ionesoft

comem+

Blind am Computer

Erarbeitung und Publikation eines Empfehlungskatalogs zur Erstellung von barrierefreien Desktop-Applikationen mit dem Praxisbeispiel beook

Travail préparatoire de Bachelor 2015

Berne, 29 mai 2015

Christian Heimann
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud - HEIG-VD
Département comem+
Filière Ingénierie des Médias
Classe MIT41

Répondant interne
Jean-Pierre Hess

Avenue des Sports 20
CH-1400 Yverdon-les-Bains
www.comem.ch

Hes·so

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

Bachelor Vorarbeit – TPB | Christian Heimann

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

comem+

[illegible]

[illegible]

[illegible]

Danksagung

Eine ausführliche Danksagung folgt in der Bachelor Arbeit (TB). Für diesen ersten Teil möchte ich aber trotzdem bereits allen herzlich Danken, die wichtige Mosaik-Steinchen und -Steine beigetragen haben:

Jean-Pierre Hess, Daniel Stainhauser, Céline Acker, Selamet Aydogdu, Ouli-Minna Elgorriaga, Urs Hiltebrand, Robert W. Kingett, Anton Bolfing, Luc Fontolliet, Jean-Marc Meyrat, Saskia Faulk und Yves Germanier – Vielen Dank!

Inhaltsverzeichnis

Einleitung

Rahmen

Dieses Dokument ist die Vorarbeit zur effektiven Bachelor-Arbeit im Rahmen meines Studiums zum Medien-Ingenieur comem+ an der heig-vd (Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud), Yverdon-les-Bains. Das globale Thema ist «Accessibility für digitale Lehrmittel», das begleitende Unternehmen ist die ionesoft GmbH, Bern.

Problemstellung

Blinde und visuell eingeschränkte Personen gehen bei der Bedienung von Computerprogrammen meist andere Wege als der normalsehende Standard-User. Da der visuelle Input oft ganz wegfällt, muss der bestehende Inhalt über andere Wege vermittelt werden. Auch das von vielen Leuten bevorzugte Navigationsmittel, die Maus, fällt für diese Gruppe von Anwendern weg. Deshalb muss man in einer barrierefreien Applikation über die Tastatur an jede Funktion sowie an den kompletten Inhalt gelangen können – und das auf möglichst effiziente und logische Weise.

Dank akustischen Hinweisen und der Sprachausgabe von Screenreadern ist eine Verwendung von Computern für die Zielgruppe[1] überhaupt erst möglich. Doch ein installierter Screenreader, genügt nicht. Die Applikationen müssen für Screenreader «lesbar» sein.

Die Applikation *beook* der ionesoft GmbH ist eine in der Schweiz und Deutschland verbreitete Plattform für digitale Lehrmittel. Doch damit auch die Zielgruppe von der Digitalisierung profitieren kann, muss die Applikation barrierefrei werden. Das ist bei einer Applikation wie *beook*, die einen respektablen Funktionsumfang aufweist, eine Herausforderung.

Ziele dieser Vorarbeit

Um für die technische Umsetzung im Anschluss an diese Vorarbeit die genauen Bedürfnisse, Erwartungen und Mängel der *beook*-Applikation zu kennen, wurden folgende Ziele gesetzt:

- Standard-Navigations-Schemen aufzeigen
- Analyse von Belegung und Verwendung der Shortcuts und Accesskeys in Standard-Programmen und *JAWS*
- Analyse der aktuellen *beook*-Applikation betreffend Barrierefreiheit: Was ist bereits zugänglich? Was funktioniert bereits?
- Erarbeiten und veröffentlichen eines Empfehlungskataloges mit Navigations-Schema und Accesskeys, aber auch mit inhaltlichen Anpassungsempfehlungen.

1 siehe «Anmerkungen» auf Seite 4

Es gibt zwar im Internet viele Seiten mit theoretischen Hinweisen, konkrete, beschriebene Beispiele aber fehlen. Daher macht es Sinn, den Empfehlungskatalog auf einer anerkannten Plattform für Wissen im Bereich «Accessibility» zu veröffentlichen, um ihn einer breiten Öffentlichkeit zugänglich zu machen – möglichst barrierefrei.

Methodik

Für diese Vorarbeit wurde in etwa gleichmässigen Teilen mit folgenden Quellen und folgendem Erfahrungsschatz gearbeitet:

- Experten-Interviews
- Literatur-Recherche im Internet
- Selbstversuch mit Augenbinde und Screenreader *JAWS*

Anmerkungen

Werden Personenbezeichnungen aus Gründen der besseren Lesbarkeit lediglich in der männlichen oder weiblichen Form verwendet, so schliesst dies das jeweils andere Geschlecht mit ein.

Um die Wiederholung des Begriffs «blinde und visuell eingeschränkte Personen» zu vermeiden, wird in dieser Arbeit von «Zielgruppe» gesprochen. Diese Zielgruppe besteht nicht nur aus Personen mit Geburtsgebrechen, sondern auch all jenen – und das ist die Mehrheit – welche eine Sehbehinderung erst im Laufe der Zeit entwickeln oder deren Sehkraft mit zunehmendem Alter nachlässt.

Dass blinde und sehbehinderte Menschen nicht die einzige zu beachtende Gruppe sind, wenn es um das Thema «Barrierefreiheit» geht, soll hier vermerkt sein. Auf weitere Typen von Handicap wird aber in dieser Arbeit nicht eingegangen. Diese würden zusätzliche Lösungen und Mittel erfordern, um das gleiche Produkt für wirklich alle barrierefrei zu machen.

Als Beispiel-Person wird für diese Arbeit, wo nichts anderes vermerkt ist, von einem vollblinden Anwender ausgegangen. Als «Normal-Benutzer» wird ein Anwender bezeichnet, der ohne weitere technische Hilfsmittel einen Computer über die Elemente Bildschirm, Tastatur und Maus bedienen kann.

Die in diesem Dokument erwähnte *beook*-Applikation beschreibt die Version für Windows, welche auf der Basis von Eclipse RCP läuft. Die Versionen für iOS und Android wurden nicht berücksichtigt. Mit dem hier erwähnten Betriebssystem Windows ist Windows 7 gemeint.

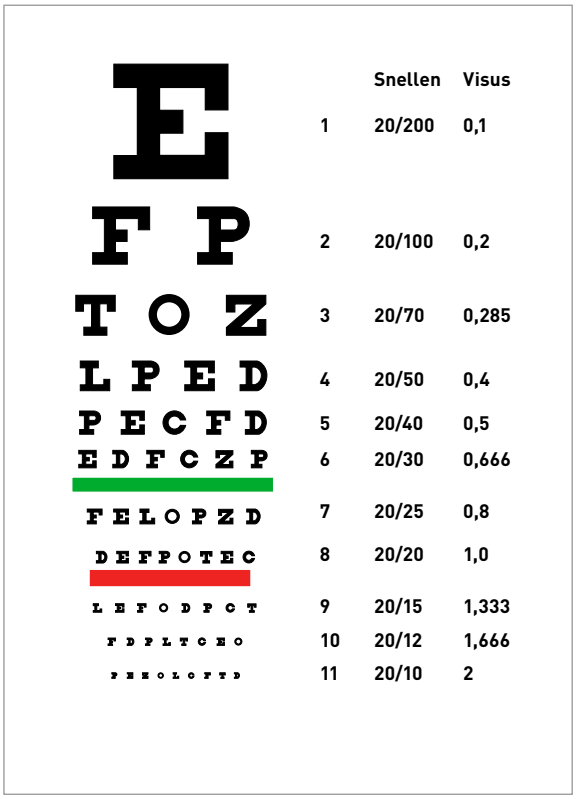
1. Blind am Computer arbeiten

1.1. Blind, sehbehindert, sehgeschädigt – eine Definition

Nur schon der Begriff «Behinderung», welcher auch Sehbehinderte mit einschliesst, wird weder im Volksmund noch in der Fachwelt einheitlich gehandhabt. Im Sehbehindertenwesen wird darauf hingewiesen, dass Behinderung aus dem Zusammenspiel von drei Faktoren entsteht:[2]

- Gesundheitszustand: Entwicklungsprobleme oder Krankheiten
- Umweltfaktoren baulicher, sozialer und organisatorischer Art
- Personale Faktoren: persönliche Einstellung und Einstellung von Mitmenschen

Dies muss verstanden sein, um eine Behinderung nicht allein als Ergebnis einer biologischen Schädigung zu betrachten.[3]



	Snellen	Visus
1	20/200	0,1
2	20/100	0,2
3	20/70	0,285
4	20/50	0,4
5	20/40	0,5
6	20/30	0,666
7	20/25	0,8
8	20/20	1,0
9	20/15	1,333
10	20/12	1,666
11	20/10	2

fig. 1 – «Snellen chart» von Jeff Dahl, http://commons.wikimedia.org/wiki/File:Snellen_chart.svg#/media/File:Snellen_chart.svg

Für die Weltgesundheitsorganisation (WHO) gilt als blind, wer einen Visus kleiner als 0,05 hat.[4] Visus bezeichnet die Sehschärfe, bzw. die Fähigkeit, mit den Augen Konturen und Muster wahrzunehmen. Zum Vergleich: bei einem 20-jährigen «normal sehenden» Menschen liegt der Visus zwischen 1,0 und 1,6, bei einem 80-jährigen zwischen 0,6 und 1,0.[5] Die bekannte Sehprobetafel, welche vom niederländischen Ophthalmologen Herman Snellen entwickelt wurde, hilft, diese Grössen besser zu verstehen *(fig. 1)*.

Die Werte zeigen auch, dass eine Abnahme der Sehkraft im Alter natürlich ist. Folgende Zahlen verdeutlichen dies: Knapp 2 Prozent aller Personen bis 40 Jahre sind sehbehindert oder blind, bei den 40- bis 59-jährigen beträgt der Anteil 3,7 Prozent, bei den 60- bis 79-jährigen 8,9 Prozent und bei den über 80-jährigen 20,5 Prozent.[6] In der Schweiz liegt die Zahl der Sehbehinderten ungefähr bei 310'000[7], die Zahl der Blinden wird auf ca. 10'000[8] geschätzt.

Sehbehindert, sehgeschädigt und visuell Eingeschränkt sind drei Begriffe, die das gleiche beschreiben: ein vermindertes Sehen, also mit Visus-Werten zwischen «normal» und «blind». Es existieren zwar Unterscheidungskrite-

2 Stefan Spring – Sehbehinderung und Blindheit: Entwicklung in der Schweiz, Seite 6
3 vgl. Stefan Spring – Sehbehinderung und Blindheit: Entwicklung in der Schweiz, Seite 6
4 Sehbehinderung und Blindheit: 10 Fragen und Antworten – www.sbz.ch > Wissen
5 Sehschärfe, Wikipedia
6 Sehbehinderung und Blindheit: 10 Fragen und Antworten – www.sbz.ch > Wissen
7 Sehbehinderung und Blindheit: 10 Fragen und Antworten – www.sbz.ch > Wissen
8 Stefan Spring – Sehbehinderung und Blindheit: Entwicklung in der Schweiz, Seite 10

rien zwischen «leichter» und «schwerer» Sehschädigung, doch keine Erhebung stellt diese Unterschiede für die Schweiz exakt dar.[9]

Die Unterscheidung zwischen blind und sehbehindert ist wichtig, denn nicht alle Hilfsmittel zur Bewältigung des Alltags sind für beide Gruppen geeignet. Viele Hilfsmittel werden entweder für «blind-nichtsehende» oder für «sehbehindert-sehende» Menschen konzipiert.[10] Das schliesst nicht aus, dass Hilfsmittel beiden Gruppen gleichzeitig dienen können: Dies ist zum Beispiel der Fall bei Screenreadern.

1.2. Gängiges Hardware Setup

Das «Standard»-Dispositiv, das eine blinde Person zur Verwendung eines Computers benötigt, besteht aus folgenden Elementen:

- Computer/Laptop mit Tastatur
- Screenreader-Software, die auch die Braillezeilen-Textausgabe unterstützt *(fig. 2)*
- Braillezeile *(fig. 3)*

fig. 2 – Logo vom JAWS-Screenreader, extrahiert von «JAWS for Windows 20th Anniversary Video», https://www.youtube.com/watch?v=0DYjkF59jeo

fig. 3 – Braillezeile, die via USB am Computer angeschlossen ist, mit Achtpunkt-Braille Buchstaben, 8-Punkt-Braille-Eingabe und Steuertasten

Screenreader und Braillezeile wirken ergänzend, nicht identisch. Sie ergänzen die Ausgabe gegenseitig. So ist die Braillezeile zum Beispiel immer fixiert auf den Ort, wo der Fokus liegt. Passiert etwas ausserhalb des Fokus, kann dies in jenem Moment nur vom Screenreader mitgeteilt werden. Umgekehrt ist es eher schwierig zu sagen, ob gemäss Sprachausgabe ein Wort korrekt geschrieben wurde oder nicht. Um die Rechtschreibung zu überprüfen ist eine Braillezeile unabdingbar.

Eingaben werden über die Tastatur oder die Steuertasten der Braillezeile gemacht. Weitere Möglichkeiten zur Befehls- und Texteingabe existieren, werden in dieser Arbeit nicht behandelt.

Visuell eingeschränkten Personen stehen durch die vorhandene Sehkraft je nach Einschränkung noch weitere Hilfsmittel zur Bedienung von Computern zur Verfügung:

- Computer-Maus
- Bildschirmlupen *(fig. 4 und fig. 5)*
- Hoch-Kontrast Farbschemen *(fig. 6)*

9 Stefan Spring – Sehbehinderung und Blindheit: Entwicklung in der Schweiz, Seite 7
10 Stefan Spring – Sehbehinderung und Blindheit: Entwicklung in der Schweiz, Seite 7

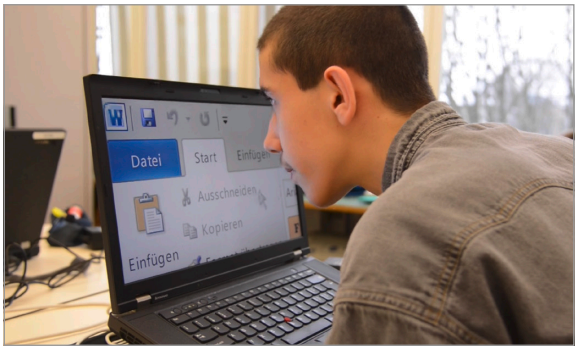


fig. 4 – Aktive Bildschirmlupe, erzeugt im Bild mit der Software MAGic

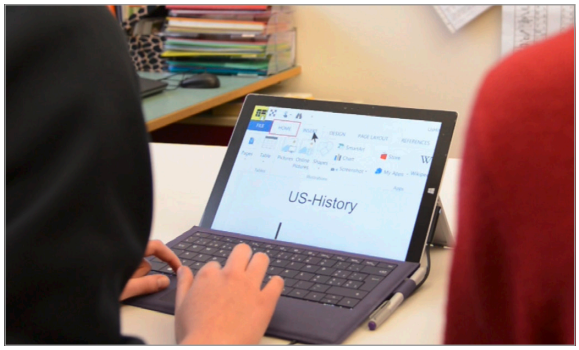


fig. 5 – Aktive Bildschirmlupe, erzeugt im Bild mit der Software ZoomText

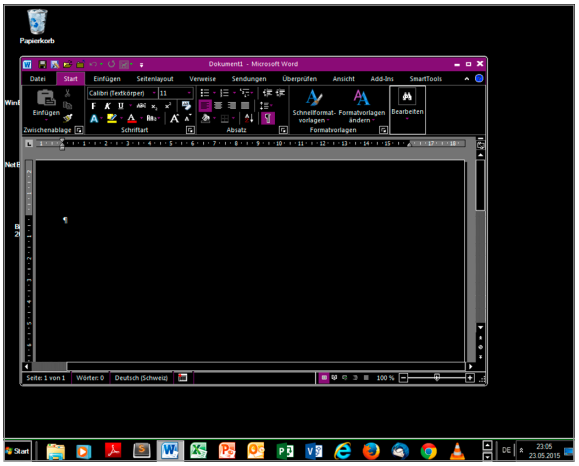


fig. 6 – Eines der Standard-Hochkontrast-Farbschemen in Windows, im Bild mit geöffnetem Word-Dokument.

In der Schweiz ist die gängigste Hardware-Ausstattung für die Zielgruppe die folgende:

- Windows-Computer mit Tastatur oder Laptop
- *JAWS*-Screenreader
- Braillezeile verschiedener Hersteller

Der Grund, warum das «Standard»-Dispositiv in der Schweiz sehr einheitlich ausfällt, liegt nicht an fehlenden Alternativen. Aufgrund des kleinen Marktes, ist die Accesstech AG in der Schweiz derzeit der führende IT-Hilfsmittel-Lieferant. Um möglichst effizient arbeiten und alle Kunden gut betreuen zu können, ist die Reduktion auf eine Auswahl gut funktionierender Technologien nötig.

JAWS, der Screenreader der Firma *Freedom Scientific* ist mit einer Installationsquote von ca. 95 % unter Blinden und Sehbehinderten der am weitesten verbreitete Screenreader der Schweiz.[11] Aufgrund der hohen Anschaffungskosten und der wiederkehrenden Upgrade-Kosten bei System-Updates scheint sich auf dem globalen Markt mehr und mehr die Gratis-Screenreader-Software von *NVDA* durchzusetzen. Diese wird bereits heute von 43 % der Benutzer verwendet – im Gegensatz zu weltweit 64 %, die mit *JAWS* arbeiten (und zum Teil mehrere Screenreader nutzen).[12] *NV Access*, das Unternehmen hinter *NVDA* wird unter anderem unterstützt von Adobe und Google.[13]

Windows war von Beginn an für Blinde zugänglich, jedoch waren die Blinden nur ein «Nebenprodukt»: Alle grafischen Benutzeroberflächen (GUI) wurden zugänglich gemacht, um diese automatisiert zu testen.[14]

11 Interview mit Urs Hiltebrand, Accesstech – Frage «Welches sind die am häufigsten verwendeten Screenreader in der Schweiz?»

12 Screen Reader User Survey #4 Results – WebAIM Web Accessibility in Mind – Home > WebAIM Projects – Frage «Which of the following desktop/laptop screen readers do you commonly use?»

13 Support Us, Partner with Us – NV Access, Home of the free NVDA screen reader – www.nvaccess.org > Support Us

14 Interview mit Urs Hiltebrand, Accesstech – Frage «Gibt es noch ‹Nice-to-knows› zum Thema?»

1.3. Beobachtungen

1.3.1. Einblick in die Praxis

Sich nicht nur von schönen Beschreibungen und Theorien leiten lassen, sondern in der Praxis erleben, wie gearbeitet wird, war ein wichtiger Punkt dieser hier vorliegenden Analyse. Durch das Beobachten von mehreren blinden und visuell eingeschränkten Benutzern, wie auch durch den Austausch mit ihnen und ihren Betreuern, konnten die hier folgenden Erkenntnisse gewonnen werden.

Wenn Sie das nächste Mal vor Ihrem Laptop sitzen, stellen Sie sich vor: Alles was Sie sehen, wäre auf einer einzigen, langen Textzeile beschrieben. Alle möglichen Aktionen und vorhandenen Informationen. Nicht einfach, da die Übersicht zu behalten. Für Blinde ist die Bedienung eines Computers genau so. Das zwingt sie, sich «durchzutasten». Wie sie es im Leben machen, so auch am Computer – natürlich machen es Geübte absolut gekonnt. Für Blinde ist die Bedienung über die Tastatur der gangbarste Weg. Sie suchen sich über Schritt-für-Schritt-Navigation den Weg zu den gewünschten Menüs. Der Screenreader liest vor, um welche Elemente es sich handelt: Ist es ein Eingabefeld, ein Schalter, eine Liste, eine Überschrift usw. Er erwähnt auch, welche Kurztasten gewählt werden könnten und gibt Zusatzinformationen zu Elementen wie Aktionen oder Menü-Punkten. Aber: das geht nur, wenn die Software für Screenreader auch lesbar ist.

1.3.2. Grundnavigation: Hierarchien mit mehreren Niveaus

Da dies nicht der Fall ist für jede Applikation, beschränkt sich die Programm-Palette der Zielgruppe leider oft auf ein paar Standards wie die Microsoft Office Suite, Internet Explorer oder Firefox und andere spezielle barrierefreie Programme um die restlichen Bedürfnisse zu decken. Die «Grundnavigation» geht zu einem grossen Teil über die altbekannte Menüleiste.

Für Blinde sind Strukturen mit mehreren Niveaus von Vorteil.[15] So fordert die Navigation über drei oder vier Hierarchiestufen zwar ein abstrakteres Denken, aber es hilft ihnen, sich schneller fortzubewegen. Zugunsten der Sehenden geht der Trend derzeit aber zu Benutzeroberflächen und Menüs mit flachen Hierarchien. Zum Beispiel hat die Einführung von Menü-Bändern in der Microsoft Office-Palette eine Verschlechterung für die Blinden bedeutet.[16] Das weite Ausbreiten von Befehlen zwingt zum «Durchtabben».

1.3.3. Programmschulung

Man darf nicht vergessen: Wie jeder Anwender muss die Zielgruppe nicht nur auf den speziellen Accessibility-Features geschult werden, sondern auch auf dem Programm selbst. Da der optische Input ganz (oder zumindest zu einem grossen Teil) fehlt, ist ein einfaches «Try and Error» nicht möglich. Bis die Zielgruppe also eine Vorstellung im Kopf hat, wie das Programm überhaupt funktioniert, wozu es da ist und was man damit machen kann, ist es ein grösserer Aufwand nötig als beim «Normal-Benutzer». Das erklärt

15 Interview mit Urs Hiltebrand, Accesstech – Frage «Was sind Hindernisse in der aktuellen Software?»
16 Besuch an der Schule für Sehbehinderte Zürich – Aussage Edith Haller

auch, warum die Klassengrösse an der Schule für Sehbehinderte Zürich für den IT-Unterricht bei maximal zwei Schülern liegt:[17] In Momenten in denen die Schüler anstehen, ist ein schnelles Helfen gefragt, um die Motivation hoch zu halten. Selbsthilfe ist oft schwierig bis unmöglich.

1.3.4. Verlässlichkeit

Für die ganze IT-Umgebung ist das Element «Verlässlichkeit» ein Schlüsselelement. Die Zielgruppe muss sich darauf verlassen können, mit den gleichen Befehlen stets das gleiche Resultat zu erzielen. Unaufgefordert auftauchende Popups und Menüs, Verspringen des Fokus und andere Bugs können dazu führen, dass sich der Benutzer komplett verliert. Über eine Standard-Taste – in der Regel **[esc]** – muss man immer zurück auf die Anfangsstellung kommen können. So kann sich der Benutzer wieder orientieren und sich in der Not aus einer misslichen Situation befreien.

1.3.5. Ansprüche

Selbst wenn die Zielgruppe froh ist um die Hilfsmittel, welche ihnen heute zur Verfügung stehen, ist der Anspruch an die Performance ebenso hoch wie beim «Normal-Benutzer». Kleinste Verzögerungen die ständig auftreten, zum Beispiel wegen dem Rendering von Bildschirmlupen, wirken störend für einen guten Arbeitsfluss.

Da die Tastatur das Primär-Eingabeinstrument ist, muss die Bedienbarkeit via Tastatur ebenfalls höchsten Ansprüchen genügen. Erstens soll die Bedienung schnell und effizient erfolgen können. Dies kann auch sehenden Power-Usern zugute kommen. Zweitens sollen möglichst die gängigen Konventionen berücksichtigt werden – definierte Standards gibt es nicht.

Auch wenn die Ansprüche zwischen Blinden und visuell Eingeschränkten unterschiedlich sind: Im Bezug auf die Nutzung des Computers und von Applikationen kann man festhalten, dass die Sprachausgabe, die einer blinden Person die Nutzung ermöglicht, auch für Sehbehinderte eine wichtige Hilfe ist. Daneben sind aber auch unterschiedliche Elemente wichtig: Für Blinde muss der Kontext in der Sprachausgabe erwähnt werden, für Sehbehinderte ist das saubere Setzen des Fokus von grösserer Wichtigkeit, damit die Bildschirmlupe diesem auch folgen kann.

Eine Aussage von Urs Hiltebrand beschreibt eine weitere, wichtige Beobachtung: «Für ein optimales Arbeiten mit dem Computer sind immer die drei Elemente ausschlaggebend: Applikation, Screenreader und Benutzer respektive dessen Fähigkeiten, einen Computer zu bedienen. Die wenigsten User können ihren Screenreader bis ins letzte Auskosten. Wäre dies der Fall, dann würde das auch schon viele der derzeit bestehenden Probleme lösen.»[18]

17 Besuch an der Schule für Sehbehinderte Zürich – Aussage Ouli-Minna Elgorriaga

18 Interview mit Urs Hiltebrand, Accesstech – Frage «Gibt es noch ‹Nice-to-knows› zum Thema?»

1.4. Internet-Nutzung

Die Nutzung des Internets ist trotz den technischen Möglichkeiten in den letzten Jahren nicht viel zugänglicher geworden, so die Empfindung der Zielgruppe.[19] Die grössten Schwierigkeiten liegen bei Flash-basierten Websites und generell unsauber strukturierten Seiten.

Stellt man sich eine Website als «nacktes» *HTML* vor, sprich reine Semantik ohne grafischen Input und ohne css, dann hat man, was Screenreader zum Lesen nehmen. Ist diese Semantik ein Durcheinander, dann gibt ein Screenreader auch ein Durcheinander aus. Daher sind unsauber strukturierte Websiten ein grosses Problem. Bereits eine einfache, gute Struktur würde in vielen Fällen zu einem besseren Zugang führen.

Flash-basierte Webseiten sind oft eine «Blackbox» für Screenreader. Da es sich um eingebettete Komponenten handelt, sind die semantischen Elemente, welche den Inhalt und die Struktur beschreiben würden, meistens nicht greifbar.

Barrierefrei konzipierte Webseiten weisen heute oft als semantische Zusatzinformation *ARIA*-Regions auf. Diese beschreiben Zonen wie «Navigation», «Inhalt» oder «Fusszeile» und dienen so einer raschen Navigation, da dem Benutzer sofort klar ist, wo er sich befindet. Er hat auch mit *JAWS* die Möglichkeit, durch einen Tastenbefehl von Region zu Region zu springen.

Wie das Springen von Region zu Region sind auch das Springen von Titel zu Titel, von Link zu Link etc. über eine Taste möglich. Mehr dazu unter «2.4. Internet Explorer/Browser».

1.5. Relevanz von Tastenbefehlen

1.5.1. Unterschied Tastenkombination – Kurztaste

Wenn man von Tastenbefehlen spricht, ist folgende Unterscheidung wichtig:

- Tastenkombinationen, auch Shortcuts genannt, sind Kombinationen wie **[ctrl + p]**. Diese können in der Regel von jedem Ort im Programm aus aufgerufen werden und lösen immer den gleichen Befehl aus.
- Kurztasten, auch Mnemonics oder Accesskeys genannt, sind einzelne Alphanumerische Tasten, die durch drücken zu einem bestimmten Menü-Punkt der aktuellen Ebene führen, wobei die gleiche Taste in unterschiedlichem Kontext zu einem unterschiedlichen Element führt. In der Browser-Navigation nehmen gewisse Kurztasten ganz bestimmte Funktionen ein.

[ctrl + s] – Befehl «Speichern» ist wohl einer der geläufigsten und gebräuchlichsten Tastenkombinationen, die «Normal-Benutzer» verwenden. Auch **[ctrl + c]** und **[ctrl + v]** sind noch vielen bekannt. Doch viele Nicht-Power-User steuern das Programm und dessen Funktionen meistens über die grafische Benutzeroberfläche. Dabei könnte man viele Menüs und Funktionen

19 Screen Reader User Survey #4 Results – WebAIM Web Accessibility in Mind – Home > WebAIM Projects – Frage «Web Accessibility Progress»

mit Tastenkombinationen aufrufen. Nutzen denn Blinde diese definierten Shortcuts?

1.5.2. Shortcut-Verwendung

Eher nein. Die Kenntnisse der Standard-Tastenkombinationen gehen manchmal ein bisschen weiter als beim «Normal-Benutzer». Doch sich dutzende von Tastenkombinationen zu merken, und dazu noch von Programm zu Programm unterschiedliche, das war bei keiner der getroffenen Personen der Fall.

Die Notwendigkeit, eigene Tastenkombinationen zu hinterlegen ist daher gering. Auch wenn Microsoft dies als bewährte Methode für gutes Keyboard UI Design aufführt[20], war diese Funktionalität von den getroffenen Benutzern nicht gefragt.

1.5.3. Accesskeys-Verwendung

Selbst Kurztasten sind nur zweite Priorität.[21] Sie dienen zum effizienten Arbeiten, sobald jemand das Programm etwas besser kennt und seine «Kernfunktionen» gefunden hat. Dazu ist es aber nötig, dass der Benutzer durch den Screenreader oder die Braillezeile erfährt, dass diese Accesskeys existieren.[22] So kann zum Beispiel jemand, der viele Dokumente in Word erstellt, durch die Sequenz **[alt]**-**[r]**-**[h]** die Formatvorlagen ansteuern.

Kein Benutzer würde sich, bevor er ein Programm das erste Mal verwendet, hinsetzen und erst einmal die Shortcut- und Mnemonics-Liste studieren. Der Ansatz, die Informationen zu Mnemonics und Shortcuts in eine Hilfsdatei auszulagern, bringt die Gefahr mit, Benutzer unnötigerweise zu frustrieren.

1.5.4. Klare Navigation

Mit der «normalen» Tastaturnavigation, ohne Tastenkombinationen und Kurztasten, sollen alle Werkzeuge, Menüs und Funktionen erreichbar sein. Die elementaren Standard-Navigationsmechanismen, näher beschrieben unter «2. Navigation via Tastatur im Detail», sollen dafür befolgt werden. Wichtig ist auch, dass nirgends ein Element isoliert steht, auf das der Screenreader beim Durchnavigieren nicht fallen kann. Denn bei unbekanntem Sachverhalt ist oft «durchtabben», ein Springen von Element zu Element mit **[tab]** und **[ctrl + tab]**, die Devise.[23]

1.5.5. Tab-Reihenfolgen

Die Reihenfolge, in welcher der Benutzer durch die vorhandenen Elemente mit Hilfe von **[tab]** bzw. **[ctrl + tab]** navigiert, ist elementar. Einen Standard gibt es nicht, welcher Bereich nach welchem kommt. Die Tab-Reihenfolge sollte der logischen und prozessorientierten Abfolge angepasst sein. Oft passiert dies bereits korrekt durch die Platzierung der Elemente im Quellcode. Überraschungen des Benutzers sollten vermieden werden, auch ein wildes Hin- und Her-Springen ist nicht gut.

---

20 Guidelines for Keyboard User Interface Design – Microsoft Corporation – Assigning Shortcut Keys

21 Interview mit Urs Hiltebrand, Accesstech – Frage «Welche Rolle spielen Mnemonics?»

22 Guidelines for Keyboard User Interface Design – Microsoft Corporation – Assigning Shortcut Keys

23 Interview und Wissensaustausch mit Selamet Aydogdu – Frage «...Tab kommt also immer zum Einsatz, wenn du nicht genau weisst, was dich erwartet?»

2. Navigation via Tastatur im Detail

2.1. Schematisch dargestellt

Um nicht eine Liste mit unzähligen Tastenbefehlen zu zeigen, die man zwar versteht aber unter der man sich nicht viel vorstellen kann, wird in diesem Kapitel mit Schemen gearbeitet. Diese sollen den normalen Navigationsfluss über die Tastatur aufzeigen. Die hier aufgezeigte Art ist die einfachste Art der Navigation – ganz ohne Mnemonics und Shortcuts (Ausnahme Browser-Navigation, dort sind sie ein wichtiges Element).

2.2. Microsoft Word

Der Startpunkt für das Schema *(fig. 8)* ist ein offenes Word-Dokument – der Standard nach dem Öffnen des Programms. Der Cursor befindet sich blinkend im Seiteninhalt. Zur Orientierung dienen die in *(fig. 7)* eingezeichneten Bereiche.

Das Schema zeigt, wie der «Einstieg» ins Menü-Band über **[alt]** passiert und der «Ausstieg» über **[esc]** oder eine ausgeführte Aktion stattfindet. Im Menü-Band (2) wird mit den Pfeiltasten zwischen den verschiedenen Reitern und auch in den Reitern navigiert. Im aktiven Reiter kann auch mit **[tab]** durchnavigiert werden, Element für Element. Beim letzten Element des Reiters angelangt, springt der Fokus durch Drücken von **[tab]** wieder auf das erste Element des gleichen Reiters.

Einmal im Menü-Band angelangt, kann mit **[ctrl + tab]** zwischen den verschiedenen Bereichen (Menü-Band, allfällig offene Menü-Fenster und der Fussleiste) gewechselt werden. In der Fussleiste wird mit den links- und rechts-Pfeiltasten wie auch mit **[tab]** zwischen den Elementen navigiert. Ebenso intuitiv über die Pfeiltasten ist die Navigation auch in Menü-Fenstern (im Beispiel: Zeichen- und Absatzformate).

Ist der Benutzer auf dem gewünschten Menü-Punkt oder Element angekommen, kann dieses mit **[enter]** oder **[leertaste]** ausgewählt werden.

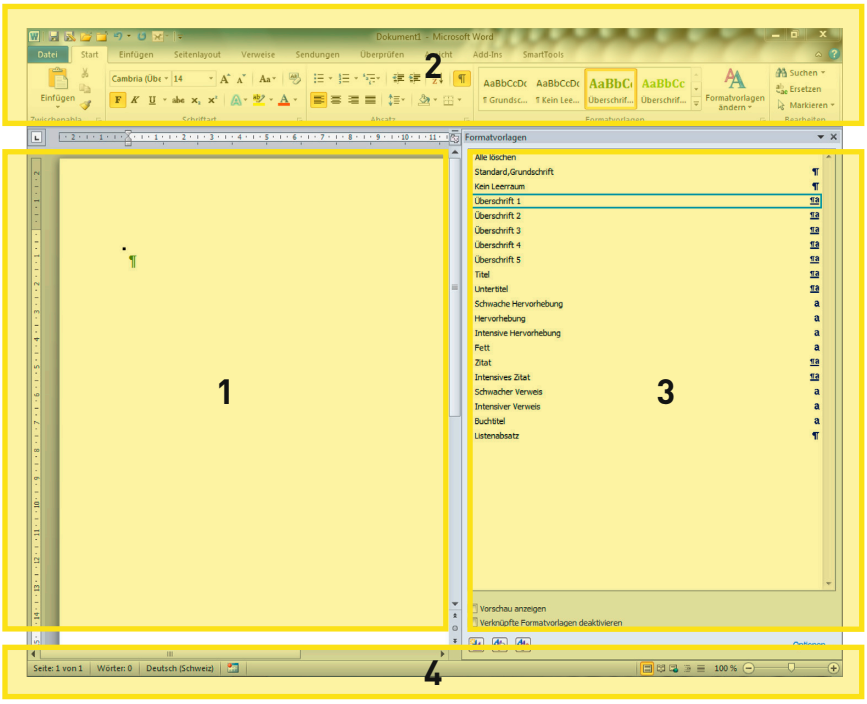


fig. 7 – Übersicht der verschiedenen Bereiche in Microsoft Word: 1) Seiteninhalt, 2) Menü-Band, 3) geöffnetes Menü-Fenster, 4) Fussleiste/Statusleiste

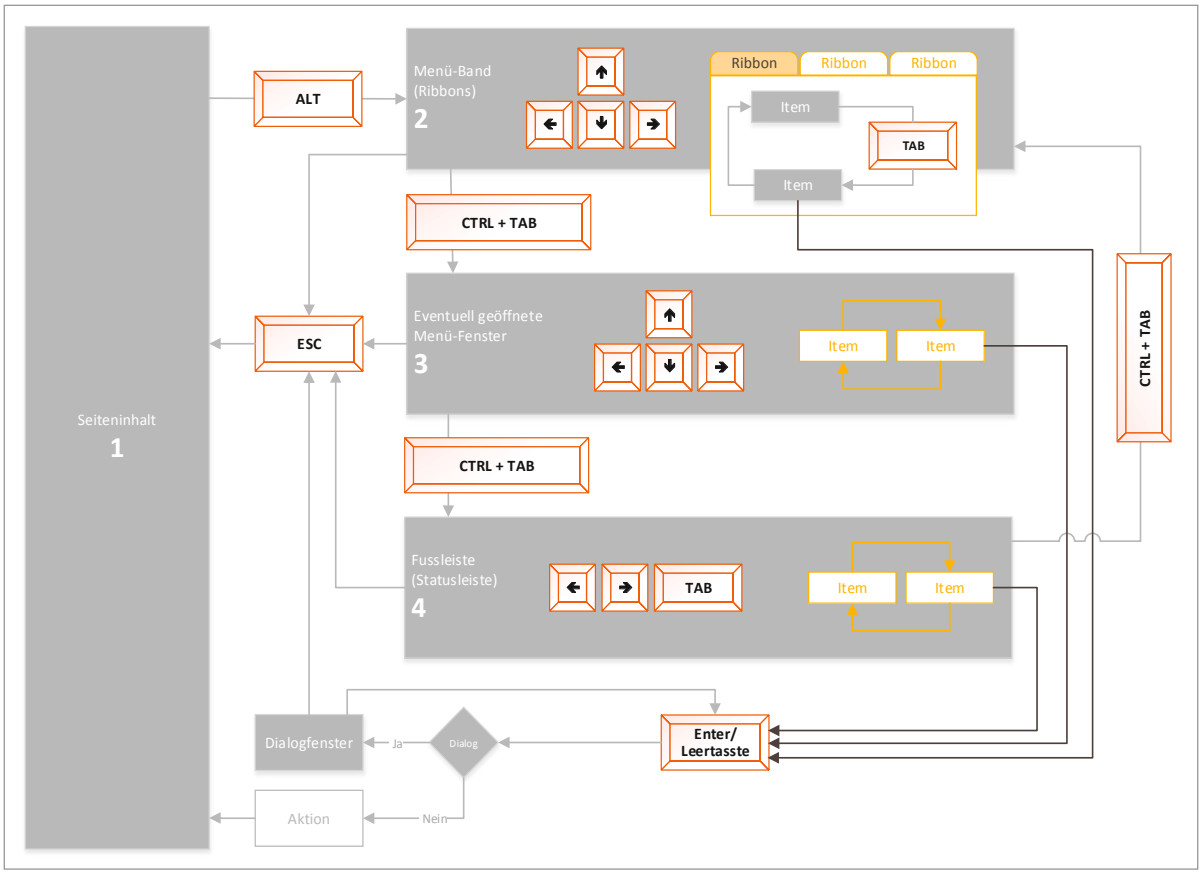


fig. 8 – Tastatur-Navigations-Schema für Microsoft Word, das mit kleinen Abweichungen auf die ganze Microsoft Office Palette übertragen werden kann.

2.3. Windows Desktop

Der Startpunkt für das Schema *(fig. 10)* ist der Desktop ohne geöffnetes Startmenü, wie dies der Standard nach der Anmeldung zum Benutzerkonto ist. Zur Orientierung dienen die in *(fig. 9)* eingezeichneten Bereiche.

Das Schema zeigt die möglichen Wege, wie man via Tastatur einen Programmstart auslösen kann. Der üblichste Weg beginnt dabei mit **[windows]** um das Startmenü zu öffnen. Mit den Pfeiltasten bewegt man sich zum gewünschten Element – mit oder ohne vorgängige Suche über das Suchfeld.

[tab] bringt den Benutzer stets von einem Bereich zum Nächsten (Startmenü, Taskleiste, Symbolbereich/Systemeinstellungen, Desktop) und kann jederzeit gedrückt werden, um weiter zu springen. In der Taskleiste (2), dem Symbolbereich (3) und auf dem Desktop (4) kann ebenfalls mit den Pfeiltasten zwischen den Elementen navigiert werden.

Die Verwendung von **[enter]** garantiert das Starten des Programms bzw. das Drücken des Schalters. Für Aktionen in der Taskleiste und dem Symbolbereich können sowohl **[enter]** wie auch **[leertaste]** verwendet werden.

fig. 9 – Übersicht der verschiedenen Bereiche des Desktops von Windows: 1) Startmenü, 2) Taskleiste, 3) Symbolbereich, 4) Desktop

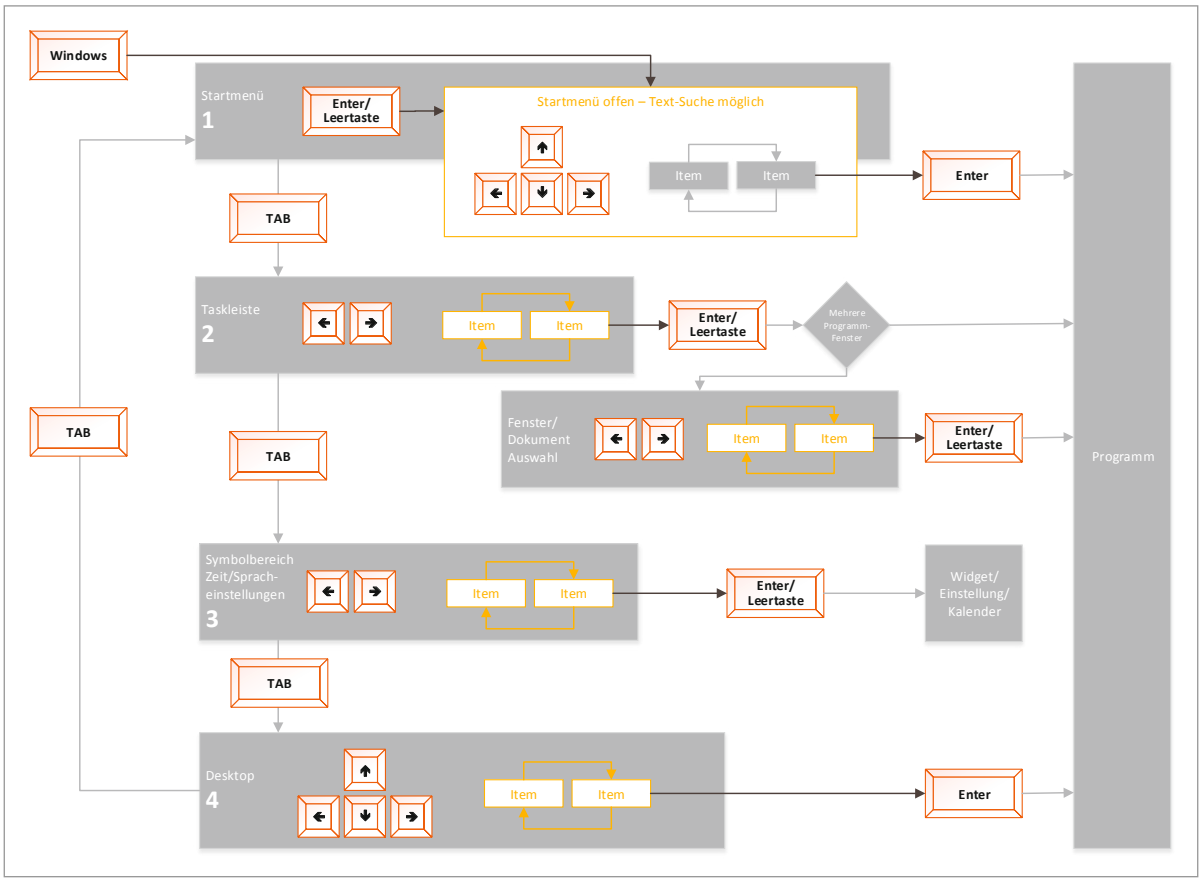


fig. 10 – Tastatur-Navigations-Schema für den Desktop von Windows

2.4. Internet Explorer / Browser

Der Startpunkt das Schema *(fig. 11)* ist eine offene Website, zum Beispiel die Startseite des Browsers, die beim Öffnen des Programms aufgerufen wird.

Das Schema zeigt, dass das klassische «Durchtabben» nur eine der Möglichkeiten ist, zum gewünschten Inhalt einer Website zu navigieren. Alternativen sind folgende:

- *JAWS* stellt Funktionen zur Verfügung, die dann via Dialogfenster auf das entsprechende Element leiten.
- Ebenfalls in Screenreadern hinterlegte Accesskeys lassen, ähnlich dem **[tab]**, den Fokus von Element zu Element einer bestimmten Art springen.

Der Zugriff in die Menüleiste geschieht über **[alt]**. Mit den Pfeiltasten kann man anschliessend durch die Menüleiste zu allen Menü-Punkten navigieren.

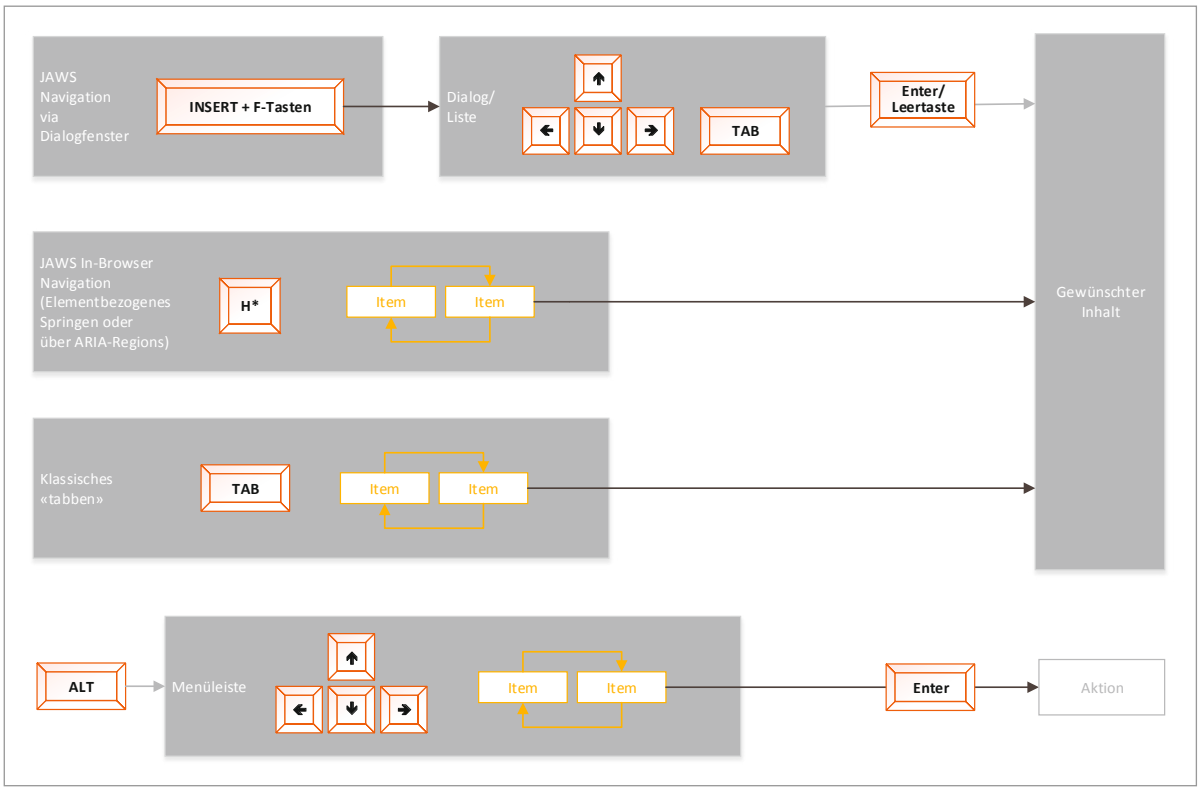


fig. 11 – Tastatur-Navigations-Schema für einen Browser in Kooperation mit JAWS. Die Taste H* steht dabei für eine ganze Reihe von möglichen Direktzugriffstasten.

Zum Navigieren im Browser muss der Benutzer die *JAWS*-eigenen Kurztasten und Tastenkombinationen kennen, um verschiedene, spezielle Accessibility-Funktionen aufrufen zu können. Diese können nur über die *JAWS*-Hilfe gefunden werden. Die hier aufgelisteten Accesskeys gelten als Konvention, die von verschiedenen Screenreadern befolgt werden.

Die wichtigsten Shortcuts zum Navigieren im Internet Explorer über **[Insert + F-Taste]** sind die folgenden:

- **[Insert + F1]** *JAWS*-Hilfe
- **[Insert + F5]** Formular-Felder auflisten
- **[Insert + F6]** Alle Titel auflisten
- **[Insert + F7]** Alle Links auflisten

Die entsprechenden Elemente werden in einem Dialogfenster zur direkten Auswahl bereitgestellt. Wählt man zum Beispiel einen der aufgelisteten Titel, so ist man anschliessend mit dem Fokus direkt im Inhalt auf dem soeben gewählten.

Für das element-bezogene Springen stehen durch die Nutzung eines Screenreaders folgende Kurztasten zur Verfügung:

- **[H]** von Titel zu Titel
- **[E]** von Eingabefeld zu Eingabefeld
- **[F]** von Formularelement zu Formularelement
- **[B]** von Button zu Button
- **[L]** von Liste zu Liste
- **[R]** von *ARIA*-Region zu *ARIA*-Region

Damit zu diesen Elementen gesprungen werden kann, ist, wie weiter oben erwähnt, eine saubere semantische Struktur nötig. Um *ARIA*-Regionen zu finden, müssen diese ausdrücklich im *HTML*-Code definiert sein.

2.5. Gängige Konventionen und Vorgaben von Microsoft für Keyboard UI Design

2.5.1. Basis-Belegung

Zum Garantieren einer möglichst geläufigen Tastaturbefehl-Belegung und zur Vermeidung von Überschneidungen ist es ratsam, bestehende Konventionen zu befolgen.

- **[alt]**-**[ ]** ist der Weg in die Menüleiste um zu Menü-Befehlen zu gelangen.
- **[ctrl +]** werden von Microsoft für programmeigene Shortcuts empfohlen, sowohl mit Buchstaben wie auch mit den F-Tasten.[24]
- **[tab]** lässt von einem Element auf der gleichen Ebene zum nächsten springen, ohne den aktuellen Kontext zu verlassen.
- **[tab + ctrl]** lässt von einem Kontext/Bereich zum nächsten springen.
- **[shift +]** lässt auf das vorhergehende Element springen, dreht die Navigations-Richtung um.
- **[esc]** muss dafür sorgen, aus jeglicher Situation raus auf die Grundstellung des Programms zu kommen, unter Umständen durch mehrmaliges drücken der Escape-Taste. **[esc]** soll den Fokus Ebene für Ebene zurückführen. In einem Untermenü führt **[esc]** immer

24 Guidelines for Keyboard User Interface Design – Microsoft Corporation

auf die nächst höhere Ebene, bis der Fokus abschliessend aus dem Menü herausspringt.[25]

- **[Fn]** oder andere geräteproprietäre Tasten dürfen nicht als Modifikator verwendet werden.[26]
- **[StickyKey]** Es darf nicht vorkommen, mehrere Tasten gleichzeitig drücken zu müssen, um eine bestimmte Funktion oder ein bestimmtes Menü auszuführen: Einzig die Tasten Shift, Ctrl, Alt und Windows können in Kombination verwendet werden, da das StickyKey-Accessibility-Feature von Windows die sequentielle Eingabe dieser Tasten mit einer weiteren Taste erlaubt.[27]
- **[Insert +]**-Kombinationen werden von *JAWS* verwendet, zum Teil um Applikationen direkt zu unterstützen, zum Teil um *JAWS*-Einstellungen zu bearbeiten.

2.5.2. Funktionen der F-Tasten

Die konventionnelle Belegung von F-Tasten sieht folgende Funktionen vor:

- **[F1]** öffnet die Hilfe des aktuellen Programms
- **[F4]** schliesst das aktuelle Fenster, **[alt + F4]** schliesst das aktuelle Programm
- **[F6]** oder **[ctrl + F6]** stehen für Springen zwischen Unterfenstern, zum Beispiel sollte damit der Wechsel von Inhalt zur Übersicht geregelt sein.

Die Belegung zum Aufrufen der Suchfunktion ist nicht einheitlich. Zu empfehlen wäre wohl ein Folgen des Standards von Web-Browsern, also **[F3]**.

25 Interview und Wissensaustausch mit Selamet Aydogdu – Selamet Aydogdu am Laptop
26 Guidelines for Keyboard User Interface Design – Microsoft Corporation
27 Guidelines for Keyboard User Interface Design – Microsoft Corporation

3. Barrierefreiheit in beook

3.1. Ausgangslage

Die Ausgangslage der Applikation *beook*, die auf *eclipse* RCP basiert, ist von Grund auf positiv. Und dies, obwohl beim Aufbau von *beook* Barrierefreiheit kein Kriterium war – wie bei vielen Anwendungen. RCP greift über *JFace* auf die Betriebssystem-eigenen Fenster-Komponenten zu.[28] Das erlaubt Screenreadern, diese Elemente standardmässig zu erfassen.

Mit JAVA *Swing* erstelle Benutzeroberflächen sind für Screenreader ein «weisses Blatt». Würde die Benutzeroberfläche von *beook* also auf JAVA *Swing*-Komponenten basieren, wäre anscheinend ein komplizierterer Vorgang nötig, um die gewünschte Barrierefreiheit zu erzielen.[29]

3.2. Vorgehen

3.2.1. Analyse

In dieser Analyse wurde getestet, welche Elemente der *beook*-Applikation bereits in unbearbeitetem Zustand barrierefrei sind. Die folgenden Punkte galt es für jede Etappe und für jeden Bereich zu analysieren:

- Ist die Navigation mit der Tastatur möglich?
- Werden Elemente vom Screenreader erkannt?
- Sind Shortcuts und Accesskeys vorhanden?
- Wiederspiegeln die GUI-Elemente das eingestellte Windows-Farbschema?

Aus den Erkenntnissen entwickelte Empfehlungen wurden in der Analyse bewusst kurz gehalten. Im Empfehlungskatalog werden dann die kompletten und «optimalen» Varianten und Macharten vorgeschlagen.

3.2.2. Definition von Bereichen in *beook*

Damit für die Ergebnis-Interpretation klar ist, von welchen Elementen gesprochen wird, ist es nötig, die in den folgenden Grafiken *(fig. 12 und fig. 13)* aufgeführten Bereichsbezeichnungen zu kennen.

28 Rich Client Platform/FAQ – Eclipse Wiki – «What is included in the Rich Client Platform?»

29 Interview und Wissensaustausch mit Selamet Aydogdu – Frage «Auf welche Hindernisse bist du schon getroffen, die du als erfahrener User umgehen kannst, aber für einen Normal-User unüberwindbar sind?»

fig. 12 – Das Portal von beook weist die folgenden Bereiche auf:
1) Menüleiste
2) Seitenleiste
3) Portal-Toolbar
4) Anbieter bzw. Inhalt

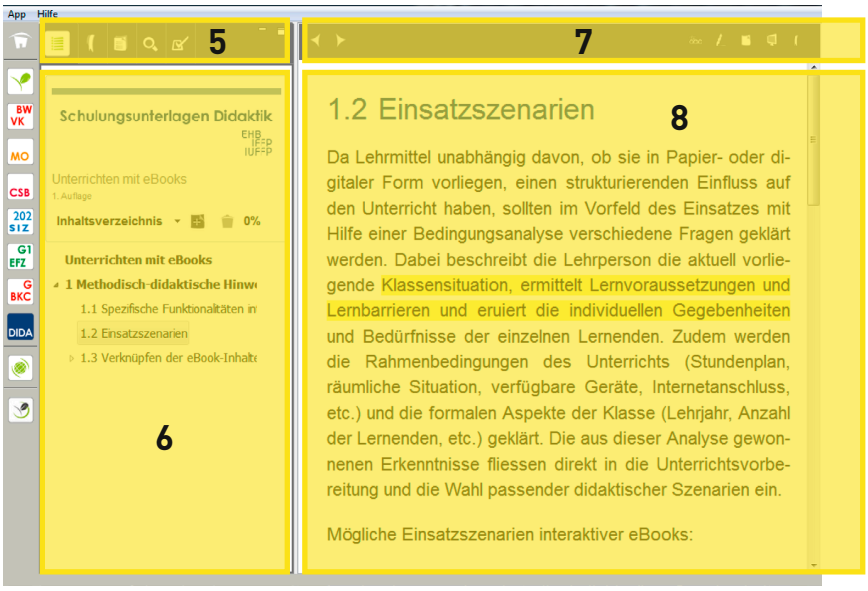


fig. 13 – Die Inhaltsansicht von beook weist, nebst bereits in fig. 12 beschriebenen, die folgenden Bereichen auf:
5) Übersichtsspalten
6) Inhaltsverzeichnis mit Buchinfos
7) Lesebereich-Toolbar
8) Lesebereich

3.3. Ergebnisse

3.3.1. Hauptprobleme

Eine detaillierte Analyse des aktuellen Zustands betreffend Barrierefreiheit[30] zeigt für die *beook*-Applikation (*RCP*-Windows) folgende Hauptprobleme auf:

- Schaltflächen sind nicht benannt. Der Benutzer kann so nicht wissen, was er mit dem Aktivieren auslöst.
- Bereiche sind nicht benannt.

30 Analyse von beook auf Barrierefreiheit, Dokument im Anhang

- Grafiken und Bilder sind nicht zugänglich. Dies stellt besonders ein Problem dar, wenn sie als Link funktionieren.
- Mit **[tab]** lässt sich zwar durch die Benutzeroberfläche «durchtabben». Das ist aber – zum Beispiel – um sich im Inhalt zurecht zu finden eher hinderlich. Das «aus dem Inhalt»-Springen ist problematisch.
- Bei der **[tab]**-Reihenfolge trifft man noch nicht auf alle vorhandenen Elemente, dafür auf solche, die nicht nötig sind (Unterfenster minimieren): Die ganze Lesebereich-Toolbar kann im Moment noch nicht angesteuert werden via Tastatur.
- Programmspezifische Accesskeys und Shortcuts existieren nicht.
- Die wichtigen Funktionen «Markierung setzen» und «Notiz einfügen» können derzeit noch nicht über die Tastatur bewerkstelligt werden. Durch die Bedingung, den Cursor im Text zu navigieren und darauf eine Aktion auszulösen, würde es auch nichts ändern, wenn die Lesebereich-Toolbar angesteuert werden könnte.
- Das im Lesebereich vorhandene Kontextmenü kann nicht aufgerufen werden, dabei enthält es wertvolle Funktionalitäten der *beook*-Applikation.
- Eigene Seiten erstellen ist zwar möglich, aber der Inhalt kann nicht sauber strukturiert werden (zum Beispiel mit <h>-Tags) – ausser man gibt sie direkt im Quellcode ein. Der Text befindet sich immer in einem grossen Eingabefeld. Ein Navigieren darin ist umständlicher als auf einer normalen Inhaltsseite.
- Die Anpassung der Applikation an das Windows-Farbschema scheitert grundsätzlich an der Verwendung von Bildern, die auf Schaltflächen eingebettet sind. Nur einfarbige Icons die auf Transparentem Hintergrund gesetzt sind, passen sich in gewissen Situationen an – in anderen Fällen verschmelzen Sie mit dem Hintergrund zur Unkenntlichkeit.

3.3.2. Bereits funktionierende Elemente

Es ist aber auch schon viel gutes Vorhanden, dass den Weg zur kompletten Barrierefreiheit verkürzt:

- Die Installation kann durch simples Entpacken eines Zip-Ordners gemacht werden. Der Weg über den Windows-Installer ist nicht nötig.
- Menüleiste kann über **[alt]** aufgerufen werden.
- In allen Dialogfenstern kann man wie gewohnt navigieren.
- Schliessen der Applikation mit **[alt + F4]**
- Im Inhaltsverzeichnis-Bereich funktioniert die Navigation mit Pfeiltasten gut (in allen Spalten), die Bedienung der Listen mit Unterpunkten ist wie im Windows Explorer, die Sprachausgabe funktioniert gut.
- Im Lesebereich kann mit den gewohnten Browser-Befehlen navigiert werden.
- Die Verwendung der systemeigenen Bildschirmlupe von Windows hat gut funktioniert.

3.3.3. Übungsteil

Der Textinhalt funktioniert gut, da dessen Struktur auf dem *EPUB*-Standard mit Erweiterungen basiert. Der Inhalt im Übungsteil ist strukturell für die Barrierefreiheit weniger gut: Die Übungen sind Zusammensetzungen aus Formular-, Drag'n'Drop- und Bildelementen, die sich in einer jeweils bestimmten, eigens entwickelten *HTML*-Struktur befinden.

Problematisch, auch aus sinnbezogenen Überlegungen, sind die folgenden Übungstypen:

- Zeichnungseditor
- Zuordnung
- Bildmarkierungen
- Kreuzworträtsel

Bei folgenden Übungstypen hindern derzeit Navigationsprobleme die Möglichkeit, sie ausschliesslich mit Tastatur und Screenreader lösen zu können:

- Markieren
- Satzzeichen
- Gross-Klein

Am ähnlichsten mit bekannten Formularfunktionalitäten und daher bereits jetzt durch Tastatur ansteuerbar und lösbar sind:

- Textaufgaben
- Multiple-Choice und Mehrfachauswahl
- Lückentext

Ihnen fehlt nur noch die oft nötigen Labels, damit auch die Sprachausgabe korrekt ist.

Eingriffe in die *HTML*-Strukturen der Übungen zwecks Barrierefreiheit sind in der Weiterentwicklung von *beook* sekundär. Zuerst soll *beook* als barrierefreie Applikation ihre Brauchbarkeit beweisen.

3.3.4. Download von *beook*-Webseite

Nicht berücksichtigt für die Analyse wurde der Ort des Downloads der Installations-Datei. Die Website im Fall von *beook* ist nicht komplett barrierefrei. Eine Navigation ist zwar möglich, die Struktur ist dafür sauber genug, die *ARIA*-Regions «Navigation» und «Inhalt» werden auch erkannt. Aber es fehlt die Konsequenz, so sind zum Beispiel keine Grafiken beschrieben.

4. Empfehlungskatalog

4.1. Rahmen der Verwendung

Dieser Empfehlungskatalog enthält konzeptionelle Grundelemente, welche zum Erstellen von barrierefreien und für Screenreader lesbaren Desktop-Applikationen wichtig sind. Zu jedem Punkt sind die geplanten Umsetzungen für die *beook*-Applikation aufgeführt, die bei der Weiterentwicklung von *beook* zu einer barrierefreien `RCP`-Applikation befolgt werden sollten.

Die hier aufgeführten Empfehlungen sind über Programmgrenzen hinweg gültig. Für Programme, die strukturell ähnlich sind wie *beook*, sind sie natürlich einfacher zu befolgen. Für andersartige Applikationen können sie aber trotzdem beherzt und abgeleitet werden. Das jeweils dazu aufgeführte Beispiel von *beook* soll die Empfehlung in einem konkreten Kontext zeigen. Daher soll dieser Empfehlungskatalog nicht eine reine Aufzählung von möglichen Tastenbefehlen sein, sondern das Konzept hinter der Bedienung beleuchten. Die technische Umsetzung, die je nach Programmiersprache unterschiedlich aussehen kann, wird nicht beschrieben.

Die Empfehlungen versuchen, existierende Konventionen zu berücksichtigen und spiegeln die Resultate der vorhergehenden Analyse. Ein Anspruch auf Vollständigkeit für alle erdenklichen Programmtypen und -strukturen existiert nicht, aber die Bedürfnisse für die bereits bestehende Applikation *beook* werden abgedeckt.

4.2. Empfehlungen

4.2.1. Laden der Applikation

Ladezeiten können kürzer oder länger sein – je nach Gerät und Applikation. Während dieser Zeit sollte trotz optisch sichtbarem Ladebildschirm ein akustisches Signal oder eine Sprachnachricht diesen Zustand mitteilen. Dies kann eine Prozentangabe sein, es kann aber auch eine Tonfolge oder ein anhaltendes Geräusch sein. Der einfachste Weg dazu ist, die Windows-Systemtöne zu unterstützen, damit die Tongebung dem Benutzer bereits bekannt ist – und automatisch bereits ein- beziehungsweise ausgeschaltet ist.

Der Eintritt in das Programm wird in der Regel vom Screenreader angekündigt, in dem die Titelleiste des geöffneten Programms gelesen wird. Somit ist das Ende des Ladeprozess auch erkennbar.

fig. 14 – Startbildschirm von beook: Akustisch sollte mitgeteilt werden, dass der Ladeprozess in Gang ist. http://i.ytimg.com/vi/X9uRCKtaOCU/maxresdefault.jpg (Lautsprecher)

Umsetzung in *beook*

Für *beook* ist die Sprachausgabe des angezeigten Textes geplant. So wird der Benutzer informiert, was passiert. Zusätzlich wird beim Programmstart die Windows-Systemsound-Ausgabe unterstützt. Dadurch ist sichergestellt, dass bereits beim ersten Programmstart die Einstellungen für blinde und visuell eingeschränkte Personen stimmen und auch der «normal» sehende Benutzer nicht gestört wird.

4.2.2. Grundstellung

Um sich immer wieder zurechtfinden zu können, ist eine immer gleiche Grundstellung nötig. Wie ein Desktop mit geschlossenem Start-Menü oder Microsoft Word mit dem blinkenden Cursor im Text. Diese Grundstellung muss zu jeder Zeit über **[esc]** erreicht werden können. **[esc]**-Befehle nach dem Erreichen der Grundposition haben keine navigierende Wirkung mehr. Ein akustisches Signal kann aber melden, dass man die Grundposition erreicht hat.

Grundsätzlich soll nach dem Programmstart diese Grundstellung eingenommen sein.

Die Frage, welches bei einem vielseitigen Programm die Grundstellung ist, kann etwas schwieriger sein. Denken Sie dabei an Ihre Benutzer und was sie erwarten, wo deren «Landmark» in Ihrem Programm sein kann – und nicht unbedingt an das, was in erster Linie logisch ist.

Umsetzung in *beook*

Bei *beook* ist die Empfehlung für die Grundstellung folgendes: Fokus auf dem Portal-Button in der Seitenleiste. Dabei kann der angezeigte Inhalt unterschiedlich sein:

- Wenn man sich aus einer Anbieter-Ansicht des Portals «escapen» will, dann wird rechts in der Grundstellung immer noch die Anbieter-Ansicht angezeigt, der Fokus wird aber auf den Portal-Button gelegt *(fig. 15)*.
- Wenn man sich aus einem Dialog zum Bearbeiten von Markierungen im Leseteil «escapen» will, dann wird rechts in der Grundstellung noch das offene Buch im zuletzt besuchten Kapitel angezeigt. *(fig. 16)*

Der Wechsel von einer Anbieter-Ansicht auf die Anbieter-Übersicht stellt keinen Niveau-Sprung dar, sondern einen Funktionsaufruf. Dies wurde mit dem Ansatz konzipiert, dass ein Benutzer in der Regel seinen «Hauptanbieter» hat.

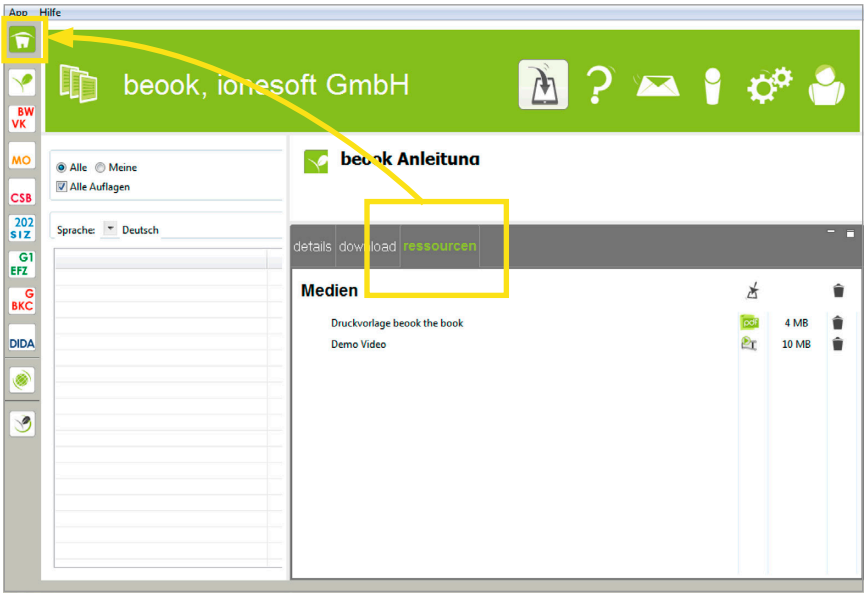


fig. 15 – Der [esc]-Sprung von einem fokussierten Register in der Anbieter-Detail-Ansicht führt den Fokus direkt auf den Portal-Button der Seitenleiste.

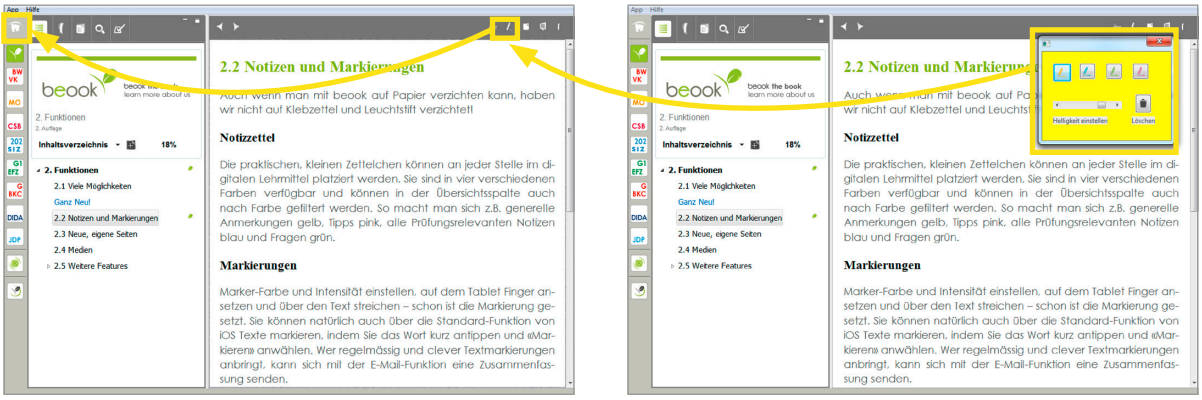


fig. 16 – Die [esc]-Sprünge von einem fokussierten Dialogfenster, in diesem Fall die Farbauswahl für die Markierung, zurück auf die Schaltfläche des Werkzeugs in der Lesebereich-Toolbar und zuletzt auf den Portal-Button der Seitenleiste.

Warum diese Wahl? Der Portal-Button ist ein konstanter Ort, von dem aus man sich neu orientieren kann. Im gleichen Bereich befindet sich zudem auch die Buchauswahl. Eine zusätzliche Information via Sprachausgabe wäre auch noch möglich, um den Benutzer aufzuklären, ob er nun das Portal oder den Lesebereich vor sich hat.

Diese Idee entspricht zudem dem konventionellen Verhalten, durch **[esc]** von einer Unterebene eine Ebene nach der anderen nach oben zu springen – dies ist bei der flachen Hierarchie der *beook*-Applikation relativ simpel.

4.2.3. Tab-Navigation von Element zu Element

Eine zweistufige Tab-Navigation kann für Klarheit und Komfort in der Navigation sorgen.

Generell was die Navigation angeht: Bloss nicht das Rad neu erfinden! In Anlehnung an *ARIA*-Regions im Web und die Bereichswechsel in Microsoft Word kann man die Navigation in einer Applikation von Bereich zu Bereich mit **[ctrl + tab]** regeln. So springt man von Zone zu Zone. Beim Landen in einem neuen Bereich informiert die Sprachausgabe, in welchem man sich befindet.

Innerhalb jedes Bereiches wird dann mit **[tab]** von Element zu Element gesprungen – ohne Gefahr zu laufen, aus der gewünschten Zone unbemerkt wieder heraus zu springen. Dies gilt auch für Symbolleisten und Registerkarten, bei denen nebst mit **[tab]** auch eine Navigation mit den Pfeiltasten möglich sein sollte.

Das Auswählen eines Elements, das Aktivieren einer Schaltfläche, wird mit **[enter]** gemacht. Alternativ die **[leertaste]** für die identische Funktion anzubieten stiftet mehr Verwirrung als es Komfort schafft.

Umsetzung Navigation in *beook*

In *beook* sieht eine zweistufige Navigation so aus, dass mit **[ctrl + tab]** in der Lesereihenfolge von Bereich zu Bereich gesprungen wird. Im Inhaltsbereich ergibt sich daraus folgende Reihenfolge: 1) Seitenleiste, 2) Übersichtsspalten, 3) Inhaltsverzeichnis, 4) Lesebereich-Toolbar, 5) Lesebereich *(fig. 17)*

Jetzt könnte man bemängeln, dass der Lesebereich, der ja das Hauptelement des ebook-Readers von *beook* ist, erst als letzter Bereich in der Tab-Reihenfolge den Fokus bekommt. Dem können drei Elemente entgegengehalten werden: Über die Kombination **[shift + ctrl + tab]** kann die Richtung des Durchlaufs geändert werden. Also ist der Lesebereich nur einen Shortcut von der Grundstellung entfernt. Zudem entspricht die Reihenfolge für die Konsultation einer neuen, bestimmten Stelle bis auf den in diesem Fall überflüssigen Sprung zur Lesebereich-Toolbar sehr gut. Und drittens ist die Reihenfolge aufgrund der Programmstruktur für jemand Sehenden logisch, also so zu erwarten.

Die Tab-Reihenfolge für den Portal-Bereich ist sehr ähnlich. *(fig. 18)*

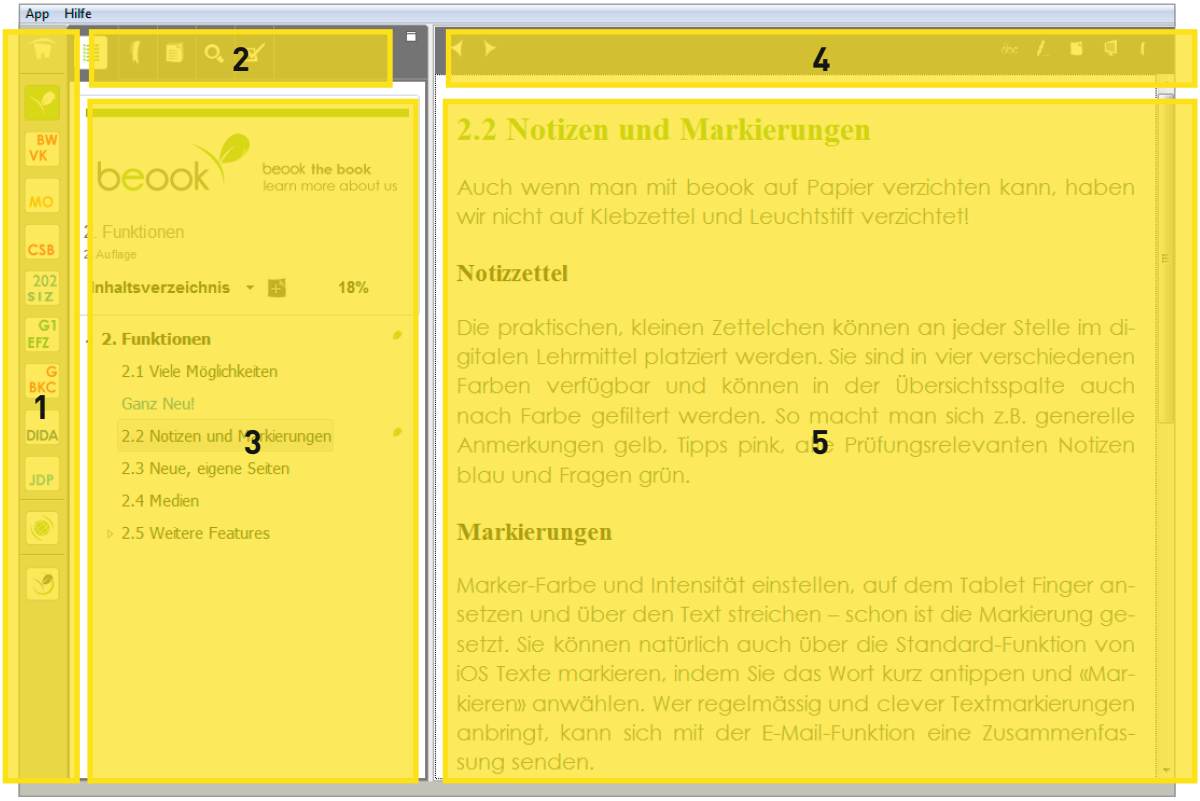


fig. 17 – Die Tab-Reihenfolge über die Bereiche des Inhalts und des Lesebereichs, über die mit [ctrl + tab] navigiert wird: 1) Seitenleiste, 2) Übersichtsspalten, 3) Inhaltsverzeichnis, 4) Lesebereich-Toolbar, 5) Lesebereich

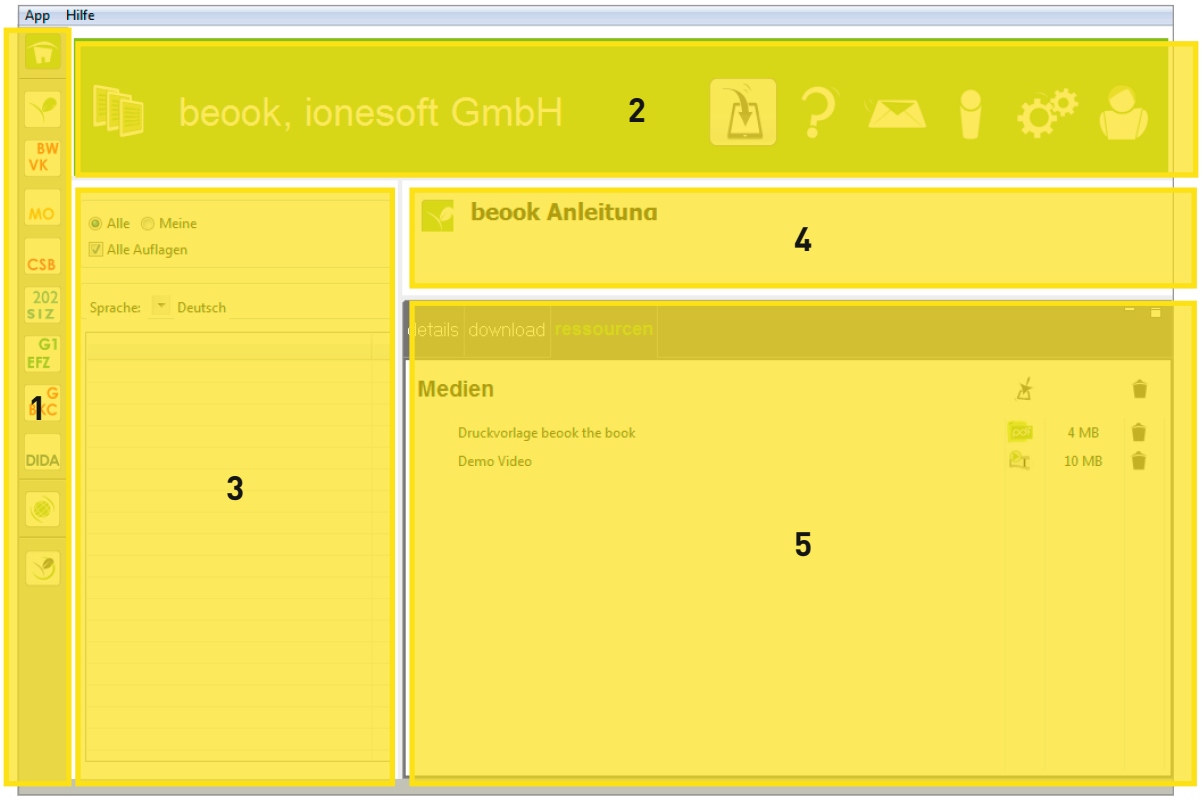


fig. 18 – Die Tab-Reihenfolge über die Bereiche des Portals, über die mit [ctrl + tab] navigiert wird: 1) Seitenleiste, 2) Portal-Toolbar, 3) Produktverzeichnis, 4) Produkttitel, 5) Produktdetails

4.2.4. Menüleiste

Die Menüleiste bieten für Benutzer die Möglichkeit, sich einen raschen Überblick über die Funktionen des Programms zu verschaffen – ähnlich einer Sitemap einer Webseite. Es ist eine gewohnte Art, über die Menüleiste an alle Befehle zu kommen. Für Power-User ist die Menüleiste auch der Ort, an dem Shortcuts nachgeschlagen werden können.

Umsetzung in *beook*

beook ist grundsätzlich auf die Touch-Bedienung ausgelegt. Dadurch wurden nur eine kleine Anzahl Funktionen in der Menüleiste platziert, was für die Barrierefreiheit wie auch menü-gewohnte Benutzer kein Vorteil ist. Eine generelle Ansteuerung von Funktionen, Ansichten und Einstellungen über die Menüleiste ist in Betracht zu ziehen – auch wenn dies nicht unbedingt notwendig ist. Die Menüleiste sollte aber, so denn diese Strategie gewählt wird, die Möglichkeiten sinnvoll und komplett abbilden.

4.2.5. Benennen von Buttons und Bereichen

Die grösste Schwierigkeit haben Screenreader mit Schaltern, die nur ein Bild aufweisen aber keinen Textinhalt haben. Bei denen ist es für Screenreader unmöglich, die eigentliche Funktion zu kennen, die durch das Drücken des Buttons aufgerufen wird.

Buttons müssen unbedingt via Tooltip oder andere Label-Möglichkeiten beschrieben werden, selbst wenn sie Text enthalten. Möglichst klare und eindeutige Beschreibungen sind wichtig. Im beschreibenden Text sollte zudem ein allfälliger Accesskey oder Shortcut erwähnt werden.

Auch Bereiche und Dropdownlisten sollten Labels oder Namen tragen, damit beim Durchtabben klar ist, wo man sich befindet. Geben Sie in der Beschreibung eines Bereichs auch an, wie man darin navigiert (ob mit **[tab]**, den Pfeiltasten, etc.).

Umsetzung für *beook*

Als Bereichsbezeichnungen werden die unter Punkt «3.2.2. Definition von Bereichen in *beook*» aufgeführten Titel verwendet. Zusätzlich sollte vom Screenreader gelesen werden können, ob es sich beim Bereich um eine Schalter-Liste, einen Listeninhalt oder einen Textbereich handelt und wie darin navigiert werden kann.

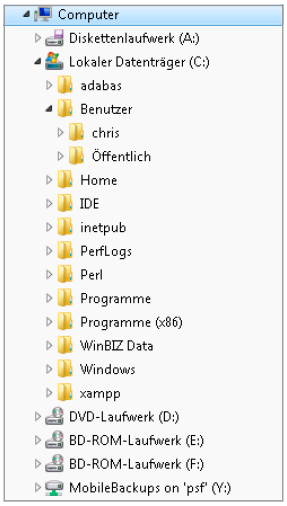


fig. 19 – Windows-Explorer-Liste mit mehreren Hierarchien

4.2.6. Listen

Für Listen mit mehreren Hierarchiestufen ist eine Funktionsweise wie bei den Listen vom Windows Explorer anzustreben *(fig. 19)*. Diese sind sehr gut für Screenreader zugänglich. Die Bezeichnung der abgebildeten Liste lautet «Strukturansicht» und es wird bei jedem Element erwähnt, ob ein Element offen ist und wie viele Elemente es enthält.

Bei normalen Listen mit nur einer Hierarchiestufe ist es gut, wenn der Benutzer am Ende der Liste angelangt und dabei ein akustisches Signal ertönt. So weiss der Benutzer, wann er am Ende der Liste angelangt ist – was bei langen Listen sehr praktisch ist. Sollte er das gesuchte Element verpasst haben, dient es dem Benutzer, wenn der Fokus nach dem letzten Element

bei einem erneuten Drücken der gleichen Taste wieder auf das Erste springt. Damit entfällt ein mühsames «Zurück-Navigieren». Und auch der generellen Orientierung ist dies zuträglich, Listen nicht gezwungenermassen vorwärts und rückwärts durchlaufen zu müssen.

Durch die Listen wird mit den Pfeiltasten navigiert.

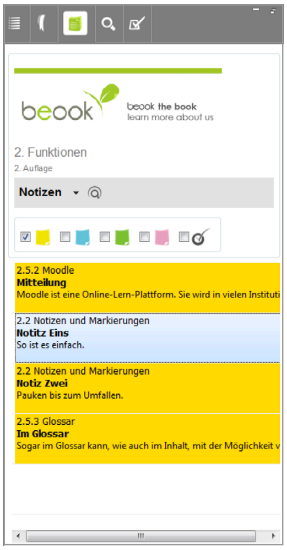


fig. 20 – Übersichtsspalte «Notizen» im Inhaltsverzeichnis-Bereich: Eine Auflistung der gelben Notizzettel, der aktuelle Fokus liegt auf dem zweiten Eintrag.

Umsetzung in *beook*

Die Strukturansicht im Inhaltsverzeichnis bringt beste Voraussetzungen mit: Die Funktionsweise ist bereits analog dem beschriebenen Verhalten von Windows-Explorer-Listen mit mehreren Hierarchien. Zusätzlich muss von der Sprachausgabe mitgeteilt werden, ob es sich beim Inhalt um eine eigene Seite oder um eine bestehende Seite handelt. Diese Unterscheidung ist derzeit nur optisch – durch eine Blaufärbung des Textes – vorhanden.

Für Listen, wie sie als Resultat in den Übersichtsspalten «Lesezeichen», «Markierungen/Notizen» *(fig. 20)* und «Suche» vorkommen, soll ein «Rundlauf» implementiert werden, damit der Fokus vom letzten Element in der Liste mit erneutem Drücken der Pfeiltaste wieder zum ersten springt. Mit einem akustischen Signal sollte das Angelangen am Ende der Liste angegeben werden. Dies kann ein «Bing» sein oder eine Sprachausgabe «Ende der Liste». Um das akustische Signal besser vom Gelesenen abzuheben, wäre einem textlosen Ton den Vorzug zu geben. Optimal wäre die Einbindung eines Windows-Systemsounds.

4.2.7. Textlicher Inhalt

Bei dem in der Analyse behandelten Programm *beook* ist der textliche Inhalt zentral. Die Hauptfunktion ist schliesslich das bequeme Lesen von digitalen Lehrmitteln. Daneben gibt es natürlich allerlei Applikationen, deren Kern aus Formularfeldern, Listen, Menüs und generell aus wenig Text bestehen. Doch auch diese Programme haben textliche Inhalte, zu Beispiel bei Hilfe-Texten, Anleitungen und Dokumentationen oder auch bei Texteditoren, die für grössere Textmengen gedacht sind.

Bei all diesen Elementen empfiehlt es sich, eine Navigationsfähigkeit ähnlich einem Browser zu schaffen: Das Springen von Titel zu Titel, Listen zu Listen, Grafik zu Grafik etc. sollte über konventionelle Accesskeys möglich sein. Optimal ist es, wenn Screenreader die semantische Struktur des Textinhalts erfassen können und nicht nur mit optischen Auszeichnungen wie zum Beispiel «fett», «grösser» oder «kleiner» gearbeitet wird.

Gerade für Hilfe-Texte und Anleitungen, die oft nicht direkt im Layout der eigentlichen Applikation angezeigt werden, eignen sich *HTML*-basierte Dokumente gut. Mit einem einfachen, geöffneten *HTML*-Dokument mit dem Standard-Browser des Benutzers kann vermieden werden, auf eine nicht barrierefreie Webseite weitergeleitet zu werden, wo sich ein Benutzer verlieren kann.

Umsetzung in *beook*

Der Inhalt im Textbereich hat eine komplette *HTML*-Struktur, die in einem integrierten Browser angezeigt wird. Dadurch kann mit dem Screenreader[31] wie gewohnt mit den Accesskeys im Inhalt navigiert werden.

Bei weiteren Textinhalten sollen die derzeit verschiedenen Typen von Textinhalten vereinheitlicht werden. So sollen die Informationen des Menüs «Über» genau gleich wie der lokal vorhandene «Hilfe-Text» und andere Erklärungen als strukturell sauberes *HTML* im integrierten Browser angezeigt werden.

Der Link zum vorhandenen Online-Support-Portal soll ein möglichst direkter Link sein – und nicht auf der Willkommensseite von *beook*.ch landen.

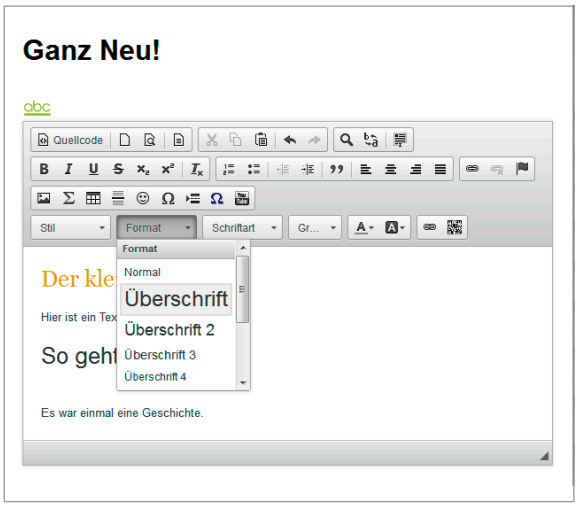


fig. 21 – Der geöffnete CKeditor zum Erstellen einer eigenen Seite, mit geöffneter Formatvorlagen-Dropdownliste

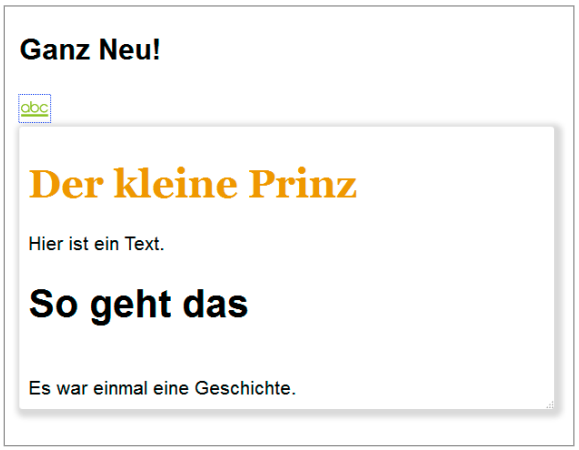


fig. 22 – Die eigene Seite mit geschlossenem Editor und dem Inhaltstext, der sich in einem riesigen Eingabefeld befindet.

4.2.8. Texteditor

Verwendete Texteditoren für längere Texte, die eine Struktur haben müssen, sollten grundsätzlich das Verwenden von Stilvorlagen fördern. Dank sauber erstellten Stilvorlagen kann direkt eine korrekte semantische Struktur entstehen. Dabei trägt schon nur das korrekte Auszeichnen von Titelhierarchien, Listen und Links viel zur Barrierefreiheit bei.

Umsetzung in *beook*

In *beook* wäre eine Anpassung des derzeit vorhandenen Texteditors zum Erstellen von neuen Seiten anzustreben. Der eingebettete CKeditor *(fig. 21)* kann über die Tastatur nicht bedient werden. Optimal wäre eine barrierefreie Bedienung des Programms im Programm. Realistischer ist wohl eine über Shortcuts definierte Bedienung, die während dem Bearbeiten im Eingabefeld die wichtigsten Menüs aufrufen kann (u.a. Formatvorlagen, Liste, Symbole einfügen).

Bei implementierten, externen Opensource-Produkten ist die Abhängigkeit von deren Entwicklung stets ein Knackpunkt.

Bei einer Anpassung der eigenen Seite-Funktionalität wäre zusätzlich Folgendes in Betracht zu ziehen: Ist eine eigene Seite erst einmal geschrieben, sollte diese als «flaches» *HTML* gezeigt werden, ohne in einem Eingabefeld integriert zu sein *(fig. 22)*. Nur so kann gewohnt – wie in bestehenden Seiten im Leseteil auch – in den eigenen Seiten mit Browser-Accesskeys navigiert werden, ohne Gefahr den Inhalt ungewollt zu verändern.

31 Funktioniert mit JAWS. Derzeit ist dies mit dem NVDA-Screenreader nicht möglich: Bug 407481, NVDA is silent when using XULRunner 10.0.4 – Eclipse.org

4.2.9. Shortcuts

Tastenkombinationen erlauben es zwar, dass Power-User schnell an Ihre Menüs kommen. Betreffend «Barrierefreiheit» spielen sie aber eine sekundäre Rolle: Als oberstes Gebot gilt: Alle Funktionen müssen mit der normalen Tastaturnavigation erreichbar sein. Es gilt nicht zu vergessen, dass blinde und sehbehinderte Benutzer bereits eine gewisse Anzahl an Shortcuts kennen müssen, um den jeweiligen Screenreader zu bedienen. Bekannte Shortcuts wie **[ctrl + c]**, **[ctrl + v]** und **[ctrl + o]** zu implementieren ist gängig, die dann auch die erwarteten Befehle Kopieren, Einfügen und Öffnen ausführen.

Sollten dennoch «neue» Shortcuts gesetzt werden, muss sichergestellt sein, dass ein Menü-Punkt oder Button zum Aufrufen der gleichen Funktion existiert, am besten mit Sprachausgabe der zugewiesenen Tastenkombination. Eine Funktion darf nicht exklusiv über den Shortcut erreicht werden. Es wird abgeraten, die Beschreibung der definierten Tastenkombinationen einfach in ein «Hilfe-Menü» oder in ein «Manual» auszulagern. Sie müssen dort beschrieben sein, wo sie gebraucht werden!

Umsetzung in *beook*

Um die Benutzerfreundlichkeit zu erhöhen, sind für die Navigation in den Übersichtsspalten Shortcuts vorgesehen. Wechsel zwischen den Übersichtsspalten sind bei der intensiven Nutzung von *beook* eine häufige Handlung. So kann ein ständiges Navigieren zum Bereich der Übersichtsspalten umgangen werden.

Ansonsten sind keine weiteren programmspezifischen Shortcuts vorgesehen, ausser sie sind bereits klare Konvention, wie zum Beispiel **[alt + F4]** zum Beenden des Programms.

4.2.10. Accesskeys

Accesskeys dienen erfahrenen Benutzern zum effizienteren Arbeiten und können in einer geschlossenen Umgebung eines Programms verwendet werden. Wie Shortcuts dürfen auch Accesskeys nie der einzige Weg sein, um eine Funktion auszuführen oder ein Menü zu wählen. Accesskeys werden ergänzend eingesetzt. Bei Menü-Punkten, Schaltflächen und Registerkarten sollten vorhandene Accesskeys beim Darübernavigieren von der Sprachausgabe erwähnt werden.

Bei der Programmkonzeption sollte die Internationalisierung dieser Accesskeys berücksichtigt werden, damit diese auch möglichst eingängig sind für die Benutzer: So macht zum Beispiel **[l]** für «Lesezeichen» auf Deutsch Sinn, ist aber für die englische Version unpassend, da Lesezeichen «bookmarks» heisst.

Wenn lange, textliche Inhalte in der Applikation vorkommen, kann eine Browser-Ähnliche Accesskey-Implementation in Erwägung gezogen werden – so denn dieser Textinhalt nicht bereits in einem implementierten Browser angezeigt wird. Doch Achtung: gerade wenn dies der Fall ist, sind eigene Accesskeys im Kontext der textlichen Inhalte fast nicht mehr möglich, ohne Konflikte zu erschaffen.

Umsetzung in *beook*

Für alle Menü-Punkte in der Menüleiste werden klare, sprachangepasste Accesskeys gesetzt. Ebenso für die Register der Übersichtsspalten, die Schaltflächen der Lesebereich-Toolbar, die Schaltflächen in der Portal-Toolbar und die Register der Produktdetails. Zusätzlich sind zwei Accesskeys für die konstanten Elemente der Seitenleiste vorgesehen: **[p]** für Portal, **[b]** für Browser.

4.2.11. Funktionstasten

Als globale Art von Accesskeys werden oft die Funktionstasten verwendet – dies ist der Grund, warum sie hier ein separates Unterkapitel erhalten. Folgende Belegung ist empfohlen:

- **[F1]** Hilfe aufrufen
- **[F3]** Suche
- **[F4]** Fenster schliessen (nicht Programm)
- **[F6]** zwischen Unterfenstern beziehungsweise Tabs wechseln

Umsetzung in *beook*

Die Taste **[F1]** führt direkt zum Hilfetext, der auch über das Menü «Hilfe» aufgerufen werden kann. **[F3]** setzt den Cursor direkt in der Übersichtsspalte «Suche» in das Suchfeld. Die Suche ist somit auf die globalst-mögliche Suche eingeschränkt, welches die Suche in einem einzelnen Buch ist.

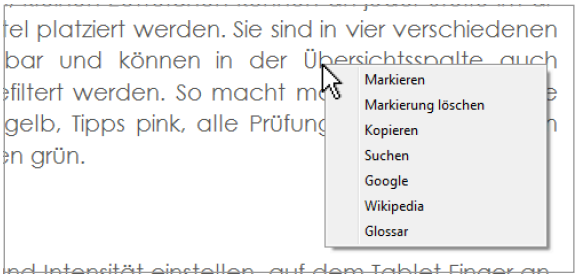


fig. 23 – Das Kontextmenü des Lesebereichs soll über die Menüleiste zugänglich werden.

4.2.12. Kontextmenü

Kontextmenüs werden in der Regel über einen Rechtsklick der Maus aufgerufen und stehen immer in Bezug mit dem Ort des Klicks. Um der reinen Tastatur-Navigation ebenfalls diese Möglichkeiten zu bieten, ist ein Weg über Menü-Punkte in der Menüleiste der logische Weg. So bleibt zum Beispiel ein selektierter Text ausgewählt, während man über **[alt]** in die Menüleiste einsteigt und dort die gewünschte Funktion aufruft.

Umsetzung in *beook*

Für das Kontextmenü des Lesebereichs *(fig. 23)* wird in der Menüleiste ein neues Menü geschaffen und nimmt all die im Kontextmenü vorhandenen Punkte auf. Zusätzlich wird dieses mit der Funktion «Notiz einfügen» ergänzt.

4.2.13. Beenden der Applikation

Das Beenden der Applikation muss in jedem Fall über **[alt + F4]** möglich sein. Ein Menü-Punkt in der Menüleiste kann auch dazu dienen, ist jedoch nicht unbedingt nötig.

Umsetzung in *beook*

Die *beook*-Applikation hat diesen Standard-Befehl implementiert. Die Applikation schliesst sich über **[alt + F4]**.

4.2.14. Accessibility-Feature: On/Off

Grundsätzlich sollte versucht werden Accessibility-Features so zu implementieren, dass sie für den normal sehenden Benutzer nicht auffallen. Wer sie aber sucht, soll sie auch finden. Denn auch ein «Normal-Benutzer» kann zum Beispiel von Shortcuts und Accesskeys profitieren.

Elemente, die zusätzlich zur Screenreader-Ausgabe hinzukommen, sollten in den Einstellungen an prominenter Stelle ein- beziehungsweise ausgeschaltet werden können. Dies weil zum Beispiel die Ausgabe von akustischen Signalen, die zusätzlich eingebaut werden, für «Normal-Benutzer» schnell störend wirken. Auch für den Wechsel von «sehr ausführlichen» auf «normale» Tool-Tipps oder Hilfs-Labels kann eine Einstellungsmöglichkeit sorgen.

Umsetzung in *beook*

Zweistufige Labels und Tooltips sind nicht vorgesehen. Dafür können im Einstellungsmenü die zusätzlichen, akustischen Hinweise ein- beziehungsweise ausgeschaltet werden – so denn diese nicht komplett über die Standard Windows-Systemsounds gelöst werden können.

4.2.15. Das Wichtigste: Selber ausprobieren!

Bevor Sie mit der Umsetzung einer barrierefreien Applikation beginnen, begeben Sie sich selbst in die Situation, ausschliesslich mit der Tastatur zu navigieren. Und dann steigern Sie das Erlebnis, in dem Sie einen Screenreader installieren und sich anschliessend eine Schlafmaske aufsetzen. Ausser natürlich, Sie gehören zu der beeindruckenden Anzahl blinder und visuell eingeschränkter Programmiererinnen und Programmierer, die es auf dieser Welt gibt – dann kennen Sie dies ja schon.

Erleben Sie selbst, was es heisst, wenn plötzlich kein Feedback kommt, wenn ein Programm für den Screenreader ein weisses Blatt ist. Oder wie Ihnen ein einfaches Interface plötzlich als «langer Text auf einer einzigen Zeile» doch kompliziert scheint. Und stellen Sie bald einmal fest, was stört.

Dabei ist klar, dass man durch eine stündige «Blind-Session» nicht zum Profi wird. Aber es hilft, Verständnis zu entwickeln um letztendlich ein (noch) besseres Resultat zu erschaffen.

4.3. Zweck des Empfehlungskatalogs

4.3.1. Barrierefreie Applikation? – Gleiche Fragen

Die Punkte «4.1. Rahmen der Verwendung» bis «4.3.3. Anmerkungen zum *beook*-Übungsteil» werden als eigenständiges Dokument veröffentlicht. So kann dies einer breiten Allgemeinheit dienen, welche bei der Umsetzung einer barrierefreien Applikation vor den gleichen Fragen steht:

- Was ist überhaupt wichtig?
- Wie muss eine barrierefreie Navigation aussehen?
- Welche Standards und Konventionen existieren?
- Wie bedient der Benutzer die Applikation?
- Was wird oft vergessen?

Einen Verweis auf diese Bachelor-Vorarbeit wird bei der Publikation ebenfalls platziert. Die Empfehlungen des Katalogs sind für die Schnellleser, die in kurzer Zeit das wichtigste Wissen müssen. Um die Hintergründe wirklich zu verstehen ist aber ein Studium der Analyse ebenfalls nötig.

4.3.2. Plattform zur Veröffentlichung: IT-Blog von *Access for All*

Als Plattform zur Veröffentlichung dient der IT-Blog von *Access for All*[32]. Dieser hat in der Deutschschweiz eine gute Reichweite und beherbergt bereits viele wertvollen Informationen zum Thema «Barrierefreiheit in der Informatik». Zudem ist die Seite barrierefrei konzipiert.

Da es möglich ist, den Blogeintrag und somit den Empfehlungskatalog zu kommentieren, ist auch auf ein vielseitiges Feedback der vorhandenen Community zu hoffen. Um durch kritische Anregungen und Inputs diese Publikation noch weiter zu verbessern.

4.3.3. Anmerkungen zum *beook*-Übungsteil

In den Empfehlungen ist kein Abschnitt dem Übungsteil von *beook* gewidmet. Dieser ist ein grosser Knackpunkt in der Applikation betreffend Barrierefreiheit. Die verschiedenen Übungstypen, die wie Mini-Anwendungen im Lesebereich implementiert sind, stellen von Typ zu Typ komplett neue Anforderungen. Um diese barrierefrei zu machen, ist eine vertiefte Analyse jedes einzelnen Typs nötig. Diese müssten nach den besten Regeln der Web-Accessibility gebaut und strukturiert werden. Funktionelle Anpassungen wären dabei nicht auszuschliessen.

Dieser Teil wäre in Angriff zu nehmen, sobald der Rahmen, die *beook*-Applikation, auf einem zufriedenstellenden Stand ist.

32 www.access4all.ch/blog/?p=4605

Fazit

Bei allen kontaktierten Personen und Organisationen wurden durch diese Arbeit offene Türen eingerannt: Eine grosse Bereitschaft zur Mithilfe war da. Nur dadurch waren diese tiefen Einblicke möglich, welche einen wichtigen Teil zur Analyse beigetragen haben.

Die Recherchen haben gezeigt, dass es betreffend barrierefreie Desktop-Applikationen keine Standards gibt, aber viele recht breit abgestützte Konventionen. Diese gilt es zu befolgen, um den blinden und sehbehinderten Benutzern die Nutzung der Applikation so einfach wie möglich zu machen.

Durch den Empfehlungskatalog besteht ab sofort eine schnell erfassbare Wissensgrundlage für die Programmierung von Desktop-Applikationen, welche auch barrierefrei sein sollen. Konventionen und bewährte konzeptionelle Ansätze können darin gefunden werden.

Ein letztes Kapitel wäre noch den Inputs der öffentlichen Publikation gewidmet worden – so denn sie bereits beim Druck dieser Arbeit vorhanden gewesen wären. Abschliessend kann man also gespannt sein, welche Verbesserungen, Kritiken und Rückmeldungen auf dem Blog von *Access for All* zusammenkommen werden. Klar ist nur: Dort wird ab sofort immer die qualitativ beste Version des Empfehlungskatalogs zu finden sein.

Literaturverzeichnis

Monographien, Beiträge und Publikationen

Stefan SPRING – *Sehbehinderung und Blindheit: Entwicklung in der Schweiz*
Schweizerischer Zentralverein für das Blindenwesen SZB, St. Gallen, 2012

Diese Publikation gibt einen Überblick der aktuellen statistischen Datenlage über Blinde und Sehbehinderte in der Schweiz, und liefert Wissen um Klarheit betreffend Terminologie zu schaffen.

Internetquellen

Sehbehinderung und Blindheit: 10 Fragen und Antworten – www.sbz.ch > Wissen
http://www.szb.ch/de/wissen/das-wichtigste-zu-blindheit-und-sehbehinderung-im-ueberblick/sehbehinderung-und-blindheit-10-fragen-und-antworten.HTML#c2
Stand 22.05.2015, Abfrage am 22.05.2015

Screen Reader User Survey #4 Results – WebAIM Web Accessibility in Mind – Home > WebAIM Projects
http://webaim.org/projects/screenreadersurvey4/
Stand 05.2012, Abfrage am 21.05.2015

Support Us, Partner with Us – *NV Access*, Home of the free NVDA screen reader – www.nvaccess.org > Support Us
http://www.nvaccess.org/support/
Stand 05.2015, Abfrage am 23.05.2015

Guidelines for Keyboard User Interface Design – Microsoft Corporation
https://msdn.microsoft.com/en-us/library/ms971323(d=printer).aspx
Stand 22.03.2010, Abfrage am 18.05.2015

Rich Client Platform/FAQ – Eclipse Wiki
https://wiki.*eclipse*.org/Rich_Client_Platform/FAQ#What_is_the_Eclipse_Rich_Client_Platform.3F
Stand 08.02.2013, Abfrage am 25.05.2015

Bug 407481, NVDA is silent when using XULRunner 10.0.4 – Eclipse.org
https://bugs.*eclipse*.org/bugs/show_bug.cgi?id=407481
Stand 08.05.2013, Abfrage am 27.05.2015

Wikipedia

Sehschärfe, Wikipedia
http://de.wikipedia.org/wiki/Sehschärfe – Stand 22.03.2015, Abfrage am 22.05.2015

Persönliche Aufzeichnungen

Besuch an der Schule für Sehbehinderte Zürich
Notizen im Anhang, Schule für Sehbehinderte besucht am 06.02.2015

Besuch am Gymnasium Wetzikon
Notizen im Anhang, Gymnasium Wetzikon besucht am 27.02.2015

Interview mit Urs Hiltebrand, Accesstech
Transkription im Anhang, geführt am 13.05.2015

Interview und Wissensaustausch mit Selamet Aydogdu
Transkription im Anhang, geführt am 19.05.2015

Analyse von beook auf Barrierefreiheit
Analyseraster mit Resultaten im Anhang, erstellt am 25.05.2015

Videoquellen

Robert KINGETT – *How blind people use the web* – Youtube
https://youtu.be/ymDf1CNMKzY – crippledcritic, Chicago, Veröffentlicht am 28.06.2012, Abfrage am 19.05.2015

Der visuell eingeschränkte Autor Robert Kingett demonstriert im Video, wie sich Blinde im Internet bewegen. In einem zweiten Teil gibt er auch Informationen dazu, was gut und was schlecht konzipierte Webseiten sind. Die Demonstration macht er mit dem Screenreader NVDA, einem für ihn angepassten Windows-Farbschema und einer prise Humor.

Abbildungsverzeichnis

Glossar

Accesskey auch manchmal Mnemonic oder Hotkey genannt, sind alphanumerische Tasten die verwendet werden, um ein Menüpunkte anzuwählen.[33] Nicht zu verwechseln mit Shortcuts, die einen direkten Aufruf eines Befehls erzeugen.

ARIA **Accessible Rich Internet Applications** Ein von der Web Accessibility Iniative entwickelter Webstandard, der die Zugänglichkeit von Webseiten und Webanwendungen für Blinde, die Screenreader verwenden, erhöht. Seit März 2014 ist ARIA ein empfohlener Webstandard des World Wide Web Consortiums (W3C).[34]

ARIA-Regions Definierte Bereiche in Webseiten, deren Bezeichnung von Screenreadern erkannt wird und zwischen denen man durch Screenreader-Software-Funktionalitäten hin- und herspringen kann.

AT **Assistive Technology** Dies ist ein Sammelbegriff, mit dem alle Technologien bezeichnet werden, die Personen mit Einschränkungen helfen, diese Einschränkungen zu überwinden. AT umfasst ein breites Spektrum, vom Hörgerät über die in dieser Arbeit oft erwähnten Screenreadern bis hin zu Prothesen oder speziell für Behindertensport entwickelte Sportgeräte.

beook Applikation für digitale Lehrmittel der ionesoft GmbH

Bug Ein kleiner Programmierungsfehler der zu Funktionsstörungen in einer Software führt.

CSS **Cascading Style Sheet** Ein Dokument, dass die Stile zur Formatierung von HTML-Elementen enthält. Dies ist so konzipiert, um Form (CSS) und Inhalt (HTML) zu trennen.

eclipse Entwicklungsumgebung, die auf dem RCP-Framework basiert.

EPUB **Electronic Publication** Ist ein auf XML basierter, global verwendeter Standard für digitale Bücher und Publikationen.

GUI **Graphic User Interface** Grafische Benutzeroberfläche, die mit optischen Elementen Daten, Funktionen und Struktur sichtbar macht.

IE **Internet Explorer** Windows Standard Internet-Browser

JAVA Programmiersprache, mit der man systemunabhängige Programme schreiben kann, die auf einer virtuellen Maschine, einer Art «Übersetzer», ausgeführt werden.

Java Swing Eine Programm-Bibliothek zum Erzeugen von Benutzeroberflächen-Elementen, die nicht vom Betriebssystem abhängig sind, sondern immer gleich aussehen, da sie JAVA-nativ sind.

JAWS **Job Access with Speech** Name der derzeit weltweit am stärksten verbreitete Screenreader-Software, hergestellt von Freedom Scientific.

JFace Eine Programm-Bibiliothek, welche die Verwendung von SWT- und weiteren Benutzeroberflächen-Komponenten vereinfacht. Bestandteil der RCP.

MAA **Microsoft Active Accessibility** Technologie, die das Zusammenspiel von Microsoft Windows, Applikationen und den Technischen Hilfsmitteln wie Screenreader verbessert.

Mnemonic siehe Accesskey

NVDA **Non Visual Desktop Acces** Ein gratis Screenreader der Firma NV Access, der von einer breiten Community unterstützt wird und zu deren Hauptsponsoren Google und Adobe gehören.

RCP **Rich Client Platform** Programm-Basis, die es erlaubt mit vordefinierten Komponenten komplette JAVA-Applikationen zu erstellen.

---

33 https://msdn.microsoft.com/en-us/library/windows/desktop/bb226831(v=vs.85).aspx

34 http://de.wikipedia.org/wiki/Accessible_Rich_Internet_Applications

Screenmagnifier Software, die Bildschirmausschnitte vergrössern kann. Beispiele sind ZoomText, MAGic und die Systemeigenen Bildschirmlupen auf Windows und Mac.

Screenreader Software, die den Inhalt eines Bildschirms erfassen und vorlesen kann. Beispiele sind JAWS, NVDA, VoiceOver (iOS) und TalkBack (Android).

SWT **Standard Widget Toolkit** Eine Programm-Bibliothek, welche eine systemunabhängige Verwendung von Betriebssystem-Eigenen Fenstern und Benutzeroberflächen-Komponenten ermöglicht.

TTS Text-To-Speech Auch bekannt als Sprachsynthese, eine künstliche Stimme, welche geschriebenen Text akustisch ausgeben kann.

XML **Extensible Markup Language** Eine Auszeichnungssprache, die für Menschen lesbar ist und zum Datenaustausch entworfen wurde.

Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Verwendung anderer als der angegebenen Hilfsmittel verfasst habe und dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt zitiert habe.

Bern, 29. Mai 2015

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

comem+

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Diplôme 2015
Travail de Bachelor

Département Comem+
Ingénierie des Médias
Heimann Christian
christian.heimann@heig-vd.ch

25.05.2015

Accessibility für elektronische Lehrmittel

Répondant externe

Daniel Stainhauser
ionesoft GmbH
Sandrainstrasse 17
3007 Bern
dstainhauser@ionesoft.ch
+41 32 513 39 91

Lieu du travail

ionesoft GmbH
Sandrainstrasse 17
3007 Bern

Le candidat

Heimann Christian

Le répondant interne

Hess Jean-Pierre

Explication générale (motivation, définition stricte du sujet)

Ausgangslage Die ionesoft GmbH ist in der Schweiz derzeit führend mit «beook», ihrer Technologie für elektronische Lehrmittel (Workflow, Umwandlung, Plattform, Urheberrechtsschutz). Im Rahmen des Projektes «Accessibility goes Mainstream» in Zusammenarbeit mit der Stiftung «Access for all» wird die Technologie erweitert, um die Inhalte auch Blinden und visuell Beeinträchtigten zugänglich zu machen.

Mit den heutigen technischen Möglichkeiten liegen alle Elemente bereit, die es visuell eingeschränkten Menschen ermöglichen, Inhalte zu erfassen, die bisher nur Sehenden vorbehalten waren. Jetzt gilt es, diese Möglichkeiten auszunutzen!

Cadrage du sujet (problématique, hypothèses de travail, méthodologie, ressources, etc.)

TPB: Konzeptionelle Überlegungen zur optimalen Navigation in der beook Applikation und zur barrierefreien Anwendung des kompletten Funktionsumfanges. Die erarbeiteten Definitionen können bei einer Implementierung von Accessibility Features als Grundlage dienen. Damit ein grösstmögliches Publikum davon profitiert, soll das Dokument öffentlich zugänglich gemacht werden.

TB: Die Bachelorarbeit konzentriert sich in einem ersten Schritt auf das Herstellen eines Proof of Concepts. An einer inhaltsfreien Beispielanwendung (RCP, Windows, mit JAWS als Screenreader) wird die Verwendung von Accessibility-Techniken erkundet, aufgezeigt und dokumentiert. Diese Elemente und der dazugehörige Quellcode wird veröffentlicht.
In einem zweiten Schritt wird das gewonnene Wissen zur Implementierung von nötigen programmierungstechnischen Elementen in der beook Applikation umgesetzt. Dadurch soll die RCP Version (Windows) der beook Applikation soll einen grossen Schritt in Richtung Barrierefreiheit machen. Vom Quellcode der beook Applikation werden nur Auszug- und Screenshot-artige Teile abgebildet, mit Kennzeichnung der während dieser Arbeit eingefügten Elementen.

Objectifs TPB

- Empfehlungskatalog mit den erarbeiteten Definitionen.
- Veröffentlichung des Dokuments.

Objectifs TB

- Eine barrierefreie Beispielanwendung mit GUI- und Steuerungs-Elementen, die auch in der beook Applikation enthalten sind.
- Veröffentlichung von Erkenntnissen und des Quellcodes der Beispielanwendung.
- RCP Applikation «beook» für den Screenreader JAWS lesbar machen und eine barrierefreie Navigation ermöglichen.

Tâches TPB

- Analyse der aktuellen Windows RCP Applikation «beook» unter dem Aspekt «Barrierefreie Bedienung».
- Analyse von Standard-Keyboard-Belegungen von Computern für visuell eingeschränkte Personen, deren Mnemonics-Verwendung und Präferenzen von Tab-Reihenfolgen.
- Definition eines Keyboard-Access-Schemas unter Berücksichtigung von Standard-Belegungen und Mnemonics, festlegen von Tab-Reihenfolgen.
- Definition von Anpassungen und Implementationen, die zur barrierefreien Bedienung nötig sind.

Tâches TB

- Proof of Concept: Erstellen einer Beispielanwendung mit GUI-Elementen der beook Applikation.
- Erstellen einer Dokumentation mit Muster-Vorgehensweisen zum Erstellen von barrierefreien RCP-Applikationen.
- Erläuterungen von Problemen und deren Lösungen, die beim Erstellen des «Proof of Concepts» erfasst werden.
- Erstellen und implementieren eines Keyboard-Access-Konfigurators.
- Implementation von Elementen zum Lesbarmachen für Screenreader und zur hindernisfreien Navigation der beook Applikation.

Anhang

Auf den folgenden Seiten finden Sie diese Dokumente:

- Aufzeichnungen vom Besuch an der Schule für Sehbehinderte Zürich
- Aufzeichnungen vom Besuch am Gymnasium Wetzikon
- Interview mit Urs Hiltebrand
- Interview und Wissensaustausch mit Selamet Aydogdu
- Analyse von *beook* auf Barrierefreiheit

Besuch der Schule für Sehbehinderte Zürich, 6. Februar 2015

Um wirklich nachvollziehen zu können, was es heisst, blinden und sehbehinderten Menschen Zugang zu Medien zu ermöglichen, war es unabdingbar zu sehen, wie Betroffene mit den bestehenden Technologien arbeiten. An der Veranstaltung «Accessibility goes Mainstream» kam ein Kontakt mit Elgorriaga Ouli-Minna zustande, die u.a. auch als Lehrperson an der Schule für Sehbehinderte unterrichtet. So konnte ein Besuch arrangiert werden, der es mir ermöglichte, einmal in eine Klasse mit blinden und sehbehinderten Personen hineinzusitzen.

Generell wollte ich mich beim Besuch noch nicht auf etwas Bestimmtes fokussieren. Das primäre Ziel war, eine Idee zu erhalten, von was wir überhaupt reden – und um den Endkunden kennen zu lernen. Eine echte Glaubwürdigkeit und eine Anerkennung kann eine Lösung nur erhalten, wenn sie akzeptiert wird. Es kam daher auch zu keinen Interviews, obwohl dass natürlich schon einige Fragen notiert waren. Die Beobachtungen und Kurzgespräche lieferten aber schon viele wichtigen Informationen.

Fragen (Facts)
- Welche Programme werden verwendet?
- Welche zusätzlichen Geräte werden verwendet?
- Welches OS ist das «Haupt-Betriebssystem» derzeit (das am häufigsten verwendete)?
- Wie hoch ist das Arbeitstempo/Lesetempo?
- Wo und wie spielen Shortcuts eine Rolle?
- Welche Unterschiede gibt es zwischen den einzelnen Anwendern?

Fragen (Persönlich)
- Wo stösst du auf Hindernisse?
- Was kannst du nicht machen aber würdest es gern?
- Wenn du vorgelesen bekommst/mit Braille-Zeile liest, kannst du das beschreiben, was da in deinem Kopf abgeht?
- Hast du einen Wunsch im Bezug auf Accessibility?
- Wie oft und wie machst du Hausaufgaben?

Beobachtungen

Severin hat eine Sehbehinderung, verwendet MAGic mit *JAWS*, sein Computer wurde von Accesstech Konfiguriert. Er hat auch eine Braille-Zeile, die ihm das Lesen von Texten vereinfacht. Zum Navigieren benutzt er sowohl die Maus wie auch die Tastatur – einfach was ihm gerade besser geht.

Sein Anspruch an das Gerät bzw. an dessen Arbeits-Geschwindigkeit kann nicht ganz erfüllt werden. Er stört sich an den Verzögerungen, die durch MAGic, der Bildschirmlupe, entstehen. Der die Vorlese-Geschwindigkeit des Screenreaders ist sehr hoch eingestellt. Man kann Severin, obwohl erst 13, als Power-User bezeichnen.

Interessant ist, dass er sein Desktop-Hintergrundbild wöchentlich wechselt. Und das, obwohl er auf eine extreme Vergrösserung angewiesen ist, um Elemente auf dem Bildschirm zu erkennen. Das verwirre ihn nicht sondern gäbe Abwechslung!

Nadja war für ein Jahr am Gymnasium Muristalden in Bern, sie konnte aber stoffmässig nicht ganz mithalten. Heute besucht sie eine Handelsmittelschule in Zürich und war am Tag meines Besuchs auch nur Zufällig an der Schule für Sehbehinderte, um ihren ehemaligen Betreuern einen Besuch abzustatten. Da sie links teilweise gelähmt ist, benutzt sie zur Eingabe eine Einhand-Tastatur. In Lehrmitteln, sagt sie, seien oft die Illustrationen das schlimmste. Diese seien quasi immer unzugänglich.

Melissa hatte eine individuelle Schulung mit Edith Haller von Accesstech. Melissa besucht ein Gymnasium in Zürich. Ihre Sicht ist derzeit noch recht gut, die wird sich aber leider in den nächsten Jahren mit hoher Wahrscheinlichkeit verschlechtern. Sie kann heute noch Texte ab einer Schriftgrösse von 30pt lesen. Zum Arbeiten und für den Schulbesuch benutzt sie ein Surface Tablet mit Windows 8, als Bildschirmlupe verwendet sie die Software «ZoomText». Für den Unterricht erhält sie in der

Regel Skripte der Lehrer als Word. Notizen von Wandtafeln-Abschriften, so denn diese nicht im Skript enthalten sind, erhält sie von ihren Mitschülerinnen. Dank der Möglichkeit zur Eingabe via Stift kann Melissa dank der Technologie etwas erhalten, was sie sonst verloren hätte: Ihre eigene Handschrift. So kann sie normal Schreiben und das Geschriebene anschliessend vergrössern und wieder lesen. Für Notizen im Unterricht, ob Handschriftlich oder via Tastatur, verwendet sie die Applikation «OneNote» von Microsoft.

Edith Haller erzählt von der Schwierigkeit, dass Word immer visueller wird und dadurch immer schwerer zu bedienen wird für Blinde und Sehbehinderte. Während dieser Schulung geht es vor allem um die Anpassung und Konfiguration des Gerätes um das Erlernen von gewissen Funktionalitäten, da in diesem Fall das Gerät eine persönliche Anschaffung von Melissa war und nicht direkt von Accesstech gelifert wurde, es ist als Dispositiv daher eher eine Ausnahme. In der Regel hat Accesstech für Schulungen einen Standardisierten Ablauf, aber immer mit einem Teil Schulung, die auf die Kundin angepasst ist. Edith Haller erwähnt auch, dass nicht alle Personen, denen man eine Schulung gibt, so einfach seien wie Melissa. «Die Aversion gegen Hilfsmittel ist manchmal sehr gross.»

Accesstech ist der einzige Anbieter für IT-Hilfsmittel in der Schweiz, mit Filialen in Luzern und Neuenburg. Plus gibt es eine Hilfsmittelstelle im Tessin, die zwar nicht zu Accesstech gehört, aber mit Know-How von Accesstech arbeitet.

Philipp benutzt ebenfalls Zoomtext zum «Cursor-Highlighten» und einen riesigen Mousecursor. Seine Ansprüche an Hilfmittel und deren Performance sind – wie Severins – sehr hoch. Und das trotzdem immer im Wissen, dass es zu den vorhandenen Mittel zum Teil keine Alternativen gibt.

Er und Severin arbeiten an einer Präsentation, welche sie im Unterricht halten müssen. Dazu verwenden sie Microsoft Powerpoint. Dabei, und das ist in die Lektion, die sie besuchen, eingeplant, lernen sie das Programm kennen. Programme müssen gelernt werden – ob man jetzt «normal» sieht oder nicht. Die Funktionen müssen gesichtet und ausprobiert werden. Das Konzept im Kopf, was man mit dem Programm überhaupt machen kann, muss zuerst entstehen. Das Funktionen anders angesteuert werden und die Prioritäten in erster Linie auf Struktur und nicht auf Optik liegen.

Ouli-Minna erzählt, dass die Entwicklung der letzten zwanig Jahren schon extrem viel geändert und verbessert hat. In den 80er-Jahren war noch die «Perkins», die Schreibmaschine für Braille eines der wichtigsten Hilfsmittel. Heute sind selbst Mini-Braillezeilen für Mobiltelefone erhältlich.

Als Klassengrösse ist bereits bei 2 das Maximum erreicht. Eine Eins-zu-Eins-Betreuung ist im Unterricht fast immer nötig. Denn sobald dass die Schülerin an einem Element hängen bleibt, ist es ungut, wenn es lange Dauert, bis man das Problem lösen kann; sich selber helfen ist in gewissen Momenten sehr schwierig.

Amisa war daran, die Braille-Schrift zu lernen. Sie ist vollblind, in der vierten Klasse und singt für ihr Leben gern. Gerne gab sie eine Demonstration – nicht nur vom Braille-Lesen, das sie derzeit noch schnell ermüden lässt. Sondern liess auch noch gleich ihre gesanglichen Qualitäten aufblitzen.

Arjuna hatte Mathematik-Unterricht bzw. Nachhilfe. Das ist für die sehbehinderte nicht immer ganz einfach. Nicht nur, dass sie Mathe nicht so «ihr» Fach findet, sondern auch weil sie die Rechnungen und Formeln mit Latex bewältigt. Dieser optische Input ist «Maschinenlesbar» und kann auch auf dem Computer in riesiger Schrift einfach eingegeben werden. Aber Latex-Formel zu verdauen ist doch schwerer als die uns «normal sehenden» bekannten Formeln. Sie hätte viel lieber direkt eine Art Rechner, wo man das einfach eintippen kann und es dann ausgerechnet wird …

Besuch des Gymnasiums Wetzikon, Laura, vollblind, integriert in Regelklasse, 27. Februar 2015

Im Gymnasium Wetzikon müssen die Klassen in der Regel das Schulzimmer je nach Fach und Tag wechseln. Für die Klasse von Laura gilt das aber nicht. Auch die Pultanordnung wird nicht verändert, ihr Platz ist ersichtlich für sie reserviert. Dazu hat sie ihren Spind gerade vis-à-vis der Tür des Klassenzimmers.

Laura wird in Fächern mit «visuellen Komponenten» unterstützt. Ouli-Minna Elgorriaga übernimmt vor allem Physik und Mathematik, eine weitere Hilfsperson übernimmt Fächer wie Biologie, Chemie und Geografie. Für Sprachfächer ist keine zusätzliche Person bei Laura in der Klasse.

An ihrem Pult hat Laura das Laptop mit *JAWS* ausgerüstet, eine Braillezeile Focus40 von *Freedom Scientific*, dazu Kopfhörer, damit die Sprachausgabe den Rest der Klasse nicht stört. Zum Schreiben auf ihrem iPhone, das sich dank VoiceOver zu einem guten Begleiter für Blinde und Sehbehinderte entwickelt hat, hat sie auch noch eine kleine Braillezeile (Focus14, also mit 14 Zeichen), die der grossen in Sachen Funktionen in nichts nachsteht. Die begrenzte Zeilenlänge ist aber zum Lesen für längere Texte nicht optimal. Diese Braillezeile verbindet sich via Bluetooth mit dem Smartphone.

Dank der Hilfsperson verschnellert sich die Eingabe bzw. die Auffassung. Tafel-Einträge des Lehrers werden von ihr nochmal klar wiederholt, dass Laura mitschreiben kann – es ist schliesslich kein Blick an die Tafel möglich. Ouli-Minna korrigiert ihr auch falsche Eingaben, die sie nicht sofort bemerkt – um spätere Verwirrung zu vermeiden. In dieser Lektion arbeitet Laura auch nur mit Braillezeile und Tastatur, aber ohne Sprachausgabe – zumindest so lange sie im Word-Dokument arbeitet. In diesem Unterricht hat sie das Skript vorab als Word erhalten und kann darin auch direkt die Aufgaben finden und lösen. Auch die in der Theorie behandelten Stellen kann sie so mit der Braillezeile mitlesen. Das Skript als Word zu erhalten klappe fast immer.

Eingaben zu Mathematik werden schon bei eigentlich einfacheren Rechnungen zu langen, schwerverdaulichen Zeichenketten in Latex, was von Laura eine andere Herangehensweise verlangt. Und obwohl sie von sich selbst sagt, dass die «Zahlenfächer» ihr nicht so liegen, bewältigt sie die neu erlernte Materie sehr einfach und gut.

In ihrer Freizeit, so erzählt Laura, spielt sie Klavier. Sie habe erst gerade begonnen damit, aber es mache ihr grossen Spass. Was sie aber unbedingt gerne mal machen würde, ist Playstation spielen, «gamen». Denn bei ihren Freunden und Mitschülern sei dies oft ein Thema. Aber es bleibt derzeit etwas völlig unzugängliches für sie.

Ouli-Minna sagt, dass nicht an allen Gymnasien so viel Rücksicht genommen wird auf Blinde Schülerinnen, wie dies hier in Wetzikon der Fall ist. Sie windet der Schulleitung diesbezüglich ein Kränzchen.

Telefon-Interview mit Urs M. Hiltebrand, Accesstech – Experten Interview

Am 13. Mai 2015

Welche Standard-Tasten-Belegungen gibt es? Welchen muss man sich fügen? Welche Standards werden von Ihnen festgelegt?
Grundsätzlich muss man Windows-Standards befolgen; Windows schreibt vor, was diese Standardkombination und -befehle machen sollen, was diese schon so vorsehen. Zum Beispiel das Ctrl + O ein Dokument öffnet oder Ctrl + Q ein Programm schliesst. Wenn Ribbons da sind [Menu-Icon-Bänder], dann müssen die entsprechend den Windows-Standards funktionieren. Mit «alt» bewegt man sich einen Schritt nach oben, mit «Tab» einen Schritt nach unten, mit «Ctrl + Pfeiltaste» kann man sich ebenfalls durchnavigieren.

Es gilt die Microsoft GUI Richtlinien zu beachten, dafür sind die Microsoft Office Programme ein gutes Beispiel.

In welchen Fällen werden Tastaturkürzel-Adaptationen gemacht?
Bei «schwachen» Benutzern wurden bis vor einiger Zeit z.B. Programm-Starts mit speziellen Tastaturkombinationen gemacht; so wurde mit «Alt + Shift + W» Word gestartet. Dies war auch ein Standard, der sich unter den verschiedenen Organisationen verbreitet hatte.

Heute fördert und lehrt man aber die Variante des Programmstarts über das Start-Menu – auch wenn natürlich über die Systemsteuerung immer noch solche speziellen Zugriffe definiert werden können.

Wann werden andere Anpassungen nötig, z.B. Skripte?
Geskriptet wird nur, wenn etwas von Natur aus überhaupt nicht zu erreichen ist und es absolut notwendig ist. Das Grösste Problem dabei ist aber, dass solche Skripte oft nur einen grösseren Release lang halten. Verändert sich die ID des entsprechenden Fensters, wird es verschoben etc. dann beginnt man schnell wieder vorne.

Welche Rolle spielen Mnemonics?
Mnemonic, auch Kurztasten genannt, dürfen nicht überbewert werden. Niemand kann sich hunderte von Mnemonics im Kopf behalten. Diese haben generell zweite Priorität. Viel wichtiger Ist, dass alle Funktionen mit der Normalen Tastaturnavigation zu erreichen sind. Da gibt es mehrere Punkte zu beherzigen:

Alle Funktionen und Elemente sollen ohne Kurztasten erreichbar sein. Eine Grundmenge von Windows-Shortcuts soll übernommen werden. Alles was programmspezifisch ist, kann mit Kurztasten belegt werden, aber die Programmnutzung muss ohne Mnemonics funktionieren. Noch einmal: Ein gutes Programm ist das Programm, bei dem mal alle weiteren Funktionen mit der Tastatur erreichbar sind. Bei Programmen mit Toolbars sind diese oft ein Problem: ein einfaches Tastenkürzel ist nicht genug – weil das Tastenkürzel es einzig erfassbar wird, in dem man die Dokumentation des Produktes liest. Daher ist zu jedem Werkzeug ein Weg über das Menü nötig.

Wie sieht die optimale Tabreihenfolge aus?
Das ist relativ klar: sie muss funktional richtig sein, sie ist also prozessabhängig. Wichtig ist, dass z.B. Wechsel von Inhalt zu Übersicht/Inhaltsverzeichnis klar geregelt ist, was oft mit F6 passiert (Unterfenster-Navigation).

Alles muss erreichbar sind. Eigenschaften von einem Buch, z.B. 337 Seiten, Autor Goethe, die sich dem Sehenden sofort erschliessen, müssen auch mit dem Fokus erreichbar sein. Deshalb muss der Fokus nicht nur auf «editierbare Felder» gelegt werden. Die Fokusverfolgung ist ein wichtiges Element.

Was sind Hindernisse in der aktuellen Software?
Ribbons [Menü-Bänder] werden in der Blindenszene nicht sehr geschätzt. Auf Grund der grossen Bildschirme hat man vor einigen Jahren angefangen, diese Aufzufüllen mit Bildchen in Ribbons. Das macht für den Sehenden auch Sinn,

er kann diese Auf seinem Monitor ausbreiten und erhält dadurch den Überblick – genau das, was den Blinden und Sehbehinderten immer fehlt. Die GUIs sind so ausgelegt für einen grossen Teil der Nutzer, die mit wenig Wissen über ein Programm möglichst viel machen sollten. Für Blinde und Sehbehinderte sind mehr Hierarchiestufen in der Regel besser, denen hilft es sogar über drei oder vier Hierarchiestufen zu navigieren. Dies erfordert aber ein abstrakteres Denken, dass so bei den Sehenden manchmal nicht optimal vorhanden ist und daher mit flachen Hierarchien gearbeitet wird.

Welches sind die am häufigsten verwendeten Screenreader in der Schweiz?
JAWS ist mit einer Installationsquote von ca. 95% unter Blinden und Sehbehindertenin der am weitesten verbreitete Screenreader der Schweiz. Diese Proportion nimmt er wohl in der ganzen westlichen Welt ein, selbst in den USA, wo noch vor ein paar Jahren das Verhältnis zwischen *JAWS* und Window Eyes nahezu 50:50 war. Heute ist Window Eyes eher auf dem sterbenden Ast.

JAWS, das Produkt von *Freedom Scientific*, unter anderem auch wegen seiner Anpassungsmöglichkeiten Marktführer. Wir bei Accesstech arbeiten aber auch aus Effizienzgründen hauptsächlich damit, um sich nicht zu verzetteln.

NVDA, ein OpenSource Screenreader, ist der Favorit nach *JAWS*. Da *JAWS* in Kombination mit Vergrösserung (auch mit *JAWS* MAGic noch nicht 100% perfekt) nicht ideal ist, kommt dafür oft ZoomText zum Zug. Daneben gibt es auch noch Dolphin SuperNova, eine eierlegende Wollmilchsau, die weder in der Vergrösserung noch im Screenreading top ist, aber ein gutes Produkt ist, für Leute die nicht komplett Blind werden. Auch ist Cobras der deutschen Firma Baum eine Erwähnung wert.

Wie lange werden User von Ihnen geschult?
Wenn es sich um eine Schulung z.B. für einen Bürojob handelt, dann werden Schulungen fast immer von der IV bezahlt – denn das wichtigste daran ist, die entsprechende Person im Beruf zu behalten. Natürlich muss die Schulung und das daraus resultierende Ergebnis in einem Verhältnis stehen. Im beruflichen Bereich ist es weniger schiwerig zu erwirken, dass die IV die Kosten für Schulungs-Stunden übernimmt. Bei nicht enorm begabten Leuten ist es aber schwierig, dass sie ihren Job behalten – und es wird immer schwieriger. Das ist ein Problem im Moment. Früher hatten Grossfirmen ein «Sozial-Budget», die Swisscom hatte 55 bis 60 Telefonisten-Stellen für solche Leute, auch grosse Versicherungen stellten ähnliches zur Verfügung. Diese Stellen sind aber in den letzten Jahrzehnten verschwunden.

Schaut man sich in Europa um, dann sieht man schnell, dass die Schweiz das einzige Land ist, das keine Regelungen in diesem Bereich hat. Die Schweiz ist da sehr liberalistisch. Die Hoffnung, dass in der Schweiz mit dem Beitritt zur UNO-Behindertenrechtskonvention ein gewisser Druck entsteht, ist da.

Eine Vollbeschäftigung ist heute nicht mehr das Ziel. Ein Sockel existiert, das sind etwa 5% der Leute im erwerbsfähigen Alter, die es einfach Schwierig haben.

Im privaten Rahmen werden grundsätzlich 35 Stunden zur Schulung in IT von der IV bezahlt. Um mehr zu erhalten ist oft ein Kampf.

Gibt es noch «Nice-to-knows» zum Thema?
Für Windows ist die Zugänglichkeit für Blinde seit Beginn nur ein Nebenprodukt. Die GUIs wurden zugänglich gemacht, um GUIs automatisiert zu testen! Wie so häufig ist oft ein zusätzlicher Nutzen nötig, damit eine Anwendung wirklich les- und navigierbar wird.

Für ein optimales Arbeiten mit dem Computer sind immer die drei Elemente ausschlaggebend: Applikation – Screenreader – Benutzer respektive seine Fähigkeiten, einen Computer zu bedienen. Die wenigsten User können ihren Screenreader bis ins letzte Auskosten. Wäre dies der Fall, dann würde das auch schon viele der derzeit bestehenden Probleme lösen.

Vielen Dank.

Treffen mit Selamet Aydogdu, Interview und Wissensaustausch

Am 19. Mai 2015, im Bahnhof Aarau.

Selamet ist Vater einer zweieinhalb-jährigen Tochter, vollblind, arbeitet beim Kanton Aargau in Aarau und ist zuständig für IT Entwicklung und gewisse Elemente im Bereich Accessibility, er codiert selbst Java und HTML und kennt vor allem Accessibility Elemente, die im Web nötig sind. Er war als Junge an der Blindenschule Zollikofen und hat danach zuerst eine KV-Lehre und anschliessend eine verkürzte Informatiker-Lehre (2 Jahre) gemacht.

Zitat
«Du kannst von Glück reden, dass *beook* auf Eclipse RCP konzipiert ist, sonst könntest du bei jedem anderen Java-Framework nochmal von vorne anfangen.»

Gesprächsnotiz / Interview

Wie arbeitest du?
Ich arbeite immer kombiniert mit Knopf im Ohr für die Sprachausgabe plus Textinput über die Braillezeile. Um Dinge zu überfliegen ist die Sprachausgabe viel schnelle, und auch um Kontext-Informationen zu erhalten. Zum Beispiel was die Hintergrundfarbe ist, ob es sich um einen Titel handelt oder nicht – das zeigt mir die Braillezeile nicht an. *ARIA* Live-Regions, die Zusatzinformationen enthalten, werden akustisch wiedergegeben. Wenn sich in einer Live-Region etwas ändert, und der Fokus der Braillezeile nicht in jenem Bereich liegt, dann kann ich das nur über die Sprachausgabe wahrnehmen. So brauche ich die beiden einfach kombiniert. Du siehst, ich arbeite eigentlich nicht mit der Maus.

In Ausnahmefällen aber kann man die Maus trotzdem benutzen. Dazu wechselt man vom PC-Cursor, der im Normalfall verwendet wird, zum *JAWS*-Cursor, mit dem man dann via die Tastatur den Mauszeiger von Objekt zu Objekt springen lassen kann und die Maus auch Pixelgenau steuern kann – damit ist man dann aber schon langsam. In beiden Fällen ist es nicht effizient, es ist einfach ein Hilfsmittel für den Notfall.

Diese Funktionalität ist bereits in *JAWS* integriert, da braucht es nichts zusätzliches.

Wenn ich zum Beispiel in einem Textdokument mit der Maus navigieren würde, dann würde sie, wenn ich von links nach rechts steuere, buchstabenweise navigieren. Steuere ich von oben nach unten, dann navigiert sie zeilenweise.

In welchen Fällen bist du gezwungen, mit der Maus zu navigieren?
Es gibt Applikationen, in denen der OnKeyDown-Event etwas anderes bedeutet als der OnMouseDown-Event beispielsweise auf gewissen Schaltern. Es gibt auch eine Funktion, welche die Maus automatisch auf den Fokus ziehen kann. Dies funktioniert aber auch nicht immer zuverlässig. Ich würde auch sagen, ich bin eher einer der wenigen Anwender, die überhaupt von dieser Funktionalität gebrauch machen. Ganz selten sind einfach Elemente mit der Tastatur in keinster Weise erreichbar, dann kann bleibt diese Möglichkeit.

Normalerweise sollte man mit Tab durch die Elemente hindurch navigieren können. Die Tab-Reihenfolge ist elementar um in einer Anwendung zu navigieren. Bei Dialogboxen muss man sich oft auch mit Tab zurecht finden. Im Web ist es sogar noch häufiger nötig, sich mit Tab zu orientieren.

...Tab kommt also immer zum Einsatz, wenn du nicht genau weisst, was dich erwartet?
Genau.

Wenn du auf eine Website kommst, wie sieht für dich die optimale Tabreihenfolge aus?
Die optimale Reihenfolge ist immer die, welche strukturell Sinn macht. Was schaue ich als erstes an? Das Menü, dann kommt der Hauptinhalt, und erst dann die Fusszeile. Es gibt Beispiele wo es wild umherspringt wenn man sich «durch-tab-bed», das ist natürlich schlecht. Am besten stellst

du es dir so vor: wenn du kein CSS hättest, wie würdest du deine Informationen gliedern. Ohne CSS. Was ist die Reihenfolge, die deine Informationen hätten? Dein nacktes *HTML*... Auf das geht auch grundsätzlich *JAWS*, so zu 99%. Das ist ausschlaggebend.

Und wie ist deine Erfahrungen bei normalen Anwendungen?
Da habe ich viele gute Erfahrungen gemacht. Wenn man zu beispielsweise verschiedene «Tabs» nebeneinander hat, kann man mit «Ctrl + Tab» zum nächsten Tab springen, bzw. «Ctrl + Shift + Tab» zum vorherigen. Mit «Tab» und «Shift + Tab» bewegt man sich dagegen innerhalb des Tabs. Das sind wohl die elementarsten Navigationsmechanismen.

Wenn man zum Beispiel eine linke Spalte hat mit einem Inhaltsverzeichnis und rechts den Hauptinhalt hat, kann man mit «F6» oder «Ctrl + F6» zwischen diesen Teilen hin und her switchen. Das funktioniert sehr zuverlässig. Das ist ein Standard der von fast allen so gehalten wird. «Standard» ist halt immer das, was konventionell ist, wenn man so sagen will.

Menüs kann man mit Kurztasten aufrufen. Zum Beispiel «Alt + D» für «Datei», «Alt + B» für «Bearbeiten» – also alles Kombinationen mit der Alt-Taste. Dies ist der Windows-Standard für Accesskeys. Oder «Ctrl + Shift + irgendetwas» sind auch oft verwendet. Und nur «Ctrl». «Alt + Shift» kommt weniger vor, das ist etwas umständlich zum greiffen.

Im Web werden zum Teil auch Accesskeys definiert. Ich bin mal auf den Fall gestossen, wo «Alt + F» als Accesskey definiert war. Und du siehst, was da passiert: Es kommt in einen Klinch mit dem Accesskey des Browsers für die Favoriten. Wenn ich da die Favoriten aufrufen möchte, habe ich Pech, denn der Accesskey hat Vorrang. Da kann ich einzig über «nacheinander drücken» den Accesskey umgehen, in dem ich zuerst «Alt» und dann «F» drücke und so zu den Favoriten komme. Es ist also wichtig zu schauen, dass es da keine Überschneidungen gibt.

Wie sieht es aus mit Überschneidungen bei JAWS?
JAWS hat sehr viele Tastenkombinationen. Aber die setzen sich immer aus «Insert + etwas» zusammen. Zum Beispiel «Insert + F», «Insert + E» und so weiter, auch Kombinationen mit «Insert + Shift + etwas» gibts. Sie haben geschaut, welche Tastenkombinationen nirgens genutzt werden, sprich mit welchem Modifikator, und darum haben sie «Insert» als Modifikator gewählt. Mit dieser «Insert»-Strategie habe ich noch nie Probleme gehabt, es sind bei mir noch nie Konflikte aufgetaucht.

Ganz ehrlich, wieviele der JAWS-Shortcuts brauchst du? Wenn man sich die Liste anschaut, das ist ja unglaublich, dass man die alle kennen kann ...
Ja die Liste ist extrem lang. Ich brauche davon aber nur wenige. Es gibt Tastenkombinationen die nur für den Browser gelten, die braucht man. In Word brauche ich vielleicht ein bis zwei *JAWS*-Tastenkombinationen, nicht mehr. Die weil Word selbst ja Tastenkombinationen hat. Da brauche ich die *JAWS*-Tastenkombinationen, um zum Beispiel Schriftart und Schriftfarbe abzufragen. Also um an die Metainformationen zu kommen.

Einzelne Einstellungen lassen sich auch direkt als eigenes Fenster einblenden, dafür brauche ich einige *JAWS*-Shortcuts. Dann kann ich diese Einstellungen direkt ändern. Für Windows Applikationen aber brauche ich selten *JAWS*-Shortcuts.

Aber wie gesagt, im Internet brauche ich sie sehr oft: «E» für «Eingabe», «F» für «Formularelement», «B» für «Buttons», «H» für «Titelelemente», «L» für «Listen». Mit «Insert + F7» kann man sich alle Links auflisten lassen, mit «Insert + F6» alle Titel.

Tastenkürzel gibt es ja viele Standardisierte. Hinterlegst du dir auch eigene?
Nein. Ich arbeite grundsätzlich mit den Standards. Eigene machen könnte man theoretisch schon. Aber das Problem entsteht, sobald man auf ein fremdes System kommt. Dort funktionieren die ja dann nicht.

Die einzige Situation, in der du eigene hinterlegen würdest ist also bei Konflikten mit anderen Tastenkürzeln, z.B. mit Shortcuts von *JAWS*?

Ja. Aber selbst dann gibt es bei *JAWS* die Möglichkeit, in die Funktion «Tastaturkombination durchreichen» aufzurufen. Damit wird die nachfolgende Tastaturkombination nicht von *JAWS* abgefangen sondern bis zur Anwendung weitergeleitet. Dies ist aber nur eine «Überbrückung». Auch in *JAWS* könnte man eigene Tastaturkombinationen hinterlegen. Das habe ich aber schon lange nicht mehr gemacht.

Auf welche Hindernisse bist du schon getroffen, die du als erfahrener User umgehen kannst, aber für einen Normal-User unüberwindbar sind?
Es gibt schon solche Hindernisse. Da kenne ich Hauptsächlich Java-Applikationen, deren GUI mit *Swing* programmiert worden sind. Das sind die Schlimmsten. Wir bei der Arbeit haben auch eine solche Applikation, und die kann selbst ich so nicht bedienen. Denn das *Swing*-Interface ist für *JAWS* ein weisses Blatt. Das heisst, die nötigen Informationen kommen nicht bis zu *JAWS*, da *JAWS* nur Windows-Komponenten erfassen kann. Die Java-Access-Bridge kann da ein Bindeglied sein, dass zwischen *JAWS* und der Java-Applikation eine Verbindung kreiert. Aber deren Installation ist nicht ganz einfach, ich musste da letzthin nach der Installation noch irgendwelche .dll Dateien in bestimmte Verzeichnisse kopieren.

Eclipse ist ja auch Java, aber läuft eben auf der RCP-Plattform und ist daher komplett zugänglich.

Das ist ja auch die Ausgangslage meiner Applikation ...
Ich sage es mal so: «Du hast Glück!» Denn wenn es anders wäre, dann hätte ich dir jetzt gesagt: «Das kannst du vergessen.» Weil sonst müsstest du wohl die Applikation neu bauen. Ich habe bisher keine andere Lösung gesehen.

Mit was für einem Browser arbeitest du? Und was gibt für dich den Ausschlag mit dem einen oder anderen zu arbeiten?
Ich habe Firefox und IE, ich brauche beides. Eigentlich bevorzuge ich Firefox, aber wenn es Webapplikationen gibt, mit denen Firefox nicht zu gange kommt, dann wechsle ich auf IE – IE ist halt Standard.

Ich merke, dass Firefox schneller ist, auch stehen für Firefox mehr Plugins zur Verfügung und Webstandards werden ein bisschen besser respektiert. Accessibility wird besser wiedergegeben, *HTML*5 besser unterstützt – Firefox ist halt wirklich ein bisschen besser.

Du bist vor kurzem von iPhone auf ein Android Smartphone umgestiegen. Wie war diese Umstellung?
Ich hatte schon vorher ein Android Tablet von Samsung, darum war die Umstellung für mich nicht so gross. Die Sprachausgabesoftware heisst statt VoiceOver einfach Talkback, dafür konnte ich eine eigene TTS-Engine hinterlegen. So verwende ich nicht jene von Google sondern die von Eloquent – das ist die gleiche wie ich auf dem Notebook nutze mit *JAWS*. Die Möglichkeit, eine andere TTS zu hinterlegen, gibt es bei iOS nicht.

Für mich ist halt Android viel offener. Die Umgebung ist dadurch viel besser konfigurierbar auf deine eigenen Bedürfnisse. Und was mir sehr gefällt ist der konfigurierbare Startbildschirm. So habe ich bei meinem diese Statusleiste oben, die dir ja Wlan, Signalstärke, Batterie und Zeit anzeigt, einfach weggenommen, das interessiert mich ja weniger. Die Grösse der Icons kann angepasst werden, du kannst Widgets platzieren, Musik kannst du einfach rüberspielen ohne iTunes.

Hast du auch von der Anzeige her etwas geändert, sprich einen Battery-Save-Modus um das Display auszuschalten?
Es gibt eine App, welche dir den Bildschirm komplett abdunkelt. Die habe ich und mit der spare ich viel Batterie. Dann habe ich aber auch eine Verknüpfung auf dem Startbildschrim zu dieser Einstellung, damit ich den Bildschirm doch auch wieder schnell einschalten kann, wenn ich jemandem etwas zeigen möchte.

Konntest du die ganze Konfiguration selbst machen?
Ja grundsätzlich schon. Die meisten Apps sind sehr kompatibel. Das einzige Problem derzeit ist, dass die beste Version von Android 4.4.4 ist. Die neuste, 5.1 macht noch Probleme. Ich habe ein Nexus, und auf meinem Gerät ist die Version 5.1 träge sobald die Sprachausgabe aktiv ist. Auch Drag'n'Drop-Gesten funktionieren nicht mehr so zuverlässig. Das war der Grund, warum ich wieder ein Downgrade auf 4.4.4 gemacht habe.

Für uns technisch begeisterte ist Android toll, für Normalanwender ist Android halt noch nicht ganz so überzeugend für die Blinden. Du musst freude haben am Basteln und nicht direkt verzweifeln, wenn mal etwas nicht funktioniert. Bei iOS funktioniert es einfach. Du hast ein Gerät, das läuft, und darauf kannst du dich verlassen. Ein Android läuft eigentlich auch, du musst es einfach richtig konfigurieren am Anfang.

Die Erfahrung hat gezeigt, dass es für viele eher schwierig ist. Ich habe einmal einen Event durchgeführt, um «Accessibility mit Android» vorzustellen. Das Interesse war da, die Leute fanden es toll, aber von jenen, die es dann wirklich probiert haben, haben es die meisten wieder aufgegeben und sind wieder auf iOS zurück. Es ist schade, dass Google dies bisher ein bisschen verpasst hat.

Es ist ja eigentlich schade, dass Google Elemente wie Talkback bereitlegt, aber dann doch nicht konsequent genug ist, um es «einfach» verwendbar zu machen ...
Das enttäuschende beginnt bereits beim Homescreen von Google. Da ist der Standard-Homescreen von Google schlechter zugänglich als einen Launcher eines Drittanbieters, den ich mir jetzt installiert habe. Der ist viel besser unterstützt. Wenn ich eine Verknüpfung vom Startbildschrim des Google-Homescreens löschen möchte, kann ich das nicht – weil mir das Lösch-Icon einfach nicht angezeigt wird. Mein «Nova-Launcher» der zeigt es hingegen an. Mit dem funktioniert es! Es ist bedauerlich das das Produkt von Haus aus schlechter ist als Drittsoftware. Ein zurückwechseln auf iOS steht für mich aber im Moment nicht zur Debatte – solange ich ein Gerät habe auf dem die Android-Version 4.4.4 läuft.

Selamet Aydogdu am Laptop
Er benutzt eine ALVA-Braillezeile, das Betriebssystem auf seinem Notebook ist Windows 7 und als Screenreader hat er JAWS. Die Braillezeile gibt während dem Aufstarten in kurz Bescheid, dass das USB angeschlossen ist, anschliessend ist aber noch kein Output zu erspühren bis zum Moment, wo die Anmelde-Maske zum User-Login kommt. In diesem Moment ist dann JAWS schon aktiv, damit der Anmelde-Dialog vorgelesen werden kann. Ihm bleibt nichts anderes übrig, als zu warten. Zwischen Windows-Login und Desktop muss JAWS noch einmal erneut gestartet werden, da auch noch Benuzerspezifische Properties geladen werden.

Wie so oft springen nach dem Erscheinen des Desktops Fenster auf. In diesem Fall ein Popup für das Update des Adobe Flash-Players. Mit «Tab» sucht er, was die Möglichkeiten sind. In diesem Fall gibt es nur den Button «Installieren». Mit «Alt + F4» kann er aber das Popup ohne Aktion schliessen.

Programmaufrufe macht er über das Startmenü, worin er eine direkte Eingabe über das Suchfeld macht.

Bei Dialogfeldern mit zum Beispiel den Buttons «OK» und «Cancel»: Da sagt die Sprachausgabe, dass es sich um Schalter handelt. Auf der Braillezeile hingegen sind einfach die zwei Begriffe nebeneinander aufgeführt, mit «blinkendem» Cursor auf dem Element mit dem Fokus. Ein klicken des entsprechenden Buttons ist direkt über die Braillezeile möglich. Bei Fortschirttsbalken werden die aktuellen Prozentwerte vorgelesen, nicht aber auf der Braillezeile angezeigt.

Für Eclipse musste er auch gewisse Tastenkürzel auswendig lernen. Zum Beispiel alle offenen Dateien anzeigen ist möglich über «Ctrl + Shift + E», damit öffnet sich ein Fenster, dass in einer Liste die entsprechenden Dateien anzeigt.

Öffnen und Aktivieren von Elementen macht sich in der Regel mit der Enter-Taste, die Leertaste dient vorallem zum Aktivieren und Deaktivieren von Kontrollkästchen. In gewissen Programmen kann eine Eingabe bzw. Bestätigung auch mit Leertaste gemacht werden. Es ist aber für ihn eher störend, wenn plötzlich die Leertaste die Funktion der Enter-Taste übernimmt. Das ist ganz klar eine Usability-Frage.

In einem Text ist mit der Braillezeile ein Cursor-Routing möglich ist. So kann der Cursor direkt über die Knöpfe auf der Braillezeile am entsprechenden Ort positioniert werden.

Es gibt keine Richtlinien für die F-Tasten, aber doch gibt es gewisse Konventionen: So benutzt man «F6» oder «Ctrl + F6» in der Regel, um zwischen den Fenstern zu wechseln, so denn man mehrere Fenster hat. «F1» ist immer Hilfe, «F3» ist zum Suchen. «Ctrl + F4» ist zum Schliessen eines Dokuments oder Fensters, mit «Alt + F4» schliesst man das Programm, oder auch um Dialogboxen zu schliessen.

Über «Alt» springt man in die Menüleiste. Darin kann er mit den Pfeiltasten navigieren, JAWS liest ihm die verschiedenen Punkte vor. Für Navigation im Menü-Band ist das Cursor-Routing mit der Braillezeile möglich, aber nicht optimal, da es bei einem Klick direkt das Menü öffnet und man so unter Umständen mit einem «Pfeil nach rechts» bereits in einem Untermenü landet. Dafür zeigt der blinkende Cursor an, auf welchem Buchstaben der Shortcut/Accesskey für jeden Menüpunkt liegt, sodenn einer definiert wurde.

In Menüs ist es für ihn wichtig, dass ausgegraute, sprich inaktive Menüpunkte als «nicht verfügbar» angegeben werden. Auch vorhandene Untermenüs mussen angegeben sein, so dass der Screenreader dies akustisch ausgeben kann.

In Eclipse-Menüs findet keine Indikation statt, wenn man am Ende einer Menü-Liste ankommt. Sie beginnt einfach wieder von oben. So wäre es für ihn Sehr wünschenswert, wenn ein kurzes, akustisches Signal das Erreichen des Endes der Liste angeben würde. Das würde er eine gute Orientierungshilfe finden. Seine Präferenz ist dafür klar, dass der Fokus bei einer Liste nach dem letzten Punkt gleich wieder zum Ersten springen sollte und man so nicht gezwungen wird, unter Umständen die ganze Liste wieder hochzunavigieren.

«ESC» muss dafür da sein, dass man aus jeglicher Situation raus kommt, auf die Grundstellung des Programms, unter Umständen durch mehrmaliges drücken der Escape-Taste. «ESC» soll einfach Ebene für Ebene zurück führen. Zum Beispiel in einem Untermenü führt «ESC» auf die nächst höhere Ebene, bis er schliesslich aus dem Menü heraus springt. Um von einem Untermenü auf die nächst höhere Ebene zu kommen benutzt Selamet Aydogdu aber im Normalfall den «Pfeil nach links».

Beim Codieren schaut er mit den Fingern, wie die Einrückungen des Codes liegen. Sehr dankbar ist in diesem Zusammenhang die Auto-Format-Funktion von Eclipse, welche die Einrückungen automatisch korrekt setzt. Um zu wissen, wo zum Beispiel ein «if»-Block anfängt und wo er endet, setzt er den Cursor zuerst beim «if», und scrollt dann via Pfeiltasten nach unten, während er mit der linken Hand auf der Braillezeile wartet, bis wieder ein Zeichen auf gleicher Höhe angezeigt wird. Dann weiss er, wo das schliessende Element des Blocks ist.

Wenn grafische Elemente semantische Informationen enthalten, die aber dann vom Screenreader nur mit «Grafik_211.png» angegeben werden, können diese in JAWS «beschriftet werden». Im JAWS Grafikbezeichner wird die Grafik beschrieben, zusätzlich kann ein Braillezeilen-Text hinterlegt werden, und diese Information wird dann abgespeichert. Deshalb: Auch Icons müssten Labels (oder ähnlich) haben, damit diese Selbstbeschriftung nicht nötig ist. So hat er sich in Eclipse zum Beispiel die Icons für den «Closed Folder», das «Package mit Error» und «Package mit Warning» beschrieben.

In Eclipse ist ja die Syntax-Analyse wichtig. So werden rot unterstrichene Fehler von JAWS als

«ungültig» ausgesprochen. Diese Information ist sehr wichtig, wird aber nur über die Sprachausgabe mitgeteilt. Die Braillezeile enthält diese Meta-Information nicht.

Wie im Windows Explorer werden auch in Eclipse in Listen mit aufklappbaren Elementen die Ebenen werden angesagt. Auch wird «vorgelesen» ob ein Listen-Objekt offen oder geschlossen ist. Das macht die Navigation einfach und klar.

Analyse von beook auf Barrierefreiheit

Analyse von beook auf dessen Barrierefreiheit betreffend Tastatur-Navigation, Screenreader und Farbschema								
Nr.	**Bezeichnung der Etappe oder des Bereichs**	**Screenshot(s)**	**Navigation mit Tastatur möglich?**	**Vorlesen/Akustisches Feedback (Screenreader) korrekt?**	**Shortcuts/Accesskeys vorhanden?**	**Kontrast/Farbschema korrekt übernommen?**	**Weitere Bemerkungen**	**Empfehlungen**
	Installationsprozess		Zur Installation muss nur das von beook zu verfügung gestellte Zip enpackt werden. Das ist möglich über WinRAR. Wählt der Benutzer den Windows-Installer (.exe das zum Download bereit steht), dann können zwar alle Dialoge durchlaufen werden, aber bei den Dialogen mit der Installtionsbeschreibung und bei der Lizenz kann – obwohl die Sprachausgabe es so angibt – nicht mit Enter weiternavigiert werden sondern nur mit Leertaste.	Dialog «Installationsbeschreibung» wird mit falscher Sprache vorgelesen. Springen von Textteil zu Textteil nicht optimal.	Ja, Standard-Dialog Keys	Der Windows-Installer respektiert das Windows-Farbschema. Text bei Lizenzvereinbarungen und beim Abschluss-Dialog wird nicht angezeigt bzw. nicht korrekt an das Farbschema angepasst.		
	Loading, Ladebildschirm	01	nein, auch kein Abbrechen möglich	Nichts. Hier sollte ein akustisches Signal zeigen, dass man am Laden ist, und auch, wann der Ladevorgang abgeschlossen ist. Dies gerade weil die Ladezeit je nach Gerät auch etwas länger sein kann.	Nein	Nein, ist ein Bild		
	Global		Alt + F4: Schliessen von beook möglich		F-Tasten haben keine Fuktion	Aktive Buttons sind nicht angepasst, inaktive hingegen schon (abgesehen von jenen in der Seitenleiste).		
	Global – Menüleiste		Die Menüleiste kann wie gewohnt über alt erreicht werden, anschliessend kann mit den Pfeiltasten navigiert werden.	In der Menüleiste werden die Menüs vorgelesen.	Automatisch sind die Anfangsbuchstaben des jeweiligen Menüpunkts die Accesskeys. Bei mehreren gleich lautenden Menüeinträgen springt man vom nacheinander vom einen zum anderen.	Ja		
	Global – Dialogfenster Anmeldung		Ja	Ja, alle Labels und der dazugehörige Text werden korrekt gelesen.	Nein	Nicht korrekt, der Text ist unlesbar.		
	Global – Dialogfenster Einstellungen		Ja, man kommt mit der Tastatur zu allen Menüpunkten und Eingabefeldern.	Nein, der Text in der rechten Spalte mit den Erklärungen lässt sich nicht vorlesen (man kommt durch konventionelle Tastenbefehle nicht zum Text).	Nein	Ja		
	Global – Seitenleiste		Über Tab kommt man immer in die Seitenleiste, in der mit den Pfeiltasten navigiert werden kann.	Die Bücher in der Seitenleiste werden schon akustisch ausgegeben, aber man kann nicht nachvollziehen, zu wem ein Buch gehört (optisch ist der Anbieter identifizierbar)..	Nein	Die Inhalte der Schaltflächen sind Bilder auf weiss – daher bleiben es Bilder.		
	Portal	02	Mit Tab springen von Symbolleiste zu Symbolleiste zu Inhalt möglich. In der Portal-Toolbar kann zwischen den Schaltern mit Pfeiltasten navigiert werden. Dagegen ist es nicht möglich, einen Anbieter auszuwählen.	Keine semanitsche Information vorhanden, in welchem Bereich man sich befindet. Die Schalter in der Portal-Toolbar heissen alle nur «Schalter». Die bestehenden Anbieter können über die Sprachausgabe nicht wahrgenommen werden.	Nein	Bild-Elemente für Kundenauswahl im Portal nicht angepasst, auch die Portal-Toolbar wird nicht angepasst..		
	Portal – Anbieter-Detail		In der Portalansicht eines Anbieters, kommt man über Tab nicht mehr in die Portal-Toolbar. Dafür aber zu den Registerkarten Detail, Download, Ressourcen. Die Liste zur Sprachauswahl der Bücher wird beim durchtabben auch übersprungen. Hingegen ist der Tab-Sprung auf Symbolleiste-Maximieren bei den Registerkarten nicht nötig.	Inhalte in Registerkarten können korrekt vorgelesen werden. Mit allen nötigen Links.	Nein	Bereiche im Inhaltsbereich sind nicht korrekt angepasst (u.a. Hintergrund gleiche Farbe wie Schrift)		
	Portal – Bedienungsanleitung, Feedback, Impressum		Ja, wie Browser	Ja	Die Browser JAWS-Accesskeys können verwendet werden.	Ja	Dies sind nur einfache HTML-Seiten	
	Inhaltsverzeichnis	05	Navigation ist zum Teil möglich, mit Tab kommt man von Element zu Element, aber nicht an alle. Auch hier ist ein Tab-Sprung auf Symbolleiste-Maximieren bei den Übersichstspalten nicht nötig. Komplett übersprungen wird die Lesebereich-Toolbar. Durch die Listen Inhaltsverzeichnis, Lesezeichen, Markierungen, Suchresultate und Übungen kann mit den Pfeiltasten navigiert werden.	Die Elemente werden grundsätzlich ausgegeben. Hingegen existieren bei den Labels der verschiedenen Übersichtsspalten keine zusätzlichen Informationen ausser «Leer, gewählt F1-Hilfe». Wählt man dann z.B. die Spalte «Inhaltsverzeichnis» ohne mit Enter zu bestätigen, wird direkt die Liste des Inhaltsverzeichnisses vorgelesen. Beim Vorlesen der Inhaltspunkte ist der Screenreader vom Laden des Leseteils abhängig. Die auf dem Listenpunkt stehende Bezeichnung nimmt er nicht. JAWS spricht erst den Menüpunkt aus, wenn der Inhalt im Leseteil auch geladen ist. Das kann, wenn dies länger dauert, verwirrend sein. Beim Sprung in den Lesebereich liest der Screenreader den «technischen Dokumenttitel» bzw. Datei-Namen (z.b. DEV-K00-Dokument). Anstatt den eigentlichen Titel (h1, h2 …).	Nein	Inhaltsverzeichnis und Lesebereich-Toolbar bleiben weiss hinterlegt.		Wählt man einen Eintrag im Inhaltsverzeichnis mit Enter, sollte der Fokus direkt auf den Titel des geöffneten Dokuments.
	Lesebereich		Navigation wie in einem Browser möglich.	Sprachausgabe des Inhalts generell gut. Grafiken ohne Alternativtext sind ein Problem. Zum Beispiel bei Links.	Ja, Standard-Browser-JAWS-Accesskeys.	Ja, ausser Grafiken und Bilder.		
	Verschicken (Markierungen, Notizen)		Navigation zum Schalter ist möglich, aber nicht Intuitiv	Schalter hat kein Label, daher kann man über die Sprachausgabe nicht wissen, wozu er dient.	Nein	Das Fenster, welches sich öffnet nach klicken den Schalters ist angepasst. Der Schalter selbst ist schlecht sichtbar auf		
	Suche		Tastaturnavigation ist möglich, auch durch die Ergebnisse hindurch.	Das Eingabefeld liefert keine Angaben, wozu es ist.	Nein	Partiell, da der Hintergrund des Inhaltsverzeichnis-Bereichs weiss hinterlegt ist, ist dort die weisse Schrift nicht lesbar.		Nach drücken von Enter für die Suche sollte der Fokus direkt auf den obersten Listenantrag springen.

Nr.	Bezeichnung der Etappe oder des Bereichs	Screenshot(s)	Navigation mit Tastatur möglich?	Vorlesen/Akustisches Feedback (Screenreader) korrekt?	Shortcuts/Accesskeys vorhanden?	Kontrast/Farbschema korrekt übernommen?	Weitere Bemerkungen	Empfehlungen
	Eigene Seite hinzufügen		Das Navigieren zum Button und das vorgängige Einstellen des gewünschten Ortes ist möglich. So kann eine neue Seite erstellt werden. Allerdings ist der Inhalt ebendieser Strukturell schwach, da nur über Quellcode mit Hierarchie-Ebenen (h1, h2 etc) gearbeitet werden kann. Auch kann nach der Bearbeitung nicht mit den normalen Browser-Accesskeys von JAWS navigiert werden, da der ganze Inhaltsbereich ein Einziges Eingabefeld ist und nur durch das angesteuert werden kann. Geht man bei eigenen Seiten im Inhaltsverzeichnis hin und drückt Enter, so öffnet sich der Dialog der Eigenschaften der Seite (wie wenn man eine neue Erstellt). Dies ist problematisch, wenn Enter das Öffnen der Seite sein soll.	Selbst erstellter Text kann vorgelesen werden, dem Schalter selbst fehlt das nötige Label, um zu wissen, für was er gut ist.	Nein, ein Accesskey zum Erstellen der Seite wäre aber sehr nützlich.	Ja, ausser dass der Button zum Seite Einfügen nicht sichtbar ist.		Im Inhaltsverzeichnis über eine Kurztaste neue Seite einfügen, damit ein zurücktabben nicht nötig ist.
	Schriftgrösse einstellen		Das Springen von Button zu Button zu Skala ist möglich. Schliessen des Popups mit ESC möglich.	Die Buttons sind nicht genannt, die Grösse beim Verschieben des Reglers auf der Skala wird nicht ausgesprochen.	Nein	Ja		
	Markierung setzen		Zur Lesebereich-Toolbar kann man derzeit mit der Tastatur nicht gelangen. Das Springen von Button zu Button zu Skala ist möglich. Schliessen des Popups mit ESC möglich.	Die Buttons sind nicht genannt, die Grösse beim Verschieben des Reglers auf der Skala wird nicht ausgesprochen.	Nein	Ja	Gesetzte Leuchmarkierungen sind mit angewendetem Hochkontrast-Farbschema nicht sichtbar.	Für in der Lesebereich-Toolbar liegenden Funktionen müssen alternativen bereit stehen, die während einer gemachten Auswahl oder einem gesetzten Cursor aktiviert werden können.
	Notiz setzen		Zur Lesezeichen-Toolbar kann man derzeit mit der Tastatur nicht gelangen. Notiz-Button kann zwar gewählt werden, aber die Notiz als soches kann nicht gesetzt werden. Hingegen wird beim Durchtabben durch den Inhalt ein im Inhalt stehender Notizzettel fokussiert. Die Navigation im Zettel ist ein bisschen mühsam: Tab wird im Inhalt zu einer Lücke, anstatt dass der Fokus weiterspringt.	Inhalte von Notizzetteln können zum Sprechen gebracht werden, allerdings ist es noch nicht Intuitiv genug.	Nein	Ja		
	Übungen global							
	Textaufgaben Einzeilig		Man kommt per Tastaturnavigation zum Eingabebereich, aber dann nicht in das Eingabefeld.	Als Label zum Eingabebereich wird die Frage vorgelesen, sodenn nicht schon eine Antwort gesetzt ist.	Ja, Standard-Browser-JAWS-Accesskeys, in diesem Fall ist E sehr nützlich.	Ja		
	Textaufgaben Mehrzeilig		Man kommt per Tastaturnavigation zum Eingabebereich, und auch in den Editor (Analog eigene Seiten). Selbst Bilder einfügen ist möglich. Ausfüllen von Tabellen in Eingabefeld schwierig.					
	Multiple-Choice/Mehrfachauswahl		Navigieren ist möglich, auch das Auswählen von Antworten.	Labels/Struktur nicht optimal.	Ja	Ja		
	Zeichnungseditor		Bis zum und in den Zeichnungs-Editor ist es möglich zu Navigieren. Da ist aber die Bedienung nicht mehr möglich. Es ist nicht möglich, den Editor über Esc zu verlassen.	Labels/Beschreibungstext fehlt	Nein	Nein	Einen Aufgabentyp, der nicht Barrierefrei gemacht werden kann für Blinde.	
	Lückentext		Es kann wie bei einem Formular hindurch navigiert werden.	Text wird fortlaufend gelesen. Insofern ist die Sprachausgabe hier gut. Weitere Labels würden stören.	Ja, Standard-Browser-JAWS-Accesskeys, in diesem Fall ist E sehr nützlich.	Ja		
	Drag'n'Drop Übungen (Zuordnung)		Über die Tastaturnavigation kommt man nicht an die zu draggenden Begriffe. Es bleibt unmöglich, diese zuzuordnen.	Die Sprachausgabe gibt Zuordnungs-Infos aus – die eigentliche Lösung.	Ja			
	Markieren		Das Navigieren zur Übung, das Auswählen der Markierung und das Setzen/Löschen der Markierung funktionieren über die Tastatur. Sprung von «Zurücksetzen» auf «vorheriges Kapitel»: hier wird der Inhalt übersprungen mit Tab.	Die Sprachausgabe gibt nicht an, welche Wörter markierts sind und daher auch nicht wie. Die Auswahlschalter zum Auswählen der Markierung geben nicht an, was sie sind (z.B. Rot für Nomen, Grün für Verben).	Nein	Ja, ausser die Markierungen, die schlicht ignoriert werden.		
	Satzzeichen		Auswählen des Satzzeichens und auch der Position möglich, aber das effektive, korrekte Setzen des Satzzeichen ist nicht möglich. Sprung von «Zurücksetzen» auf «vorheriges Kapitel»: hier wird der Inhalt übersprungen mit Tab.	Die Auswahlschalter zum Auswählen der Satzzeichen und Position geben nicht an, was sie sind. Einzig, ob sie aktiviert sind oder nicht, wird gesagt.	Nein	Ja		
	Gross-Klein		Navigation zur Übung möglich, aber Anwählen des Wortes, welches geändert werden muss, ist nicht möglich. Sprung von «Zurücksetzen» auf «vorheriges Kapitel»: hier wird der Inhalt übersprungen mit Tab.	Keine Erklärung vorhanden, wie man Navigiert.	Nein	Ja		Erklärung abgeben/einbauen, wie man genau Navigiert. Wie das auch der Fall ist bei Eingaben, Dialogen etc.
	Bildmarkierungen		Navigation zur Übung möglich. Theoretisch ist das Auswählen von Polygonen möglich, aber es hat kein Fokus und nichts, das angeben würde, auf welchem dass man sich befindet.	Sprachausgabe liest «Polygonen» vor.	Nein	Nein, da Bild/Canvas	Einen Aufgabentyp, der nicht Barrierefrei gemacht werden kann für Blinde.	
	Kreuzworträtsel		Navigation zur Übung möglich, aber das Navigieren im Kreuzworträtsel ist nicht möglich.	Die Sprachausgabe gibt nicht an, welche nummer in welche Zelle gehört, da die Nummern nur mit Bildern integriert sind.	JA, Standard-Browser-JAWS-Accesskeys, in diesem Fall ist L sehr nützlich.		Es gibt speziell für Blinde entwickelte Kreuzworträtsel-Apps, dies würde wohl aber den Umfang von Beook sprenen …	

Nr.	Bezeichnung der Etappe oder des Bereichs	Screenshot(s)	Navigation mit Tastatur möglich?	Vorlesen/Akustisches Feedback (Screenreader) korrekt?	Shortcuts/Accesskeys vorhanden?	Kontrast/Farbschema korrekt übernommen?	Weitere Bemerkungen	Empfehlungen
	Lösungen anzeigen		Zum Button kann derzeit nicht navigiert werden mit der Tastatur.	Die Korrigierten Werte werden wie die selbst gegebenen Antworten vorgelesen. Der Term «Eingabefeld» wird aber erst erwähnt, nachdem die Lösung gelesen wurde. Das ist verwirrend.	Nein	Ja		
	Korrigieren		Zum Button kann derzeit nicht navigiert werden mit der Tastatur.	Die Korrektur-Werte werde nicht vorgelesen (ob falsch oder richtig), da das Symbol (√ oder X) nicht interpretiert wird bei der Sprachausgabe				
	Auswertung Übungen		Zum Button kann derzeit nicht navigiert werden mit der Tastatur. Im geöffneten Dialog kann mit Tab und Pfeilen navigiert werden. Der Dialog kann nicht mit esc geschlossen werden, sondern nur über «Speichern» oder «Abbrechen»..	Ein Vorlesen der Auswertung ist nicht möglich (Inhalt wird nicht erkannt).	Nein	Ja		

CD mit Daten

Auf der beigelegten CD befindet sich der gesamte Rapport mit den kompletten Anhängen als PDF. Profitieren Sie von einer angenehmen Lektüre dank Lesezeichen und Hyperlinks.

Zusammenfassung

Die Ergebnisse von literarischer Recherche, dem Beobachten und Interviewen von blinden und sehbehinderten Computer-Benutzern, der Analyse von Standard-Tastaturschemas und einem Selbstversuch wurden zusammengesetzt um einen Empfehlungskatalog für die Entwicklung von barrierefreien Desktop-Applikationen zu erstellen.

Als praktisches Beispiel wurde die *beook*-Applikation für Windows analysiert, wobei sich gezeigt hat, dass durch das darunterliegende Framework bereits gewisse Elemente für Barrierefreiheit ausgelegt sind. Zu verbessern bleiben aber Navigation, Beschriftungen von Elementen und Bereichen, der Zugang zu Kontextmenüs und die Vergabe von Accesskeys.

Der Empfehlungskatalog, der pro Thema eine praktische Umsetzungs-Variante für *beook* enthält, wird als Grundlage für die Weiterentwicklung der *beook*-Applikation verwendet. Um die Erkenntnisse einer breiten Masse bereitzustellen, wurden die Empfehlungen auf dem Blog von *Access for All* publiziert. Zudem soll die Publikation auch zu Inputs, Kommentaren und Kritik anregen, um die Qualität der Empfehlungen zu erhöhen.