dem VIL nach Meinung der Probanden lernen? Würden sie es auch zum Lernen verwenden? Wie würden sie allgemein das VIL und die Listen-Lektion beurteilen? Hat sie etwas am VIL gestört/ nicht so funktioniert wie gedacht?

Ergebnisse und Änderungen

Um Verwirrungen vorzubeugen will ich kurz vorweg nehmen, dass die hier vorgestellten Änderungen bereits komplett in die jetzige Listen-Lektion übernommen wurden. Ich werde somit kurz das Ergebnis des Tests beschreiben und jeweils einen Einblick in die Ursachen der Kritikpunkte sowie die implementierte Lösung geben.

Navigationskonzepte

Die bereits im ersten Test zu erkennende Tendenz hat sich auch hier bestätigt. Alle Probanden kamen sichtlich mit der Navigation der Listen-Lektion besser und schneller zurecht als mit der früheren Navigation der Rekursions-Lektion.

Wie im Test der Projektgruppe (FEHRENBACH (2012)) wurde auch von diesen Probanden oft die Lupen-Navigation nicht verstanden. Auch das Konzept der zwei Lernwege verwirrte wieder.

Die Listen-Navigation wurde dementsprechend vor allem wegen ihrer Klarheit gelobt. Hier fand jeder Proband schnell die entsprechenden Pfeile zur Navigation und nach einer entsprechenden Nachfrage

auch sehr schnell die neue Zettel-Navigation. Da hier keine Verbesserungsvorschläge genannt wurden, blieb die neue Navigation wie geplant erhalten.

Einstiegsseite der Listen-Lektion

Der Text wurde hier als sehr verständlich und klar beurteilt, die Textmenge als gutes Maß. Allen Probanden war die Darstellung mittels Rohrpost-System verständlich. Probanden mit Vorkenntnissen waren sich hierbei nicht sicher, ob Lernende ohne Vorkenntnisse diese Anschauung verstehen. Beide Probanden ohne Vorkenntnisse gaben jedoch an dieses Bild verstanden zu haben.

Ursprünglich war die Grafik der Rohrpost mit einer Animation verbunden. Im Test gab allerdings jeder Proband an, dass die Animation an dieser Stelle überflüssig sei. Gerade da der begleitende Text auch ohne die Animation (nur anhand der Grafik) verständlich ist. Da zwei der Probanden die Animation zusätzlich als eher verwirrend ansahen, wurde diese komplett weggelassen.

Anwendung „Ver-Kette-t“

Hier wurde von jedem Probanden gelobt, dass eine zusätzliche Beschreibung der Bedienung vorhanden ist und diese gleichzeitig zu den anderen Inhalten angezeigt werden kann. Auch die Möglichkeit der Instruktionsanwendung (dass eine Visualisierung parallel zum Struktogramm abläuft) wurde ausdrücklich gelobt. Die Möglichkeiten die Listen-Struktogramme von allen möglichen Situationen aus anzeigen zu lassen wurde positiv bewertet.

Zu dieser Anwendung fand keiner der Probanden einen Verbesserungsvorschlag oder einen Kritikpunkt. Alle bewerteten die Idee und Umsetzung als gut gelungen und hervorragend geeignet, um die dargestellten Stuktogramme zu verstehen.

Anwendung „ICE-Bahnhof“

Wie es von einer neu entwickelten Interaktionsmöglichkeit erwartet werden kann, traten hier die größten Notwendigkeiten zur Verbesserung auf.

Vor allem der Text zur Bedienung musste nach dem ersten Test noch einmal überarbeitet werden, da hier zunächst die falschen Stichworte erklärt wurden. Nachdem ich der ersten Probandin beim Lesen der Beschreibung und dem Verwenden dieser Anwendung zugesehen habe, konnte ich gut nachvollziehen was eigentlich in der Bedienung stehen sollte, um das Konzept verständlich zu machen. Nach dem eine neue Anleitung zur Verfügung stand war die Interaktion klar. Zuvor hatte vor allem der Hinweis gefehlt, dass nur die Haken angefasst werden können.

Eine weitere Erweiterung war, dass eine Rückmeldung gegeben wird wenn der Haken über einem Wagon-Kreis eingehakt werden kann. Ohne diese Rückmeldung war den Probanden in der ersten Version nicht klar, ob sie die richtige Stelle zum loslassen gewählt hatte.

Insgesamt wurde die Anwendung gut bewertet. Besonders da die Interaktion Spaß macht und die Rückmeldungen augenscheinlich dazu anregen die beste Lösung zu finden (bis auf eine Probandin wollten alle von sich aus die beste Möglichkeit finden).

Allgemeines/ Zusammenfassung

Insgesamt ist das Ergebnis dieses Tests, auch im Hinblick auf den Test des früheren VILs, sehr positiv ausgefallen. Alle Probanden können sich vorstellen mit dem VIL zu lernen und fanden die Inhalte der Listen-Lektion gut und ansprechend dargestellt.

Alle positiven Effekte die eine hohe Interaktivität hervorbringen sollte, treten nun in den Vordergrund, da sich die Lernenden nicht mehr mit der Hürde der Bedieungs-Schwierigkeiten konfrontiert sehen. So wurde z.B. von allen Probanden ausdrücklich gelobt, dass die Erkundung der Listen mit dem Experiment „ICE-Bahnhof“ Spaß macht und durch die eigene Tätigkeit der Eindruck vermittelt wird etwas gelernt zu haben.

Die hier beschriebenen Ergebnisse stellen eine Zusammenfassung der durchgeführten Tests dar. Der Mitschrieb aus den Tests der einzelnen Probanden kann im Anhang unter „Mitschrieb des Usability Tests“ (S. 112ff) nachgelesen werden.

6.7 Online-stellen der neuen Inhalte

Nachdem alle Inhalte umgesetzt und getestet wurden, konnte die neue Lektion online gehen. Das eigentliche Hochladen der Daten übernahm hier Carolina Bernal (Assistentin an der Hochschule Offenburg), da sie die nötigen Zugangsdaten besitzt.

Da ich mich an dem bereits realisierten Lektionen des VIL orientiert hatte, konnte der neue Inhalt ohne Probleme oder nötigen Korrekturmaßnahmen online gestellt werden.

7 ZUSAMMENFASSUNG & FAZIT

Meine Tätigkeiten umfassten im Rahmen dieser Thesis sehr vielfältige Aufgaben. So habe ich zunächst die Grundlagen des E-Learnings und dessen Erstellung recherchiert und festgehalten. Aus diesen gewonnenen Informationen konnte ich ein Regelwerk für die Erstellung bzw. Neuerschaffung von intuitiver Interaktion erstellen, welches bis jetzt noch in keiner Literatur eindeutig beschrieben wurde.

Der zweite Teil meiner Arbeit bestand in der praktischen Umsetzung einer neuen E-Learning Lektion in der ich die zuvor recherchierten Grundlagen einsetzte. Hier habe ich Änderungen bzw. Erweiterungen zu den bereits bestehenden Inhalten konzipiert und implementiert. Im Rahmen dieser Änderungen habe ich ein neues Navigationskonzept entworfen und umgesetzt, sowie die neu-Positionierung verschiedener Interaktionselemente übernommen.

Auf Grundlage dieser erweiterten Version des Virtuellen Informatiklabors habe ich die Inhalte der Listen-Lektion komplett konzipiert und alles, bis auf die Struktogrammentwicklung, umgesetzt und in die bestehende Anwendung integriert.

Die besondere Schwierigkeit lag hier vor allem darin, ohne Dokumentation und mit nur teilweise kommentiertem Code die Änderungen/Erweiterungen durchzuführen. Dank dieser Erfahrung konnte ich die Ermahnung vieler Professoren, entwickelte Codes immer zu kommentieren und möglichst auch zu dokumentieren, gut nachvollziehen. Ich habe meinen Programm-Code aus diesem Grund kommentiert und alle meine Vorgehensweisen beschrieben um den nächsten Projekten einen besseren Startpunkt zu geben.

Auch die Umsetzung des Experiments „ICE-Bahnhof“ stellte, aufgrund der hohen Komplexität, eine große Herausforderung für mich dar. Dieser habe ich mich jedoch gerne gestellt und diese Aufgabe, obwohl sie nicht zwingend Teil der Aufgabenstellung war (es sollte mindestens die Visualisierung umgesetzt werden), gemeistert.

Die einzige noch fehlende Anwendung der Listen-Lektion, die To-Do Liste (Struktogrammentwicklung), wird noch in einem anderen Rahmen implementiert werden. Für den Rahmen dieser Thesis war diese Aufgabe leider zu komplex und zeitaufwendig.

Wie die Ergebnisse des Usability-Tests zeigen, hat das VIL ein großes Potenzial für Lernende. Gerade im Einsatz, begleitend zu einer Vorlesung oder einem entsprechenden Seminar, werden viele Lernende gerne auf diese interaktive Art des Lernens zurückgreifen.

Allerdings sollte, wie im theoretischen Teil dieser Thesis beschrieben, eine solche Anwendung nicht als Vorlesungs-/Seminar-Ersatz behandelt werden. Es sollte vielmehr dem Reflektieren oder Vorbereiten der darin behandelten Themen dienen.

Dies gilt nicht alleine für das VIL, sondern für alle E-Learning Anwendungen. Leider wird in der heutigen Zeit oft versucht E-Learning als Ersatz der eigentlichen Veranstaltungen einzusetzen. Es sollte viel mehr das Potenzial von E-Learning Angeboten, als begleitendes Medium mit hervorragender Möglichkeit zur Interaktivität und Multimedialität, erkannt und in diesem Sinne eingesetzt werden.

DANKSAGUNG

Mit der Abgabe dieser Bachelor Thesis endet für mich eine interessante, schöne, jedoch auch anstrengende Studienzeit. Voller Stolz auf meine bisherigen Leistungen möchte ich an dieser Stelle meine Aufmerksamkeit den Personen widmen, die es mir ermöglichten nun an dieser Stelle zu stehen.

Ich bedanke mich bei meinen betreuenden Professoren, Prof. Dr. Volker Sanger und Prof. Dr-Ing. Claudia Schmidt. Sie haben die E-Learning Anwendung des VIL ins Leben gerufen, und mir die Moglichkeit gegeben meine Thesis in diesem interessanten und abwechslungsreichen Umfeld zu schreiben. Auch danke ich ihnen fur die stets freundliche, einwandfreie und professionelle Betreuung der Arbeit. Ihre Ruckmeldungen, Anregungen und auch Diskussionen trugen mageblich zur Entstehung der Listen-Lektion bei.

Im Rahmen der Thesis bedanke ich mich auch bei Carolina Bernal und Khan Do, die mir bei kleinen, aber entscheidenden Problemen geholfen haben und mich bei der Umsetzung der Lektion unterstutzten.

Besonderer Dank gilt meiner Familie und meinen Freunden, fur die Unterstutzung und Motivation die ich durch sie erfahren durfte. Allein durch sie konnte ich mein Studium in nur sieben Semestern und mit einem sehr positiven Ergebnis absolvieren.

Fur die Zeit meines Studiums danke ich meinen Kommilitonen, mit denen ich die guten, aber auch schwierigen Zeiten des Studiums durchlebt habe.

Zudem danke ich der Printus GmbH, besonders Herrn Schrempp und Frau Specht, fur das Vertrauen das sie mir wahrend meines Praxissemesters und meiner anschließenden Werkstudententatigkeit zukommen lieen. Die dort gewonnenen, sowohl praktischen als auch menschlichen, Erfahrungen haben mich fur den weiteren Verlauf meines Studiums und wohl auch daruber hinaus sehr gepragt und motiviert.

Auch danke ich allen Informatik-Professoren der Hochschule Offenburg, die mich fur die Themen der Informatik begeisterten. Erst durch sie konnte ich den Spa, den diese Fachrichtung mit sich bringt und mein Talent in diesem Bereich entdecken und nutzen.

Kapitel 2.1 (Das Virtuelle Informatiklabor)

Abb. 1	Laborraum des VIL (Quelle: VIL(2012))	Seite 8
Abb. 2	Animation „Türme von Hanoi“ im VIL (Quelle: VIL(2012))	Seite 8
Abb. 3	Struktogramm „Schneckenhaus“ im VIL (Quelle: VIL(2012))	Seite 9
Abb. 4	MIA, Guide des VIL (Quelle: VIL(2012))	Seite 9
Abb. 5	Navigation und Kontrollpanel im VIL (Quelle: VIL(2012))	Seite 10
Abb. 6	Tool Box des VIL (Quelle: VIL(2012))	Seite 10

Kapitel 2.2 (Listen in der Informatik)

Abb. 7	Visualisierung einer linearen Liste (eigener Entwurf)	Seite 11
Abb. 8	Visualisierung – Suchen in einer Liste (eigener Entwurf)	Seite 13
Abb. 9	Struktogramm – Suchen in einer Liste (eigener Entwurf)	Seite 13
Abb. 10	Visualisierung – Einfügen in eine Liste (eigener Entwurf)	Seite 13
Abb. 11	Struktogramm – Einfügen in eine Liste (eigener Entwurf)	Seite 14
Abb. 12	Visualisierung – Löschen aus einer Liste (eigener Entwurf)	Seite 14
Abb. 13	Struktogramm – Löschen aus einer Liste (eigener Entwurf)	Seite 14

Kapitel 4 (Änderungen & Erweiterungen für die Lektionen)

Abb. 14	modifizierte Navigation (ausgeklappt) (eigener Entwurf)	Seite 35
Abb. 15	modifizierte Seitenansicht für Animationen und Experimente (eigener Entwurf)	Seite 37
Abb. 16	modifizierte Tool-Box für Struktogrammentwicklung (eigener Entwurf)	Seite 37
Abb. 17	Screen der neuen structogram fla (links) und der früheren (Quelle: Screenshot Flash-Entwicklungsansicht)	Seite 40
Abb. 18	Screen der neuen templateApplication fla (links) und der früheren (rechts) (Quelle: Screenshot Flash-Entwicklungsansicht)	Seite 41

Kapitel 5 (Lektion „Listen“ - Idee und Konzeption)

Abb. 19 Struktur der Listen-Lektion (eigene Skizzierung)	Seite 50
Abb. 20 grobe Aufteilung der Elemente der Einstiegsseite (eigene Skizzierung)	Seite 51
Abb. 21 Grafik des Rohrpost-Systems (eigener Entwurf)	Seite 52
Abb. 22 grobe Aufteilung der Elemente der Instruktiionsanwendung (eigene Skizzierung)	Seite 53
Abb. 23 Struktogrammentwurf zum Suchen in einer Liste/Kette (eigener Entwurf)	Seite 54
Abb. 24 Ansichten zum Suchen in einer Liste/Kette (eigene Skizzierung)	Seite 55
Abb. 25 Struktogrammentwurf zum Einfügen in eine Liste/Kette (eigener Entwurf)	Seite 56
Abb. 26 Ansichten zum Einfügen am Anfang einer Liste/Kette (eigene Skizzierung)	Seite 57
Abb. 27 Ansichten zum Einfügen in der Mitte einer Liste/Kette (eigene Skizzierung)	Seite 58
Abb. 28 Struktogrammentwurf zum Löschen aus einer Liste/Kette (eigener Entwurf)	Seite 58
Abb. 29 Ansichten zum Löschen des ersten Elements einer Liste/Kette (eigene Skizzierung)	Seite 59
Abb. 30 Ansichten zum Löschen aus der Mitte einer Liste/Kette (eigene Skizzierung)	Seite 60
Abb. 31 grobe Aufteilung der Elemente des Experiments (eigene Skizzierung)	Seite 62
Abb. 32 Ansicht des Experiments (eigener Entwurf)	Seite 63
Abb. 33 Wagon-Arten (eigener Entwurf)	Seite 64
Abb. 34 Visuelle Rückmeldung, wenn ein Haken ergriffen wird (eigener Entwurf)	Seite 64
Abb. 35 Ansicht einer Experiment-Aufgabe (eigener Entwurf)	Seite 64
Abb. 36 Ansicht der Löschen-Aufgabe, Start (oben) & Lösung (eigener Entwurf)	Seite 65
Abb. 37 Ansicht der Löschen-Aufgabe, Lösungsweg (eigener Entwurf)	Seite 65
Abb. 38 Ansicht der Einfügen-Aufgabe, Start (oben) & Lösung (eigener Entwurf)	Seite 66
Abb. 39 Ansicht der Einfügen-Aufgabe, Lösungsweg (eigener Entwurf)	Seite 66
Abb. 40 Ansicht der Kombinations-Aufgabe, Start & Ende (eigener Entwurf)	Seite 67
Abb. 41 grobe Aufteilung der Elemente im Struktogrammentwurf (eigene Skizzierung)	Seite 69
Abb. 42 Lösungs-Struktogramme (eigener Entwurf)	Seite 71
Abb. 43 Ansichten der Visualisierung der To-Do Liste (eigener Entwurf)	Seite 72
Abb. 44 grobe Aufteilung der Elemente im Lexikon-Eintrag (eigene Skizzierung)	Seite 77
Abb. 45 Darstellung zur Erklärung der Listen-Elemente (eigener Entwurf)	Seite 77

Bücher & Schriften

- ARNOLD, Patricia; Kilian, Lars; Thillosen, Anne; Zimmer, Gerhard (2011): **Handbuch E-Learning - Lehren und Lernen mit digitalen Medien**; 2. Auflage; W. Bertelsmann Verlag; Bielefeld
- BÖHRINGER, Joachim / Bühler, Peter / Schlaich, Patrick (2008): **Kompendium der Mediengestaltung für Digital und Printmedien – Band 1**; 4. Auflage, Springer-Verlag; Berlin Heidelberg
- DORAU, Rainer (2011): **Emotionales Interaktionsdesign – Gesten und Mimik interaktiver Systeme**; 1. Auflage; Springer-Verlag; Berlin Heidelberg
- FEHRENBACH, Simone/ Kriegeskorte, Claudia/ Pohla, Diana/ Schönberg, Alena/ Weigel, Adrian (2012): **Ein-satz und Lernerfolgsanalyse einer E-Learning-Lektion zur Rekursion**; Projektarbeit an der Hochschule Offenburg
- KLIMSA, Paul; Issing, Ludwig (Hrsg.) (2011): **Online-Lernen - Handbuch für Wissenschaft und Praxis**; 2. Auflage; Oldenbourg Wissenschaftsverlag; München
- KRUG, Steve (2006): **Don't Make Me Think! Web Usability - das intuitive Web**; 2. Auflage; mitp /Redline GmbH; Heidelberg
- NIELSEN, Jakob / Tahir, Marie (2002): **Homepage Usability – 50 enttarnte Websites**; Markt + Technik Verlag; München
- SAAKE, Gunter / Sattler, Kai-Uwe (2006): **Algorithmen und Datenstrukturen - Eine Einführung mit Java**; 3. Auflage; dpunkt.verlag; Heidelberg
- SCHMIDT, Claudia / Sänger, Volker (2011): **Das Virtuelle Informatiklabor – Entwurf, Design und Realisierung einer E-Learning-Umgebung zum Erlernen von Algorithmen**; Hochschule Offenburg
- STAPELKAMP, Torsten (2010): **Interaction- und Interfacedesign – Web-, Game-, Produkt- und Service-design**; 1. Auflage; Springer-Verlag; Berlin Heidelberg
- THESMANN, Stephan (2010): **Einführung in das Design multimedialer Webanwendungen**; 1. Auflage; Vieweg+Teubner|GWV Fachverlage GmbH; Wiesbaden
- WESCHKALNIES, Nick / Gasser, Sven (2010): **Adobe Flash CS5 - Das umfassende Handbuch**; 1. Auflage; Galileo Design / Galileo Press, Bonn
- WIRTH, Niklaus (1986): **Algorithmen und Datenstrukturen mit Modula 2**; 4. Auflage; B. G. Teubner Verlag; Stuttgart

Websites

- ALTHBAUER, Heinrich (Hrg.) / Stangl, Werner (2012b): **eLearning, E-Learning, Blended Learning**; <http://www.stangl-taller.at/ARBEITSBLAETTER/LERNEN/Elearning.shtml> [Stand: 10.04.2012]
- ALTHBAUER, Heinrich (Hrg.) / Stangl, Werner (2012): **Lexikon für Psychologie und Pädagogik** [Stand: 07.04.2012]
- ALTHBAUER (2010a): **Didaktik**; <http://lexikon.stangl.eu/706/didaktik/>
 - ALTHBAUER (2010b): **Lernen**; <http://lexikon.stangl.eu/551/lernen/>
 - ALTHBAUER (2011a): **Lernstil**; <http://lexikon.stangl.eu/4469/lernstil/>
 - ALTHBAUER (2011b): **Lernstrategie**; <http://lexikon.stangl.eu/2513/lernstrategie/>
 - ALTHBAUER (2011c): **Lernziele**; <http://lexikon.stangl.eu/2077/lernziele/>
 - ALTHBAUER (2011d): **Intuition**; <http://lexikon.stangl.eu/3540/intuition/>
 - ALTHBAUER (2012a): **Lernmotivation**; <http://lexikon.stangl.eu/4496/lernmotivation-ist/>
- ARNOLD, Patricia (2005): **Einsatz digitaler Medien in der Hochschullehre aus lerntheoretischer Sicht**; <http://www.e-teaching.org/didaktik/theorie/lerntheorie/arnold.pdf> [Stand: 30.03.2012]
- BLOMERT, Peter (NN): **Benjamin Blooms Lernzieltaxonomie**; http://www.kooperatives-lernen.de/dc/netautor/napro4/appl/na_professional/parse.php?mlay_id=2500&mdoc_id=1000459 [Stand: 07.04.2012]
- BRÜGGE, Bernd (2000): **Einführung in die Informatik I; Vorlesung der Technischen Universität München**; http://www.bruegge.informatik.tu-muenchen.de/teaching/ws00/Info1/vorlesung/folien/11_Suchen_Listen.pdf [Stand: 28.03.2012]
- DÖRING, Sandra (2010): **Didaktische Handreichung - Formulierung von Lernzielen**; Sächsisches E-Competence Zertifikat; <https://www.seco-sachsen.de/index.php?id=23> --> Handreichung Formulierung Lernziele (Stand 02.03.2010) [Stand: 30.03.2012]
- FRAUENHOFER IPK (2012): **Intuitive Interaktion mit virtuellen Prototypen**; Fraunhofer Institut für Produktionsanlagen und Konstruktionstechnik; <http://www.ipk.fraunhofer.de/geschaeftsfelder/virtuelle-produktentstehung/forschung/forschungsfelder/intuitive-interaktion-mit-virtuellen-prototypen> [Stand: 01.05.2012]
- FREYHOFF, Geert (1998): **Sag es einfach! - Europäische Richtlinien für die Erstellung von leicht lesbaren Informationen**; <http://www.ibft.at/upload/sages.pdf> [Stand: 29.04.2012]
- HESSE, Friedrich W. (2012): **e-teaching.org** [Stand: 07.04.2012]
- HESSE (2011): **Lehrziele**; <http://www.e-teaching.org/didaktik/konzeption/lehrziele/>
 - HESSE (2010): **Lerntheorie**; <http://www.e-teaching.org/didaktik/theorie/lerntheorie/>
- HOLZINGER, Andreas (??): **Human-Computer Interaction – Usability Engineering im Bildungskontext** (In: Ebner, M./ Schön S. (Hrsg.) (2012): **Lehrbuch für Lernen und Lehren mit Technologien**; <http://l3t.tugraz.at/index.php/LehrbuchEbner10/index>); <http://l3t.tugraz.at/index.php/LehrbuchEbner10/article/view/71> [Stand: 07.04.2012]

QUELLENVERZEICHNIS

- ISO 13407 (1999): Deutsche Fassung – **Benutzer-orientierte Gestaltung interaktiver Systeme**; http://users-www.wineme.fb5.uni-siegen.de/home/VolkmarPipek/PUBLIC/IMuG/iso_13407.pdf [Stand: 30.04.2012]
- ISO 9241-10 (1995): Deutsche Fassung – **Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 10: Grundsätze der Dialoggestaltung**; http://interactive-quality.de/site/DE/int/pdf/ISO_9241-10.pdf [Stand: 29.04.2012]
- KELLNER, Bernhard / Rübiger, Beate (2006): **Taxonomie der kognitiven Lernziele**; Präsentation; <http://www.stangl.eu/psychologie/praesentation/lernziele.shtml> [Stand: 30.03.2012]
- LETSCHERT, Thomas (2002): Programmierung II – **Datenstrukturen; Modulunterlagen der FH Giessen-Friedberg**; <http://velociraptor.mni.fh-giessen.de/Programmierung/ProgII-htmldir/node6.html> [Stand: 28.03.2012]
- LIPINSKI, Klaus (Hrg.) (2012): **eLearning (electronic learning) - Definition**; <http://www.itwissen.info/definition/lexikon/E-Learning-eLearning-electronic-learning.html> [Stand: 10.04.2012]
- REICH, Klaus / Miesenberger, Klaus (2011): **Barrierefreiheit – Grundlage gerechter webbasierter Lernchancen** (In: Ebner, M./ Schön S. (Hrsg.) (2012): **Lehrbuch für Lernen und Lehren mit Technologien**; <http://l3t.tugraz.at/index.php/LehrbuchEbner10/index>); <http://l3t.tugraz.at/index.php/LehrbuchEbner10/article/view/34> [Stand: 07.04.2012]
- REINMANN, Gabi (2011): **Didaktisches Design – Von der Lerntheorie zur Gestaltungsstrategie** (In: Ebner, M./ Schön S. (Hrsg.) (2012): **Lehrbuch für Lernen und Lehren mit Technologien**; <http://l3t.tugraz.at/index.php/LehrbuchEbner10/index>); <http://l3t.tugraz.at/index.php/LehrbuchEbner10/article/view/18/27> [Stand: 07.04.2012]
- REY, Günter Daniel (2009): **E-Learning – Theorien, Gestaltungsempfehlungen und Forschung**; 1. Auflage; Huber Verlag; Bern; Website zum Buch: <http://www.elearning-psychologie.de> [Stand: 30.03.2012]
- REY (2009a): **Lerneigenschaften**; <http://www.elearning-psychologie.de/lernereigenschaften.html>
 - REY (2009b): **Einleitung**; http://www.elearning-psychologie.de/einleitung_theorien.html
 - REY (2009c): **Behaviorismus**; <http://www.elearning-psychologie.de/behaviorismus.html>
 - REY (2009d): **Kognitivismus**; <http://www.elearning-psychologie.de/kognitivismus.html>
 - REY (2009d): **Konstruktivismus**; <http://www.elearning-psychologie.de/konstruktivismus.html>
 - REY (2009e): **Interaktivität**; <http://www.elearning-psychologie.de/interaktivitaet.html>
- RÜDEBUSCH, Tom (2012): **Skript der Vorlesung Informatik I an der HS Offenburg**; <https://elearning.hs-offenburg.de/moodle/file.php/39/info1s12.pdf> [Stand: 28.03.2012]
- RZAZA, Jochen (Hrg.) (2012a): **Begriffsbestimmung: Was ist e-Learning?**; http://verdi-innotec.de/elearning/freie_seite.php3_hauptkategorieelearning_basisinforma~1.htm [Stand: 10.04.2012]
- RZAZA, Jochen (Hrg.) (2012b): **Vor- und Nachteile von e-Learning**; http://verdi-innotec.de/elearning/freie_seite.php3_hauptkategorieelearning_basisinform~2.htm [Stand: 10.04.2012]
- SCHOLZ, Monique (2010): **E-Learning: Klassifikation & Nachteile**; <http://www.h-age.net/hinter-den-kulissen/154-e-learning-klassifikation-a-nachteile.html> [Stand: 10.04.2012]

QUELLENVERZEICHNIS

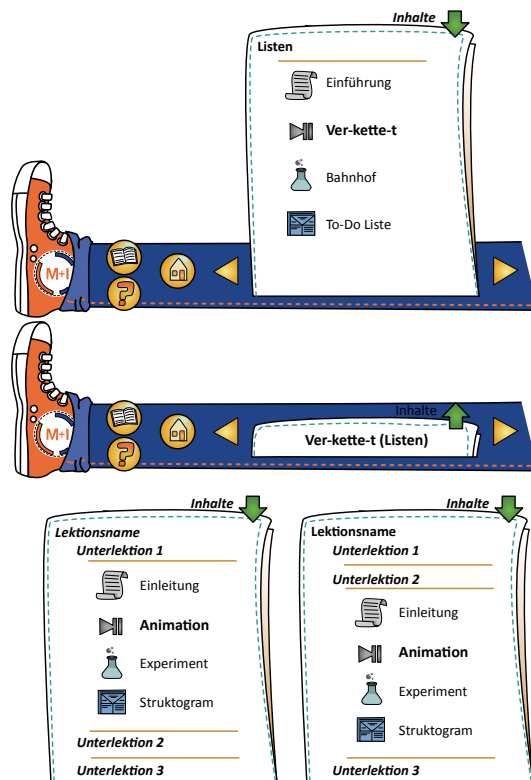
- SCHULMEISTER, Rolf (2005): **Interaktivität in Multimedia-Anwendungen**; <http://www.e-teaching.org/didaktik/gestaltung/interaktiv/InteraktivitaetSchulmeister.pdf> [Stand: 29.04.2012]
- SCHWAIGER, Petra (2009): **Informatik am Naturwissenschaftlich-technologischen Gymnasium; Staatsinstitut für Schulqualität und Bildungsforschung, München** – <http://www.isb.bayern.de/isb/download.aspx?DownloadFileID=b200b247c5f0d271dfc559b4412b6a42> [Stand: 28.03.2012]
- TSE (2012a): **Intuitive Bedienung**; Gesellschaft für Technologieberatung und Systementwicklung mbH, Hamburg; http://www.tse.de/papiere/ergonomie/softwareerg_anlagen/Bedienung.html [Stand: 01.05.2012]
- TSE (2012b): **Natürliche Symbole**; Gesellschaft für Technologieberatung und Systementwicklung mbH, Hamburg; http://www.tse.de/papiere/ergonomie/softwareerg_anlagen/Symbole.html [Stand: 01.05.2012]
- W3C (2002): **Zugänglichkeitsrichtlinien für Web-Inhalte**;
<http://www.w3c.de/Trans/WAI/webinhalt.html> [Stand: 29.04.2012]
- WCAG (2008): **Die vier Prinzipien der Web Content Accessibility Guidelines**;
<http://www.barrierefreies-webdesign.de/wcag2/> [Stand: 29.04.2012]
- WEBER, Joachim (Hrg.) (2008): **e-larning – eine Definition**; <http://www.crm2.de/?p=14> [Stand: 10.04.2012]
- VIL (2012): **E-Learning-Umgebung „Virtuelles Informatiklabor“ der Hochschule Offenburg**;
<http://mi-learning.mi.fh-offenburg.de/virtualinfolab/index.html> [Stand: 03.07.2012]

Projektstrukturplan

Kennung	Aufgabenname	Anfang	Abschluss	Dauer	Gantt Chart															
					18.3	25.3	1.4	8.4	15.4	22.4	29.4	6.5	13.5	20.5	27.5	3.6	10.6	17.6	24.6	1.7
1	Kick-Off Meeting	21.03.2012	21.03.2012	1T																
2	Anmeldung der Thesis	27.03.2012	27.03.2012	0T	◆															
3	Vorab-Recherchen	19.03.2012	30.03.2012	10T	■															
4	Idee & Konzeption	29.03.2012	04.06.2012	47T	▾															
5	Navigation/ Interaktion	29.03.2012	20.04.2012	17T	■															
6	Prototypischer Test	23.04.2012	27.04.2012	5T	■															
7	Inhalte Einstiegsseite/Lexikon	25.04.2012	08.05.2012	10T	■															
8	Inhalte Animation	07.05.2012	11.05.2012	5T	■															
9	Inhalte Experiment	14.05.2012	18.05.2012	5T	■															
10	Inhalte Struktogrammentwicklung	21.05.2012	01.06.2012	10T	■															
11	Abschluss Konzeption	04.06.2012	04.06.2012	0T	◆															
12	Umsetzung & Testen	05.04.2012	11.07.2012	70T	▾															
13	Überblick verschaffen	05.04.2012	02.05.2012	20T	■															
14	Einstiegsseite/Lexikon	15.05.2012	24.05.2012	8T	■															
15	Rahmen / Navigation und Einbinden	25.05.2012	13.06.2012	14T	■															
16	Verkettet und ICE-Bahnhof	13.06.2012	26.06.2012	10T	■															
17	Usability Test und evtl. Anpassung	26.06.2012	04.07.2012	7T	■															
18	Online stellen und debuggen	04.07.2012	11.07.2012	6T	■															
19	Abschluss Umsetzung	11.07.2012	11.07.2012	0T	◆															
20	Vorbereitung Kolloquium	02.07.2012	18.07.2012	13T	■															
21	Kolloquium	20.07.2012	20.07.2012	0T	◆															
22	Dokumentation	27.03.2012	25.07.2012	87T	▾															
23	Gliederung & Layout	27.03.2012	02.04.2012	5T	■															
24	Theorie	30.03.2012	10.05.2012	30T	■															
25	Konzeption	30.04.2012	08.06.2012	30T	■															
26	Umsetzung	18.06.2012	20.07.2012	25T	■															
27	Korrektur + Fazit	11.07.2012	25.07.2012	11T	■															
28	Abgabe der Thesis	25.07.2012	25.07.2012	0T	◆															

Dieser Projektstrukturplan wurde zu Beginn der Thesis entworfen und dann immer weiter verfeinert und an die jeweiligen Gegebenheiten angepasst. Er hat mir sehr geholfen in meinem gesetzten Zeitrahmen zu bleiben und die anstehenden Aufgaben termingerecht fertig zustellen.

Neues Navigationskonzept (eigener Entwurf)



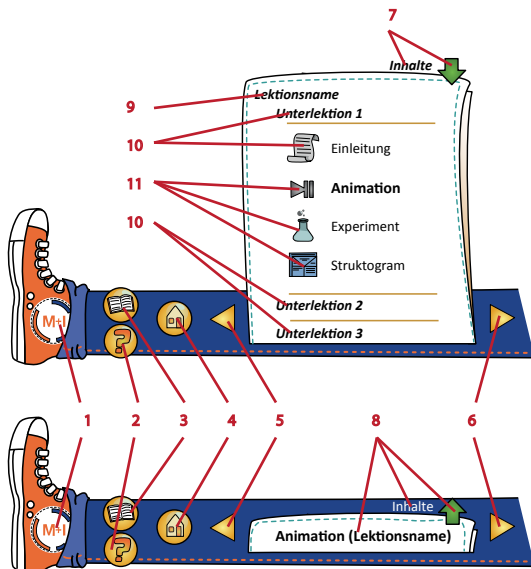
Neues Navigationskonzept / neuer Rahmen

- nur Wäschezettel
 - geändertes Symbol für Animation (bessere Erkennbarkeit)
 - keine Lernwege mehr (Lernende können selbstständig entscheiden welche Anwendungsart sie bearbeiten wollen)
- das Bein wird verlängert um alle Elemente unterzubringen
- Home-Button (zurück zum Laborraum) separat neben der Navigation
- zusammengeklappt ist nur noch der Name der aktuellen Anwendung und die Lektion bzw. Unterlektion (in Klammern) sichtbar
- zusätzlich Vor- und Zurück-Button (Pfeile) zur linearen Navigation
- Schrift: Calibri, Regular, 13pt, #000
 - > aktuelle Anwendung: Bold, 15pt
 - > Lektionsname: Bold, 13pt
 - > Mouseover: 15pt, außer aktuelles (ist nicht anwählbar)

- bei mehreren Unterlektionen ist immer nur die aktuelle sichtbar
 - > Navigation zu den anderen durch entsprechenden Lektionsnamen
 - > Unterlektionen: Bold Italic, 13pt

- der Name der Anwendung (oben im Rahmen) wird komplett angegeben mit Nennung des Lektionsnamens
 - > Aufbau: Anwendungsname (Lektionsname)
 - > Schrift: Calibri, Regular, 18pt, #FFF

ICE-Bahnhof (Listen)



Neues Navigationskonzept - Funktionsweise

Bein/Fuß:

- 1 (MI) - zurück zur MI-E-Learning Plattform
- 2 (Hilfe) - anzeigen der Hilfe für die aktuelle Anwendung/Lektion
- 3 (Lexikon) - aufschlagen des Lexikons an zur Lektion passender Stelle
- 4 (Home) - zurück in den Laborraum
- 5 und 6 - lineare Navigation durch alle Inhalte der Lektion
- 5 (Zurück) - vorhergehender Inhalt, ausgeblendet wenn erste Seite der Lektion
- 6 (Nächste) - nachfolgender Inhalt, ausgeblendet wenn letzte Seite der Lektion

Wäschezettel:

- 7 (einklappen) - klappt Navigation ein --> 8 (nur aktueller Inhalt sichtbar)
- 8 (ausklappen) - klappt Navigation aus --> 7 (Übersicht über Inhalte)
- 9 (Lektion) - Startseite der Lektion wird angezeigt, auf Navigationszettel sind nur noch die Unterlektionen sichtbar (keine ist aufgeklappt)
- 10 (Unterlektion/auf) - Startseite/ Einleitung der Unterlektion wird angezeigt, nur die jeweilige Unterlektion wird aufgeklappt (mit zugehöriger Einleitung, Animations-, Experiment- und Struktogramm-Anwendungen dieser Lektion sind sichtbar)
- 11 (Anwendungen) - Anzeigen der jeweiligen Anwendung

Screenshot der neu erstellten Listen-Einführungsseite (eigene Umsetzung)

HAST DU SCHON MAL WAS VON LISTEN GEHÖRT?
[genauer: verkettete Lineare Liste]

Einleitung (Listen)

Listen kommen zum Einsatz, wenn Daten dynamisch gespeichert werden sollen. Dynamisch bedeutet hierbei, dass die Anzahl der Elemente frei variierbar ist, auch während das Programm läuft. Im Gegenteil dazu ist z.B. bei einem Array bzw. Feld die Anzahl der Elemente festgelegt und kann, wenn das Programm läuft, nicht geändert werden.

Dieses Rohrpost-System zeigt dir den Aufbau einer typischen Liste.

Das Rohr zu Mark ist sozusagen der Startpunkt der Liste, der sogenannte „head“. Er besteht aus einem „Verweis“ (Rohr) auf den ersten „Knoten“ (Mark).

Susi ist nur zu erreichen, wenn Mark die Post weiterleiten kann. Sobald Mark nicht mehr an seinem Platz ist können Susi und damit auch Dani nicht mehr erreicht werden.

Dani, die letzte Person der Liste, kann die Post nicht weiterleiten weil von ihr kein Rohr mehr weg führt (Verweis auf „NULL“).

Das Rohr zu Susi bzw. Mark ist zu steil um die Post zurückzuschicken.

Nachdem du jetzt weißt was eine Liste ist, kannst du auf den nächsten Seiten dieser Lektion herausfinden wie eine Liste funktioniert und selbst einige Aufgaben lösen.

Screenshot der neu erstellten Listen-Lexikonseite (eigene Umsetzung)

Listen

Die erste Datenstruktur, die Lernenden in der Informatik vorgestellt wird ist der „Array“ bzw. das „Feld“. In einem Array kann eine zuvor festgelegte begrenzte Anzahl an Elementen gespeichert, und über den Index des Speicherortes wieder erreicht werden.

„Listen“ dagegen sind eine dynamische Datenstruktur. Sie können also zur Laufzeit des Programms beliebig vergrößert (neue Elemente speichern), oder verkleinert (Elemente löschen) werden und passen sich so dem jeweiligen Speicherbedarf optimal an.

Dem entsprechend werden Listen in der Praxis vor allem angewendet, wenn eine hohe Flexibilität benötigt wird. Also wenn nur wenige Elemente gespeichert, gelöscht und oft neue eingefügt werden sollen. Auch bei einer topologischen Sortierung, also wenn Dinge aufeinander aufbauen, werden Listen gerne verwendet. Z.B: wenn eine Aufgabe abgeschlossen sein muss bevor die nächste gestartet werden kann und man dann nicht mehr zur ersten Aufgabe zurückkehren kann.

Anhand der verketteten linearen Liste kann das Grundverständnis zur Funktion und Umgangsweise mit allen Listenarten erklärt werden. Auch die Grundlagen anderer Datenstrukturen, wie Bäume oder Schlange bauen auf diesen Prinzipien auf.

Aufbau und Funktion einer Liste:

Die verkettete Liste besteht aus mehreren Elementen („Knoten“), die miteinander verkettet sind. Jeder Knoten besteht aus seinen spezifischen Daten und einer Referenz bzw. einem Verweis auf ein Folgeelement. Das letzte Element der Liste wird durch den Verweis auf „NULL“ (kein Element) gekennzeichnet.

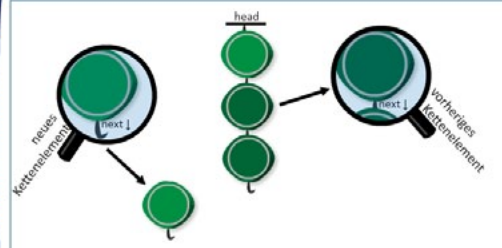
Um auf die komplette Liste zugreifen zu können, muss demnach nur das erste Element der Liste durch einen Anker-Verweis („Head“ oder „Kopf“) referenziert werden. Alle anderen Elemente können über das Navigieren über die Verweise, also das von vorne durchlaufen der Liste, erreicht werden.

Wenn z.B. das zweite Element der Liste aufgerufen werden soll, lautet die Anweisung für die Listennavigation „Gehe an den Anfang (Head) der Liste, dann gehe zum nächsten Element“. Ähnlich einer Einbahnstraße kann bei einer linearen Liste jedoch nicht zum vorherigen Element zurückgekehrt werden.

8 9

Screenshots der neu erstellten Listen-Instruktionsanwendung (eigene Umsetzung)

Ver-Kette-t (Listen)



Was möchtest du dir ansehen? **Einfügen in der Mitte**

Einfügen (vorherigesKettenelement, neuesKettenelement, head)


vorherigesKettenelement = NULL	
Wahr	Falsch
neuesKettenelement.next := head	merken := vorherigesKettenelement.next
head := neuesKettenelement	vorherigesKettenelement.next := neuesKettenelement
	neuesKettenelement.next := merken

Ausgabe head

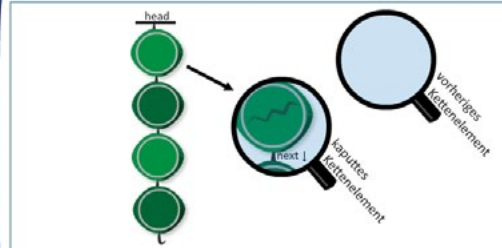
Aufgabe
Bedienung

Wie das Suchen in einer Liste und das Einfügen und Löschen eines Knotens funktionieren, kannst du mit dieser Visualisierung genau beobachten. Jedes Kettenelement hat hier einen „Status“ (kaputt oder ganz) und einen Platz, um eine Referenz auf ein anderes Kettenelement aufzunehmen („next“).

Im Struktogramm siehst du parallel zur Animation den Algorithmus, der für die ausgewählte Aufgabe verwendet wird. Du kannst also genau nachvollziehen was der jeweilige Struktogramm-Befehl bewirkt. Der Befehl "Ausgabe" beendet hierbei den Durchlauf und gibt den entsprechenden Wert zurück.



Ver-Kette-t (Listen)



Was möchtest du dir ansehen? **Löschen am Anfang**

Löschen (vorherigesKettenelement, kaputtesKettenelement, head)

vorherigesKettenelement = NULL	
Wahr	Falsch
head := kaputtesKettenelement.next	vorherigesKettenelement.next := kaputtesKettenelement.next


Ausgabe head

Aufgabe
Bedienung

Für die Animation kannst du zwischen den drei Aufgaben – Suchen, Einfügen und Löschen – wählen (Drop-Down unter der Animation).

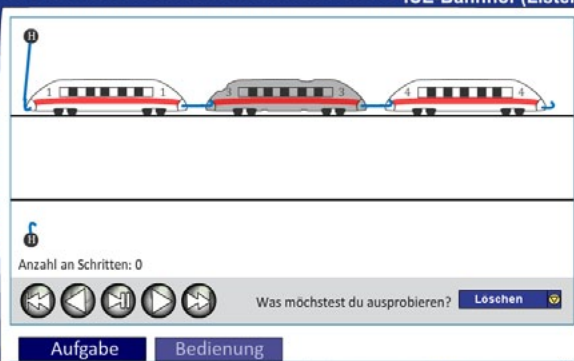
Gestartet wird die Anwendung über die Kontrollbuttons (Buttons unter der Animation). Sie bieten folgende Möglichkeiten, von links nach rechts:

- zum Beginn der Animation springen
- einen Schritt in der Animation zurück
- Play/Pause
- einen Schritt in der Animation weiter
- zum Ende der Animation springen



Screenshots des neu erstellten Listen-Experiments (eigene Umsetzung)

ICE-Bahnhof (Listen)



Aufgabe **Bedienung**

Die Züge, die in diesem Rangierbahnhof stehen, funktionieren wie eine Liste. Deine Aufgabe ist es die Ersatzwagen (unten) in den ICE (oben) einzubauen oder einen defekten Wagon (grau und mit Dellen) auszubauen.

Die Zugteile müssen dabei in möglichst wenigen Schritten und in der richtigen Reihenfolge (aufsteigend nach den Wagen-Nummern sortiert) eingebaut werden, damit die Fahrgäste sich schnell zurechtfinden.


In drei Aufgabenstellungen kannst du selbst ausprobieren wie Listen funktionieren. Wenn du dabei den Algorithmus auf der rechten Seite anwendest, findest du immer die optimale Lösung.

Lösungsidee:

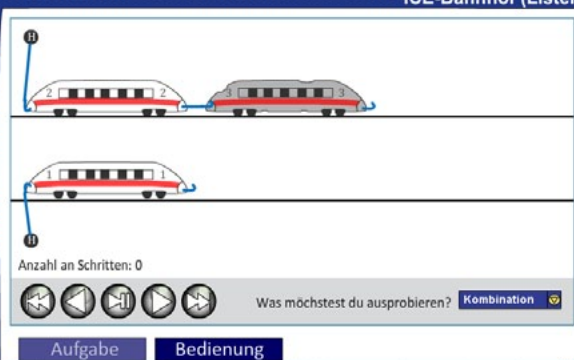
Durchlaufe die obere Wagonkette solange, bis

entweder

- „vorherige“Wagonnummer < Wagonnummer Abstellgleis und Wagonnummer Abstellgleis < „nächste“Wagonnummer dann den Wagon einfügen:
 1. „neuer“Wagon.next := „nächster“Wagon
 2. „vorheriger“Wagon.next := „neuer“Wagon
- oder
- der nächste Wagon kaputt ist dann den Wagon löschen:
 1. „vorheriger“Wagon.next := „kaputter“Wagon.next



ICE-Bahnhof (Listen)



Aufgabe **Bedienung**

In diesem Experiment stehen Dir folgende Möglichkeiten zur Verfügung um die gestellten Aufgaben zu lösen:

Alle Haken (L - jeweils am 1 oder dem Wagon-Ende befestigt) können per Drag&Drop und zu einem Wagon-Ankerpunkt (A) gezogen werden. Die verfügbaren Ankerpunkte werden dir mittels eines blauen Kreises (B) angezeigt, sobald du einen Haken ergreifst.

Diese Haken funktionieren wie Verweise in einer Liste. Sobald ein Wagon also von keinem Haken (am Anfang des Wagens) mehr gehalten wird, wird dieser gelöscht.


Wie in einer normalen linearen Liste kannst du auch hier die Verweise/Haken nach

Lösungsidee:

Durchlaufe die obere Wagonkette solange, bis

entweder

- „vorherige“Wagonnummer < Wagonnummer Abstellgleis und Wagonnummer Abstellgleis < „nächste“Wagonnummer dann den Wagon einfügen:
 1. „neuer“Wagon.next := „nächster“Wagon
 2. „vorheriger“Wagon.next := „neuer“Wagon
- oder
- der nächste Wagon kaputt ist dann den Wagon löschen:
 1. „vorheriger“Wagon.next := „kaputter“Wagon.next




Screenshots der geänderten Matroschka-Anwendung
(eigene Umsetzung/Änderung)

Matroschka (Listen)

fac(n)
 fac(5)


n > 0	
Wahr	Falsch
ergebnis := n * fac(n - 1)	ergebnis := 1
Ausgabe ergebnis	



Fakultät(n)


Aufgabe **Bedienung**

Diese Visualisierung zeigt die rekursive Berechnung der Fakultätsfunktion mit einem Struktogramm (Nassi-Shneiderman-Diagramm). Wähle aus, welche Fakultät berechnet werden soll und stelle fest wie oft die Funktion Fakultät rekursiv aufgerufen wird.



Matroschka (Listen)

fac(n)



$fac(5) = 5 * fac(4)$
 $fac(4) = 4 * fac(3)$
 $fac(3) = 3 * fac(2)$
 $fac(2) = 2 * fac(1)$
 $fac(1) = 1 * fac(0)$
 $fac(0) = 1$

Fakultät(n)



Aufgabe **Bedienung**

Für die Animation kannst du verschiedene Anzahlen von Rekursionsaufrufen wählen (Drop-Down unter der Animation).

Gestartet wird die Anwendung über die Kontrollbuttons (Buttons unter der Animation). Sie bieten folgende Möglichkeiten, von links nach rechts:

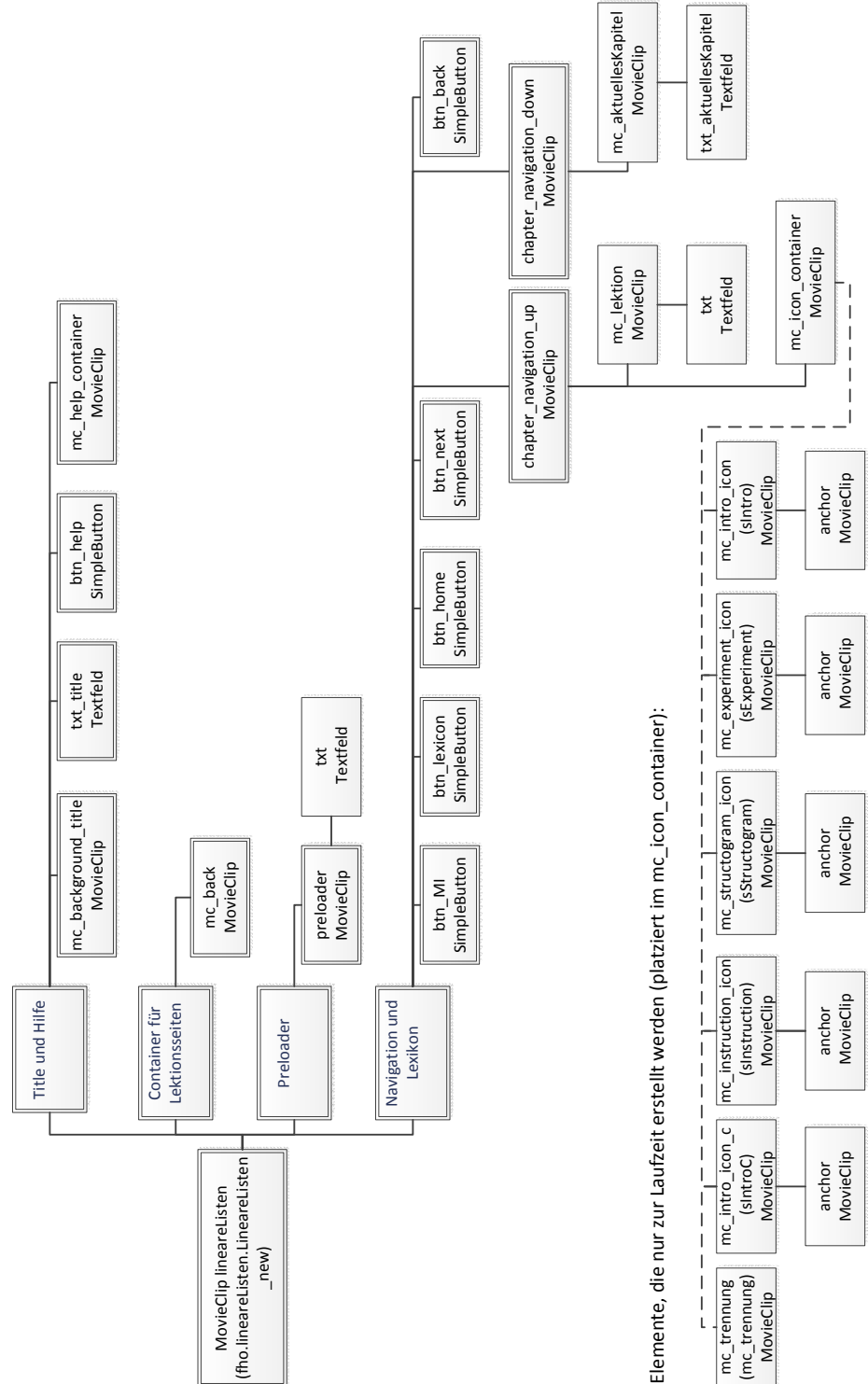
- Zum Beginn der Animation springen,
- einen Schritt in der Animation zurück,
- Play/Pause,
- einen Schritt in der Animation weiter und
- zum Ende der Animation springen.

Möchtest Du noch für einen weiteren Parameter die Fakultät berechnen? Dann klicke den Button "Zum Anfang"

Übersicht über den Aufbau der LineareListen.fla (eigene Skizzierung)

Übersicht über den Aufbau der LineareListen.fla (Navigationsstruktur)



Elemente, die nur zur Laufzeit erstellt werden (platziert im mc_icon_container):

Auszug aus neuem Code (templateApplication_new.as - eigene Ergänzungen)

```
// Aufruf der beiden neuen Text-Varianten, damit sie aufgerufen werden können
Zeile 44 description2 = new Description2(); // Aufgabe
Zeile 45 description3 = new Description3(); // Bedienung

// Aufruf des Aufgabentextes, da dieser angezeigt werden soll, wenn die Seite neu
geladen wird
Zeile 72 showDescription2();

// Funktion zur Initialisierung des Aufgabentextfeldes
Zeile 89 public function showDescription2():void {
        // Die „description2“ wird angezeigt, indem sie dem Anzeige-Objekt
        zugewiesen wird
Zeile 90 SPane.source = description2;
        // Der aktive Button wird hervorgehoben, indem der inaktive blasser wird
Zeile 91 btn_bedienung.alpha = 0.7;
Zeile 92 btn_aufgabe.alpha = 1;
        // Der bereits aktive Button kann nicht mehr geklickt werden
Zeile 93 btn_aufgabe.buttonMode = false;
        // Der inaktive Button kann geklickt werden
Zeile 94 btn_bedienung.buttonMode = true;
        // Wird der noch inaktive Button geklick, wird der Bedienungstext geladen
Zeile 95 btn_bedienung.addEventListener(MouseEvent.CLICK, function():void{
Zeile 96 showDescription3();
Zeile 97 });
Zeile 98 }

// Funktion zur Initialisierung des Bedienungstextfeldes, wie die vorrangegangene
Funktion nur mit jeweils vertauschten Inhalten
Zeile 101 public function showDescription3():void {
Zeile 102 SPane.source = description3;
Zeile 103 btn_aufgabe.alpha = 0.7;
Zeile 104 btn_bedienung.alpha = 1;
Zeile 105 btn_bedienung.buttonMode = false;
Zeile 106 btn_aufgabe.buttonMode = true;
Zeile 107 btn_aufgabe.addEventListener(MouseEvent.CLICK, function():void{
Zeile 108 showDescription2();
Zeile 109 });
Zeile 110 }
```

Mitschrieb des Usability Tests - 1. Proband (keine Listen-Vorkenntnisse)

Navigation:

- Listen besser als Rekursion (schneller zurechtgefunden, klarer Verständlich)
- Pfeile jeweils zuerst benutzt, hätte auch von sich aus keine andere Navigation benutzt/gesucht (Rekursion – Lupe nicht als Navigationsmöglichkeit verstanden / Liste – etwas länger nach Zettel gesucht, aber sofort verständlich)
- Home-Button war auch bei Listen sofort klar

Einleitung-Listen:

- Animation beim ersten mal nicht verstanden (Ende der Liste nicht klar - packt Dani Post aus oder macht er sie kaputt?)
- Laut Proband muss Animation nicht zum Verständnis sein – Text und Startgrafik reichen
- Text sehr gut verständlich, gute Textmenge, hat trotz fehlender Vorkenntnisse hier alles verstanden

Ver-Kette-t:

- Gute Idee, das Struktogramm neben Animation abläuft → sehr Verständlich
- Man kann laut Probandin jeden Schritt dank der Animation (hat verschiedene Ansichten von sich aus ausprobiert) gut nachvollziehen und bei mehr Zeit zum Verstehen der Vorgänge sicher alles Begreifen, was hier veranschaulicht war → von sich aus positive Wertung
- Einzig verwirrendes war für die Probandin, dass während der ersten Stuktogram-Schritte nichts in der Animation passiert.
- Auch die Bedienung der Anwendung wurde von der Probandin selbstständig aufgerufen und ausdrücklich gelobt (auch das die Funktionen der Buttons näher beschrieben ist – man weiß dadurch eher um was es bei der Anwendung geht) → von sich aus positive Wertung

ICE:

- Die Probandin war mit dieser Aufgabe zunächst etwas überfordert. Nach eigenen Angaben lag dies an den vielen Hinweisen, die hier gegeben werden (Aufgabe, Lösungsidee, Bedienung) ihr war nicht klar ob sie sich an die Lösungsidee halten sollte oder an die Bedienung – dieses Problem trat vor allem Aufgrund der fehlenden Vorkenntnisse in diesem Bereich auf (Sie konnte mit der Lösungsidee nichts anfangen, da sie keinerlei Kenntnisse in Java hatte)
- Die „Bedienung“ muss noch einmal überarbeitet und klarer Definiert werden (Ich habe es ihr in meinen Worten erklärt, wie es funktioniert – dann war das Vorgehen klarer) – die dort erlangten Erkenntnisse baute ich gleich nach dem Test ein, da ich selbst die Verwirrung durch diese Beschreibung verstehen konnte (ich war anscheinend zuvor zu tief in diese Anwendung versunken um eine objektive Bedienung zu schreiben)
- Nachdem Sie die Bedienung verstanden hatte und ich ihr eine Zusammenfassung der Listen-Eigenschaften gegeben hatte (was ein MI-Student in etwa in der Vorlesung darüber lernt und für diese Anwendung relevant war – da sie keinerlei Vorkenntnisse hatte) konnte sie die dort gestellten Aufgaben gut Lösen

Allgemeines:

- Sie lobte die Möglichkeiten, die Aufgabe und Bedienung getrennt präsentiert zu bekommen (dadurch werden die Texte überschaubarer und man weiß eher wo man lesen muss falls noch etwas unklar ist). Die Auswahl verschiedener Varianten (Drop-Down bei den Anwendungen) findet sie sehr gut, da somit der Inhalt aus „verschiedenen Blickwinkeln“ begriffen werden kann.
- Die Anwendungen werden generell als „gut zum Üben“ eingeschätzt und würden auf jeden Fall

zum Lernen für ein entsprechendes Fach verwendet werden. Bzw. dazu verwendet werden um die Inhalte der Vorlesung in einem anderen Umfeld (und mit anderen Beispielen als in der Vorlesung) nachzubereiten.

- Einziger Haken ihrer Meinung nach ist, dass die Lektion nur mit ein bisschen Vorkenntnissen in diesem Bereich (Informatik-Listen) verwendbar ist und somit für z.B. Schüler zum Lernen eher ungeeignet ist.

Mitschrieb des Usability Tests - 2. Proband (gute Listen-Kenntnisse)

Navigation:

- Listen besser als Rekursion (schneller zurechtgefunden, klarer Verständlich)
- Pfeile jeweils zuerst benutzt, hätte auch von sich aus keine andere Navigation benutzt/gesucht (Rekursion – verstanden aber lange gesucht, verwirrend dass es zwei Lernwege gibt, hätte mit einer Lupe eher eine Suche vermutet/ Liste – gleich gefunden, sofort verständlich, auch Pfeile sind hier verständlicher da mit Tool-Tipp unmissverständlich)
- Home-Button war bei Listen sofort klar

Einleitung-Listen:

- gut verständlich aber ohne jede Vorkenntnisse evtl. schwerer
- Text und Grafik reichen, Textmenge gut, gut dass Dynamik erklärt wird
- Anmerkung: In Listen wird nichts transportiert wie in der Rohrpost → kann evtl. zu Verwirrung führen (Animation)

Ver-Kette-t:

- Gute Idee, dass Struktogramm neben Animation abläuft → sehr Verständlich (jeder einzelne Schritt kann im eigenen Tempo durchgegangen werden → Zeit um alles zu verstehen → alles auf einen Blick, es motiviert sich mit dem Algorithmus auseinander zu setzen) → von sich aus positive Wertung
- Auch gut, dass alles auf einen Blick sichtbar ist (kein Hilfefenster, das zusätzlich aufgeht → alles zusammen und übersichtlich)
- Proband hat verschiedene Ansichten von sich aus getestet
- Auch die Bedienung der Anwendung wurde vom Proband selbstständig aufgerufen
- Algorithmus und Visualisierung bieten guten Überblick über Möglichkeiten einer Liste

ICE:

- dank Bedienungsanleitung klar & schnell verstanden (konnte Aufgaben sofort lösen, Konzept sehr schnell verstanden)
- Wünscht sich größere Fläche zum klicken (Haken ab und zu „fremdarbeit“), und besser mit Rückmeldung wann man über einem Kreis ist und loslassen kann (→ alles sehr sinnvoll, wurde sofort umgesetzt)
- gut, mit Drag&Drop zu arbeiten → macht die ganze Anwendung spannend und man hat eher das Gefühl etwas zu lernen wenn man es selbst ausprobieren kann (Anwendung macht Spaß)
- Gut dass Bedienungsanleitung mit Bildern beschrieben → von sich aus positive Wertung

Allgemeines:

- zur Nachbereitung der Info-2 Vorlesung gut geeignet
- alles gut erklärt und verständlich

Mitschrieb des Usability Tests - 3. Proband (Listen-Grundkenntnisse)

Navigation:

- Listen besser als Rekursion (schneller zurechtgefunden, klarer Verständlich)
- Pfeile jeweils zuerst benutzt, hätte auch von sich aus keine andere Navigation benutzt/gesucht – Pfeile reichen nach ihrer Meinung (Rekursion – verstanden aber lange gesucht, verwirrend das es zwei Lernwege gibt/ Liste – gut gefunden, sofort verständlich, auch Pfeile sind hier verständlicher da mit Tool-Tipp)
- Home-Button war auch bei Listen sofort klar

Einleitung-Listen:

- gut verständlich aber ohne jede Vorkenntnisse evtl. schwerer
- Text und Grafik reichen, Textmenge gut, gut das Dynamik erklärt wird
- Animationen sind eigentlich immer positiver als reine Grafiken, da in diesem Fall (Rohrpost) aber kein zusätzlicher relevanter Inhalt durch die Animation vermittelt wird kann diese weggelassen werden.

Ver-Kette-t:

- Verständlich, gut das beides parallel zueinander gezeigt und abgespielt wird, gut das in der Bedienung alles erklärt wird → nimmt die Unsicherheit was alles möglich ist und gibt besseren Einblick in die verschiedenen Möglichkeiten der Anwendung (wäre sonst nie auf die Idee gekommen, dass hinter den Struktogrammpunkten Tool-Tipps sind)

ICE:

- dank Bedienungsanleitung klar & schnell verstanden (konnte Aufgaben sofort lösen, Konzept sehr schnell verstanden)
- evtl. etwas viel Text zum durchlesen, aber alles nötige Erklärungen, darum ok
- Interaktion regt dazu an sich mit dem Thema zu beschäftigen

Allgemeines:

- zur Nachbereitung der Info-2 Vorlesung gut geeignet
- Sie selbst bewertete das VIL gut, hatte allerdings Probleme ihre schlechte Meinung von der Informatik auch mit diesem E-Learning abzulegen

Mitschrieb des Usability Tests - 4. Proband (keine Listen-Vorkenntnisse)

Navigation:

- Listen besser als Rekursion (schneller zurechtgefunden, klarer Verständlich)
- Pfeile jeweils zuerst benutzt, hätte auch von sich aus keine andere Navigation benutzt/gesucht – Pfeile reichen nach seiner Meinung (Rekursion – lange gesucht und nicht verstanden das man mit der Lupe navigieren kann, verwirrend das es zwei Lernwege gibt/ Liste – gut gefunden, sofort verständlich, auch Pfeile sind hier verständlicher da mit Tool-Tipp)
- Home-Button war auch bei Listen sofort klar

Einleitung-Listen:

- gut verständlich, Text und Grafik reichen zum Verständnis, zusätzliche Animation verwirrt etwas, da für den Probanden hier keine neuen Erkenntnisse vermittelt werden und auch im Text nichts mehr dazu steht

Ver-Kette-t:

- Verständlich, gut das beides parallel zueinander gezeigt und abgespielt wird, gut das in der Bedienung alles erklärt wird (auch hier Struktogrammpunkte - Tool-Tipps sonst nicht vermutet)

ICE:

- dank Bedienungsanleitung klar & schnell verstanden (hatte anfangs noch etwas Verständnisprobleme, da ihm das Listen-Konzept bis dahin nicht bekannt war, nach ein paar versuchen hat es aber gut geklappt)
- gut erklärt und darum auch gut nachvollziehbar – Anwendung macht Spaß, deshalb hat der Proband von sich aus auch ohne Vorkenntnisse versucht die richtige Lösung zu finden / hat alle gefunden und die Lösungsidee dazu verstanden)

Allgemeines (Abschlussfragen):

- gut zum lernen, durch viele Interaktionsmöglichkeiten Interessant zum ausprobieren (ICE) oder alle Möglichkeiten zu begreifen (Kette)
- Bewertete das VIL gut, mit der Anmerkung, das es für viele schwierig sein wird nur auf dieser Grundlage ein Thema zu lernen – nach Ansicht dieses Probanden ist es wichtig Vorlesungen zu diesen Themen zu besuchen und die Themen dann mittels E-Learning nachzubereiten (Vorlesungen sollten nicht ersetzt werden)