DiKo
Details im Kontext anzeigen

Bachelorarbeit

von

Felix Benz-Baldas

Matrikelnummer: [redacted]

Studiengang: Informatik

An der Fakultät für Wirtschaftswissenschaften

Institut für Angewandte Informatik und

Formale Beschreibungsverfahren

Gutachter:	Prof. Dr. Rudi Studer
Betreuender Mitarbeiter:	M.Sc. Tobias Weller
Eingereicht am:	15.03.2017

Eidesstattliche Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, den 15. März 2017 *Vorname Nachname*

Zusammenfassung

Ein Großteil der Interaktion mit der grafischen Benutzeroberfläche hat mit dem Anzeigen neuer Inhalte zu tun. Wenn die Benutzeroberfläche sich sehr verändert, muss der Benutzer sich neu orientieren, was einen mentalen Aufwand bedeutet.
Das Ziel dieser Arbeit ist es, den mentalen Aufwand beim Anzeigen neuer Inhalte zu minimieren. Die neuen Inhalte sollen in die bestehende Ansicht integriert werden, anstatt diese zu verdecken. Der Kontext soll erhalten bleiben.
Es wurde ein neuartiges Benutzeroberflächenkonzept entwickelt. Dieses wird vorgestellt, erklärt und untersucht. Das Konzept wurde implementiert um dessen praktische Umsetzbarkeit zu untersuchen. Außerdem wird es mit ähnlichen Techniken verglichen.
Es wird gezeigt, dass das neue Konzept Vorteile gegenüber diesen Techniken hat. Es kann für viele Informationsstrukturen angewendet werden, darunter Dokumente, Hypertexte und semantische Daten. Bei der Untersuchung wurden konkrete Lösungen für einzelne Probleme gefunden, welche ebenfalls präsentiert werden. Diese sind auch für die weitverbreite Baumansicht relevant.

Inhaltsverzeichnis

1 Einleitung

Bei der Recherche mit Wikipedia interessiert man sich häufig nur für einen kleinen Teil eines Artikels, muss sich jedoch trotzdem den ganzen Artikel in einem eigenen Fenster anzeigen lassen. Das Fenster verdeckt dann unnötigerweise wichtige Objekte auf der Benutzeroberfläche, was zum Verlust der Übersicht führen kann.
In dieser Arbeit wird ein grafisches Benutzeroberflächen-Konzept vorgestellt und untersucht, welches es ermöglicht, neue Inhalte in eine bestehende Ansicht zu integrieren. Dadurch werden vorhandene Inhalte nicht verdeckt und der Kontext bleibt erhalten.
Im folgenden Abschnitt 1.1 werden die grundlegenden Anforderungen an das neue Konzept motiviert.
Abschnitt *Semantisches Skalieren* 1.2 stellt eine Vorüberlegung dar. Darauf aufbauend wird in Abschnitt 2 das neue Konzept erklärt. Dieses wird *DiKo* (= **D**etails **i**m **Ko**ntext) genannt. Es wird zunächst theoretisch gezeigt, welche Informationsstrukturen mit DiKo dargestellt werden können.
In Abschnitt *Implementierung* 3 wird dann die praktische Umsetzbarkeit gezeigt. Es wird untersucht, inwiefern DiKo die Anforderungen erfüllt. In Abschnitt 4 wird das neue Konzept mit anderen Techniken und Konzepten verglichen, welche ebenfalls kontexterhaltendes Anzeigen von Inhalten ermöglichen.
Eine Einordnung in die wissenschaftliche Landschaft erfolgt in Abschnitt 6. Insbesondere wird der Zusammenhang zu ausgewählten Publikationen und Konzepten aus der Informationsvisualisierung gezeigt. Das Design von DiKo wird anhand von Erkenntnissen aus der Mensch-Maschine-Interaktion diskutiert.
Bei der Implementierung wurden Lösungen für einzelne konkrete Probleme gefunden, welche nicht nur für DiKo, sondern auch für die Baumansicht (siehe Abschnitt 4.3) relevant sind. Diese werden in Abschnitt *Problemlösungen* 5 vorgestellt.

1.1 Motivation

In den folgenden beiden Abschnitten werden die grundlegenden Anforderungen an das neue Konzept DiKo motiviert und erklärt.

1.1.1 Bedarf an Details

Bei der Arbeit mit einer grafischen Benutzeroberfläche ist es eine grundlegende, ständig wiederkehrende Operation, neue Objekte (zum Beispiel ein neues Fenster) anzeigen zu lassen. Der Begriff *Detail* wird hier in einem sehr allgemeinen Sinn verwendet. Es wird gezeigt, dass die neuen Objekte meist als Details von Objekten, welche zuvor auf der Benutzeroberfläche zu sehen waren, aufgefasst werden können.
Es folgen verschiedene Möglichkeiten, Objekte detailreicher darzustellen bzw. weitere De-

tails anzuzeigen. Objekte können auf grafische Weise vergrößert werden. Dies entspricht dem grafischen Zoomen eines Objektes. Es gibt aber auch die Möglichkeit Objekte *semantisch* zu vergrößern. Dies bedeutet, dass das Objekt durch ein anderes Objekt ersetzt wird, welches das gleiche logische Objekt repräsentiert, jedoch mehr Inhalt bietet und mehr Platz auf der Benutzeroberfläche einnimmt[1]. Eine weitere Möglichkeit ist das Anzeigen von zusätzlichen neuen Objekten. Diese neuen Objekte können Details (im engeren Sinne) des Objektes darstellen. Ein Beispiel wäre ein Link in einem Text, bei dem der verlinkte Inhalt das Link-Wort erklärt. Wie bereits erwähnt, wird der Detail-Begriff hier jedoch verallgemeinert. Als *Details* eines Objektes werden alle Objekte bezeichnet, welche *über* diesen Begriff erreicht wurden. Öffnet sich ein Fenster, nachdem man auf ein Link-Wort geklickt hat, wird das Fenster hier als *Detail* des Link-Wortes bezeichnet. Weiter unten in diesem Abschnitt folgen Beispiele, welche den Detail-Begriff veranschaulichen.
Das Objekt, zu welchem der Benutzer weitere Details anfordert, kann einen ganz anderen Zweck haben, als das Portal für weitere Objekte (die Details) zu sein[2]. Außerdem kann das Objekt auf der Benutzeroberfläche in eine größere Struktur eingebunden sein. Ein Beispiel sind Link-Wörter in einem Fließtext. Pop-Ups[3] stellen eine Möglichkeit dar, zu beliebigen Objekten auf der Benutzeroberfläche weitere Details anzuzeigen.
Das Anzeigen von Details ist ein *nicht-linearer* Prozess. Zum Beispiel möchte man bei der Recherche mit Hypertexten[4] im Allgemeinen nicht zuerst einen Artikel komplett lesen und danach die verlinkten Inhalte. Statt dessen möchte man flexibel manchen Links folgen und manchen nicht.
Der Bedarf an Details ist häufig nicht gedeckt, wenn Details zu einem Objekt angezeigt wurden. Es kann sein, dass der Benutzer Details zu einem Objekt, welches selbst Detail eines anderen Objektes ist, anfordert. Beispiel: Der verlinkte Inhalt eines Link-Wortes kann wiederum Hypertext sein. Es zeigt sich hier ein *rekursives* Prinzip.
Es folgen Beispiele zur Veranschaulichung.
Ein bekanntes Hypertext-System ist Wikipedia. Die Details der Link-Worte in einem Wikipedia-Artikel sind hier meist wiederum Wikipedia-Artikel. Beim Recherchieren geschieht es häufig, dass man mehreren Links *rekursiv* folgt.
Eingangs wurde der Detail-Begriff verallgemeinert. Dies soll hier veranschaulicht werden: Auch Bearbeitungsfenster stellen Details (im verallgemeinerten Sinn) eines Objektes dar. Klickt der Benutzer zum Beispiel auf den Namen einer Person in einem Kurznachrichtendienst, wäre es denkbar, dass sich ein Fenster öffnet, in dem der Benutzer eine Nachricht an die entsprechende Person senden kann. In diesem Beispiel wäre das Fenster zum Schrei-

[1] Dies wird auch als semantisches Zoomen *eines* Objektes bezeichnet. Siehe auch `http://www.infovis-wiki.net/index.php/Semantic_Zoom`

[2] Ein Element einer Menüleiste hat offensichtlich die Funktion, weitere Details (das *Drop-Down-Menü*) anzuzeigen. (Eine Definition des Wortes Drop-Down-Menü findet sich im Online-Duden: `http://www.duden.de/rechtschreibung/Drop_down_Menue`)

[3] siehe Abschnitt 4.1

[4] Wikipedia stellt zum Beispiel ein Hypertext-System dar.

ben der Nachricht ein Detail des Namens.
So gesehen ist ein Großteil der Interaktion mit der grafischen Benutzeroberfläche das (rekursive) Anzeigen von Details. Das Ausblenden von Details wird hier nur am Rande untersucht und wird als analog zum Anzeigen von Details betrachtet.

1.1.2 Kontexterhaltende Interaktion

In diesem Abschnitt soll gezeigt werden, dass *Details* möglichst auf *kontexterhaltende* Weise angezeigt werden sollen. Es wird hier sowohl die Kontexterhaltung auf visueller Ebene, als auch auf logischer Ebene betrachtet.
Auf visueller Ebene bedeutet Kontexterhaltung, dass hinreichend viele visuelle Ankerpunkte[5] auf der Benutzeroberfläche durch die eingeblendeten Details weder verdeckt werden noch verrutschen. Zur Veranschaulichung dient hier das Beispiel von kleinen *Pop-Ups*[6]. Mit kleinen Pop-Ups können Details so angezeigt werden, dass der größte Teil der Benutzeroberfläche unverändert bleibt – das heißt die meisten visuellen Ankerpunkte sind weiterhin zu sehen. Ein Problem an Pop-Ups ist jedoch, dass sie einen Teil der Benutzeroberfläche verdecken.
Auf logischer Ebene bedeutet Kontexterhaltung, dass der Benutzer die Übersicht behält. Er weiß stets, wie die sichtbaren Inhalte in den übergeordneten Kontext einzuordnen sind. Wenn der Betrachtungsgegenstand eine größere Informationsstruktur ist, bedeutet Kontexterhaltung, dass der Benutzer stets weiß, wo er sich gerade befindet. Er versteht den Zusammenhang zwischen den neuen Details und den Inhalten, welche zuvor auf der Benutzeroberfläche zu sehen waren.
Häufig sind Details nur verständlich, wenn sie im Kontext angezeigt werden. In einem Inhaltsverzeichnis kann es zum Beispiel sein, dass die Gliederungspunkte, welche sich in tieferen Ebenen befinden, ohne übergeordnete Gliederungspunkte ihre Bedeutung verlieren. (In diesem Beispiel stellt ein Gliederungspunkt ein Detail des übergeordneten Gliederungspunktes dar.)
Kontexterhaltung bedeutet auch, dass der Benutzer schnell überall hin navigieren kann. Auf visueller Ebene helfen ihm dabei visuelle Ankerpunkte. Auf logischer Ebene hilft ihm dabei die Übersicht über die Gesamtstruktur.
Die Gefahr den Kontext zu verlieren besteht vor allem dann, wenn neue Inhalte auf der Benutzeroberfläche erscheinen. Im vorhergehenden Abschnitt wurde gezeigt, dass die neuen Inhalte meist als *Details* von Objekten, welche zuvor auf der Benutzeroberfläche zu sehen waren, aufgefasst werden können. Deshalb ist Kontexterhaltung beim Anzeigen von Details besonders wichtig.
Diese Arbeit untersucht Konzepte, mit denen Details auf kontexterhaltende Weise ange-

[5] Als visuelle Ankerpunkte sind hier allgemein Objekte auf der Benutzeroberfläche gemeint, welche dem Benutzer helfen sich auf ihr zu orientieren.

[6] Siehe auch Abschnitt 4.1

zeigt werden können.

1.2 Semantisches Skalieren

Semantisches Skalieren bedeutet, dass der Benutzer sich ein Objekt unterschiedlich detailliert anzeigen lassen kann. Das *Hochskalieren* bewirkt, dass die Darstellung mehr Platz einnimmt. Gleichzeitig bietet sie mehr Inhalt.[7] Auch das Anzeigen (oder Ausblenden) weiterer *Details* zu einem Objekt, wird als semantisches Skalieren bezeichnet.[8]
Durch das semantische Skalieren können also neue Objekte (Details eines anderen Objektes) auftauchen. Die neu erschienen Objekte können wiederum semantisch skaliert werden. Dies wird als *rekursives* semantisches Skalieren bezeichnet.
Das semantische Skalieren ist also eine Technik, um den in Abschnitt 1.1.1 thematisierten Bedarf an Details zu decken.
Eine positive Eigenschaft des semantisches Skalierens hat mit der Platzausnutzung auf der Benutzeroberfläche zu tun. Durch das stufenweise Hochskalieren wird das Objekt nicht unnötig detailliert dargestellt. Statt dessen werden nur so viele Details angezeigt, wie der Benutzer auch sehen möchte. Unnötiger Platzverbrauch wird verhindert.
Meist kann der Bedarf an Details durch eine kleine Darstellung der Details gedeckt werden. Folgendes Beispiel soll dies veranschaulichen: Bei Online-Zeitungen werden Artikel auf der Übersichtsseite meist *komprimiert* dargestellt. Es sind lediglich die Überschriften und die ersten Sätze zu sehen. Dies reicht, um sich einen Überblick zu verschaffen und zu entscheiden, ob man den gesamten Artikel lesen möchte. Erst wenn man auf den Artikel klickt, wird der komplette Artikel angezeigt.
Das *rekursive* semantische Skalieren ermöglicht es, dass der Benutzer sich nur das anzeigen lässt, was ihn auch interessiert. Dies trägt ebenfalls dazu bei, dass die Darstellung nicht unnötig viel Platz einnimmt.
Durch die gute Platzausnutzung bleibt mehr Platz für andere wichtige Objekte. Insbesondere gibt es mehr Möglichkeiten für wertvolle visuelle Ankerpunkte, welche helfen, den Kontext zu erhalten (siehe Abschnitt 1.1.2).

[7] Beim *Herunterskalieren* wird die Darstellung kleiner und es wird weniger Inhalt präsentiert.

[8] In Abschnitt 6.1.2 unter *Zoom* wird gezeigt, dass die Begriffe Zoomen und dementsprechend auch der Begriff *Semantisches Zoomen* nicht eindeutig definiert sind. Deshalb wird hier der weitgehend ungeprägte Begriff *Semantisches Skalieren* definiert und verwendet.

2 DiKo

DiKo ist ein grafisches Benutzeroberflächen-Konzept, welches *kontexterhaltendes semantisches Skalieren* ermöglicht. Insbesondere ist es möglich, *rekursiv* Details auf kontexterhaltende Weise anzuzeigen. DiKo kann für alle Objekte bzw. Informationsstrukturen verwendet werden, welche mit einem *DiKo-Objekt* (wird gleich erklärt) dargestellt werden können.
Abbildung 4 in Abschnitt 3 zeigt einen Screenshot von DiKo. Die theoretischen Grundlagen von DiKo werden im Folgenden besprochen. Dazu werden DiKo-Objekte definiert, es wird die Funktionsweise von DiKo erklärt und anschließend wird gezeigt, dass Objekte (bzw. Informationsstrukturen) häufig als DiKo-Objekt dargestellt werden können.

2.1 DiKo-Objekte

Ein DiKo-Objekt bietet Darstellungen eines Objektes in unterschiedlichen semantischen Skalierungsstufen[9]. Es hat folgende Eigenschaften:

E1 – Feingranulares semantisches Skalieren
In der *ersten* Skalierungsstufe wird das Objekt in einer sehr kleinen Fläche dargestellt, zum Beispiel mit einem Wort, einem kurzem Text oder einem Icon. Auch in der *zweiten* Skalierungsstufe wird das Objekt noch in einer relativ kleinen Fläche dargestellt. Es ist möglich, das Objekt noch weiter hochzuskalieren (je nach Informationsstruktur hat das weitere Hochskalieren jedoch keinen Effekt mehr).

E2 – Anpassung an verfügbare Breite
Wird das DiKo-Objekt hochskaliert, wird der *verfügbare*[10] Platz in der Horizontalen sinnvoll genutzt. (Für die Funktionsweise von DiKo ist die Platzausnutzung nicht notwendig.) Die verfügbare Breite wird nicht überschritten.

E3 – Enthaltene DiKo-Objekte
Wenn das Objekt weitere Objekte enthält, werden diese wiederum mit einem DiKo-Objekt dargestellt. Es kann an der entsprechenden Stelle Platz für die Skalierung des enthaltenen DiKo-Objektes gemacht werden. Wenn das enthaltene DiKo-Objekt hochskaliert wird, erhält es eine angemessene Breite zur *Verfügung*. In der Vertikalen kann das enthaltene DiKo-Objekt dann beliebig viel Platz einnehmen (siehe auch E4).

E4 – Beliebiges Wachsen in der Vertikalen
Ein DiKo-Objekt kann in der Vertikalen beliebig wachsen. In Abhängigkeit der verfügbaren

[9] Siehe Abschnitt *semantisches Skalieren* 1.2
[10] Siehe auch Eigenschaft E3

Breite (siehe E2) benötigt das DiKo-Objekt eine bestimmte Höhe.
Wenn das DiKo-Objekt weitere DiKo-Objekte enthält und diese *hochskaliert* werden, kann dies zu einer Vergrößerung des DiKo-Objektes führen. Insbesondere dafür ist es wichtig, dass das DiKo-Objekt beliebig viel Platz in der Vertikalen einnehmen kann.

2.2 Funktionsweise

DiKo wird in einem Fenster auf der Benutzeroberfläche angezeigt. Das Objekt bzw. die Informationsstruktur wird mit einem DiKo-Objekt dargestellt. Es erhält den gesamten Platz in der Horizontalen zur Verfügung (siehe auch Eigenschaft E2). In der Vertikalen benötigt das DiKo-Objekt eine ungewisse Menge an Platz (siehe auch Eigenschaft E4). Deshalb wird das Fenster nach Bedarf virtuell erweitert, siehe hierfür auch Abschnitt *Scrollen* 5.1.
Die enthaltenen Objekte werden rekursiv dargestellt. Eigenschaft E3 stellt sicher, dass die enthaltenen Objekte genügend Platz haben und Eigenschaft E2 stellt sicher, dass die verfügbare Breite nicht überschritten wird. Abschnitt 2.4.1 zeigt eine Möglichkeit, Platz für enthaltene DiKo-Objekte zu schaffen.

2.3 Darstellbarkeit von Informationsstrukturen mit DiKo-Objekten

Es wird gezeigt, dass folgende Objekte bzw. Informationsstrukturen mit DiKo-Objekten darstellbar sind:

- Gliederungen
- Hypertexte
- Dokumente (mit Querverweisen auf weitere Dokumente)
- semantische Daten
- längere Fließtexte

Zuerst wird erklärt, wie gewisse Grundbausteine mit DiKo-Objekten dargestellt werden können. Danach wird gezeigt, dass obige Objekte bzw. Informationsstrukturen *Zusammensetzungen* dieser Grundbausteine sind.

2.3.1 Grundbausteine

Im Folgenden werden die Eigenschaften E1 bis E4 für die jeweiligen Grundbausteine erklärt.

2.3.1.1 *Wort*

Ein *Wort* wird hier als *kleinstes* DiKo-Objekt betrachtet. Es dient als kleinste Einheit.

E1 – Feingranulares semantisches Skalieren
In der ersten Skalierungsstufe wird das Wort dargestellt. Bei weiterem Hochskalieren wird ebenfalls nur das Wort dargestellt.

E2 – Anpassung an verfügbare Breite
Die verfügbare Breite wird nicht überschritten, da das Objekt immer gleich, als einfaches Wort, dargestellt wird.

E3 – Enthaltene DiKo-Objekte
Ein Wort enthält keine weiteren Objekte. Wie eingangs erwähnt handelt es sich bei diesem Baustein um die kleinste Einheit.[11]

E4 – Beliebiges Wachsen in der Vertikalen
Das Objekt wächst nicht in der Vertikalen, da es immer gleich dargestellt wird.

2.3.1.2 *Liste-mit-Name*

Als *Liste-mit-Name* wird hier eine Liste von DiKo-Objekten mit einem *Namen* (die Überschrift der Liste) bezeichnet. Die DiKo-Objekte der Liste sind *enthaltene DiKo-Objekte* gemäß E3.

E1 – Feingranulares semantisches Skalieren
In der ersten Skalierungsstufe wird nur der *Name* dargestellt. In der zweiten Skalierungsstufe wird zusätzlich die Liste der enthaltenen DiKo-Objekte dargestellt (jeweils in Skalierungsstufe eins). Weiteres Hochskalieren überträgt sich auf die enthaltenen DiKo-Objekte. Das heißt, alle enthaltenen DiKo-Objekte werden um eine Stufe hochskaliert. Das Herunterskalieren verhält sich analog. Die enthaltenen DiKo-Objekte können auch manuell, unabhängig voneinander skaliert werden. Dies entspricht dem *rekursiven* semantischen Skalieren[12].

E2 – Anpassung an verfügbare Breite
In der ersten Skalierungsstufe wird nur der *Name* dargestellt, welcher einen kleinen Platz einnimmt (die Liste ist „eingeklappt“).
Beim Hochskalieren werden die enthaltenen DiKo-Objekte (jeweils in Skalierungsstufe

[11] Man könnte sich überlegen, dass ein Wort eine Liste von Buchstaben ist und die einzelnen Buchstaben als enthaltene DiKo-Objekte auffassen. Der Verständlichkeit zuliebe wird ein Wort hier jedoch als atomare Einheit aufgefasst.

[12] Siehe Abschnitt *Semantisches Skalieren* 1.2

eins) *unter* dem *Namen* eingefügt (die Liste ist „ausgeklappt“). Die enthaltenen DiKo-Objekte werden im sogenannten Fließ-Layout angeordnet. Das Fließ-Layout wird in Abschnitt 2.4 erklärt. Durch das Fließ-Layout wird der Platz in der Horizontalen sinnvoll genutzt.
Wenn ein enthaltenes DiKo-Objekt hochskaliert wird, rutscht es in eine eigene Zeile und erhält eine Breite zur Verfügung, welche etwas geringer ist, als die verfügbare Breite. Die verfügbare Breite wird somit nicht überschritten. Siehe auch Abschnitt 2.4.1.

E3 – Enthaltene DiKo-Objekte
Abschnitt 2.4.1 erklärt, wie beim Fließ-Layout Platz für Details geschaffen werden kann. Dieses Prinzip wird verwendet, um Platz für DiKo-Objekte, welche hochskaliert werden, zu schaffen. Die hochskalierten enthaltenen DiKo-Objekte haben also (fast) die gleiche Breite zur Verfügung und in der Vertikalen können sie beliebig wachsen.

E4 – Beliebiges Wachsen in der Vertikalen
Die enthaltenen DiKo-Objekte werden im Fließ-Layout angeordnet (solange sie nicht hochskaliert wurden). Durch das Fließ-Layout braucht die Darstellung eine bestimmte Höhe in Abhängigkeit von der verfügbaren Breite. Wenn die enthaltenen DiKo-Objekte hochskaliert werden, wächst auch das DiKo-Objekt selbst in der Vertikalen.

2.3.1.3 *Kurze-Liste*

Als *Kurze-Liste* wird hier eine Liste von DiKo-Objekten bezeichnet, welche in einer relativ kleinen Fläche dargestellt werden kann, wenn alle enthaltenen DiKo-Objekte in Skalierungsstufe eins dargestellt werden.

In Skalierungsstufe eins wird direkt die Liste angezeigt, wobei die enthaltenen DiKo-Objekte jeweils in Skalierungsstufe eins dargestellt werden. Analog zum vorhergehenden Abschnitt können die Eigenschaften (E1 bis E4) erklärt werden.

2.3.1.4 *Längerer-Fließtext*

Als *Längerer-Fließtext* wird hier ein längerer Text mit einer Überschrift bezeichnet. In der ersten Skalierungsstufe wird nur die Überschrift angezeigt. In der zweiten Skalierungsstufe werden zusätzlich die ersten Sätze des Textes angezeigt. Bei Online-Zeitungen werden auf der Übersichtsseite die einzelnen Artikel meist entsprechend Skalierungsstufe zwei dargestellt. Erst wenn man auf einen Artikel klickt wird der komplette Artikel angezeigt. Die komplette Darstellung entspricht hier Skalierungsstufe drei. (Weiteres Hochskalieren hat keinen Effekt mehr.)
Beim Fließtext werden die einzelnen Wörter entsprechend dem Fließ-Layout angeordnet. Dadurch passt sich die Darstellung an die verfügbare Breite an (E2). Der Platzverbrauch

in der Vertikalen ist abhängig von der verfügbaren Breite (E4).

2.3.1.5 *Link*

Als *Link* wird ein Verweis auf ein DiKo-Objekt bezeichnet. In der ersten Skalierungsstufe wird nur das Link-Wort dargestellt. In der zweiten Skalierungsstufe wird unter dem Link-Wort das verlinkte DiKo-Objekt angezeigt. Je nach Informationsstruktur wird das verlinkte DiKo-Objekt direkt in Skalierungsstufe zwei angezeigt, damit der Benutzer sofort Details zu dem verlinkten DiKo-Objekt sieht. Davon abgesehen ist ein *Link* eine *Liste-mit-Name*, wobei das Link-Wort der *Name* ist und das verlinkte DiKo-Objekt eine *Liste* der Länge *eins* ergibt.

2.3.2 Zusammengesetzte Objekte und Informationsstrukturen

In diesem Abschnitt wird gezeigt, wie größere Informationsstrukturen mit den *Grundbausteinen* aus dem vorherigem Abschnitt *zusammengesetzt* werden können.
Gliederungen sind hierarchische Zusammensetzungen von *Liste-mit-Name*-Objekten.
Ein *Text mit Überschrift* ist ein *Liste-mit-Name*-Objekt. Die Überschrift ist der *Name*. Der Text ist eine Liste von *Wörtern*.
Ein *Hypertext mit Überschrift* ist ein *Text mit Überschrift*, außer dass die *Wörter* auch *Links* sein können.
Dokumente sind zusammengesetzt aus einer *Gliederung*, *(Hyper-)Texten mit Überschrift* (Überschrift = Abschnittsüberschrift) und *längeren-Fließtexten*.
Semantische Daten werden im folgenden Unterabschnitt behandelt.

2.3.2.1 Semantische Daten

Semantische Daten werden hier als Menge von *Elementen*, welche über *subject-predicate-object*-Beziehungen netzartig verbunden sind, aufgefasst. *subject*, *predicate* und *object* sind *Elemente*. Jedes *Element* habe einen *Bezeichner*.
Es wird gezeigt, dass ein *Element* als DiKo-Objekt dargestellt werden kann (Die restlichen *Elemente* sind dann über dieses *Element* zu erreichen):
Dazu wird gezeigt, dass das *Element* als *Liste-mit-Name* aufgefasst werden kann:
Der *Bezeichner* des *Elements* ist der *Name*.
Die zugehörigen *subject-predicate-object*-Beziehungen werden als *Liste* von *(predicate-object)-Paaren* aufgefasst.
(predicate-object)-Paare sind *Kurze-Liste*-Objekte (der Länge zwei).
predicate und *object* sind wiederum *Elemente*.

2.4 Fließ-Layout

Beim *Fließ-Layout* werden die Objekte ähnlich den Wörtern in einem Fließtext angeordnet. Das bedeutet, ein Objekt wird in der gleichen Zeile hinter dem vorhergehendem Objekt eingefügt, sofern dort noch genügend Platz ist. Andernfalls kommt es an den Anfang einer neuen Zeile. Dieses Layout-Prinzip wird hier *Fließ-Layout* genannt. Abbildung 1 zeigt die Verwendung des Fließ-Layouts.[13]
Die Fläche, welche das Fließ-Layout nutzt, *passt* sich an den Platz in der Horizontalen *an*. Abhängig vom verfügbaren Platz in der Horizontalen, *benötigt* sie einen bestimmten Platz in der Vertikalen. Siehe Abbildung 2.
Mit dem Fließ-Layout können offensichtlich *Listen* dargestellt werden.[14]

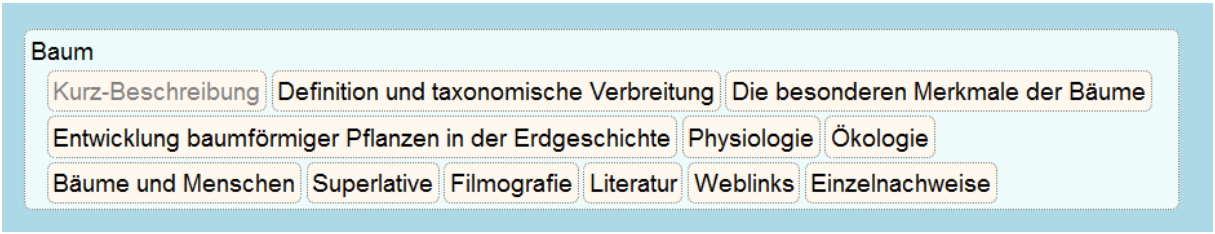

Abbildung 1: Veranschaulichung des Fließ-Layouts.[15]

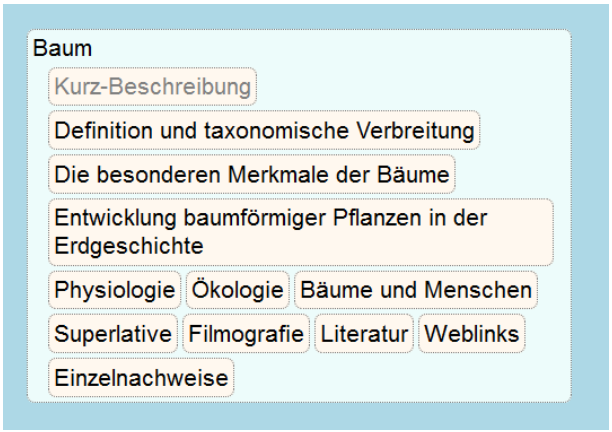

Abbildung 2: Anpassung an Breite

[13] In Java wird dieses Layout übrigens *FlowLayout* genannt.
`https://de.wikipedia.org/wiki/Layoutmanager`

[14] Das Fließ-Layout wird häufig in unterschiedlichen Situationen verwendet. Beispielsweise können im Windows-Explorer Dateien (oder auch Miniaturansichten von Bildern) im Fließ-Layout angeordnet werden. HTML-Objekte werden standardmäßig im Fließ-Layout angeordnet.

[15] Quellenangabe: In dieser Arbeit wurde der Artikel *Baum* aus der freien Enzyklopädie Wikipedia an mehreren Stellen verwendet. Zum Einen dienen das Inhaltsverzeichnis und einige Abschnitte als Beispiel-Text. Zum Anderen werden Screenshots des Artikels verwendet, wobei der eigentliche Inhalt für diese Arbeit nicht relevant ist. https://de.wikipedia.org/wiki/Baum (Zuletzt abgerufen 10.03.17)

2.4.1 Einfügen von Details beim Fließ-Layout

Es wird gezeigt, wie beim Fließ-Layout Platz für Details geschaffen werden kann, so dass Folgendes erfüllt ist: Die Fläche für die Details ist fast so breit, wie die Gesamtbreite und in die Vertikale können die Details beliebig viel Platz einnehmen.
Abbildung 3 zeigt wie bei Abbildung 1 *Details* zu *Die besonderen Merkmale der Bäume* eingefügt werden können.
Hinter dem Objekt, zu dem Details angezeigt werden sollen, findet ein Zeilenumbruch statt und die nachfolgenden Objekte werden nach unten verschoben (um Platz für die Details zu machen). Die Details werden direkt unter dem Objekt eingefügt. Das Objekt selbst rutscht ebenfalls in eine eigene Zeile[16].

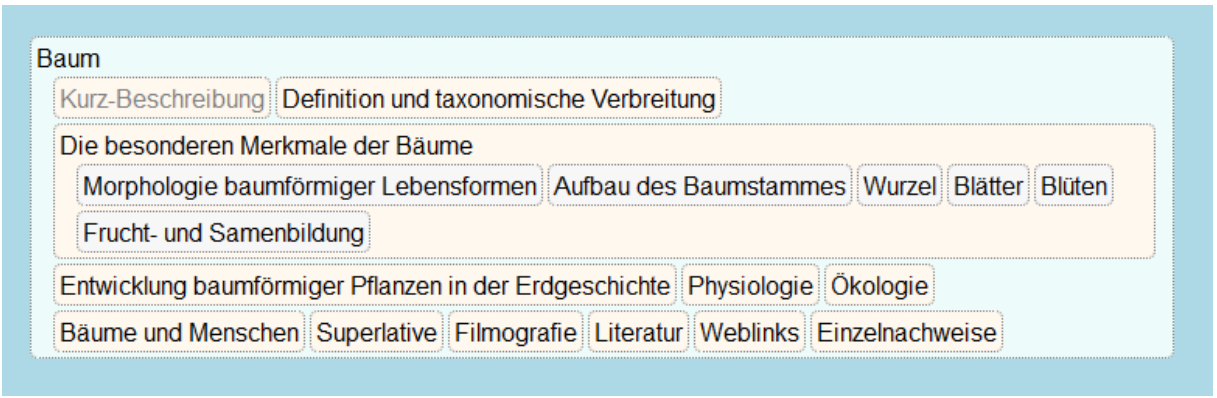

Abbildung 3: *Details* zu *Die besonderen Merkmale der Bäume* wurden eingefügt

[16] In Abschnitt 3.1 wird die Frage thematisiert, ob das Objekt auch an seinem Platz bleiben kann, also nicht verrutscht.

3 Implementierung

Das DiKo-Konzept wurde implementiert. Die entsprechenden Abbildungen sind Screenshots dieser Implementierung. Sie diente der Untersuchung, ob das DiKo-Konzept auch in der Praxis funktioniert.
Die Implementierung kann per Maus und per Tastatur bedient werden. Das Hoch- bzw. Herunterskalieren wird mit einer einfachen Tastenkombination oder durch Klicks gesteuert. Abschnitt 3.1 erklärt das visuelle Design und Abschnitt 3.2 geht auf die Frage ein, inwiefern das DiKo-Konzept umgesetzt werden konnte.

Im Folgenden werden die technischen Details der Implementierung kurz vorgestellt:
Es wurde HTML/JavaScript/TypeScript verwendet. Im Rahmen dieser Arbeit wurde das Programm als lokale Anwendung umgesetzt, das heißt, es kann mit einem Browser lokal geöffnet und gestartet werden. Die Verwendung von Webtechnolgien war dadurch motiviert, dass das Programm dann leicht zu einer Webanwendung erweitert werden kann. Im Abschnitt 6.3 wird kurz vorgestellt, welche Möglichkeiten DiKo als Webanwendung bieten könnte. Das Programm funktioniert mit allen gängigen Browsern.
Die Verwendung von TypeScript war insbesondere dadurch motiviert, dass das Programm später auch softwaretechnisch korrekt erweitert werden kann. (TypeScript versieht JavaScript mit einem Typ-System).
Durch die Verwendung von *span*-Elementen, deren CSS-Eigenschaft *contenteditable* auf *true* gesetzt wurde, gleicht die Benutzung von DiKo einem Texteditor. Man kann sich mit dem Cursor über die Objekte bewegen, als ob es sich um ein zusammenhängendes Textfeld handeln würde. Außerdem können die Objekte direkt manipuliert und die Datenbasis exportiert bzw. importiert werden. Dies wurde insbesondere für die Erstellung und Speicherung von Test-Szenarien verwendet. Mit der Implementierung können auch Daten von *iMaps* dargestellt werden. Als *iMaps* werden Wissenslandschaften bezeichnet, welche mit dem *iMapping*-Verfahren[17] erstellt wurden (siehe Abbildung 7 auf Seite 22).

3.1 Visuelles Design

Im Rahmen dieser Arbeit wurde nicht untersucht, inwiefern das visuelle Design optimiert werden kann. Es diente hier lediglich der Veranschaulichung und Umsetzung des DiKo-Konzepts.
Um die rekursiven Detail-Beziehungen darzustellen, wurde die Analogie von verschachtelten Kärtchen verwendet[18]. Details werden als Kärtchen dargestellt, welche unter dem entsprechenden Objekt positioniert werden. Es soll der Eindruck entstehen, dass die Detail-

[17] Das iMapping-Verfahren wird in [Hal11] erklärt. Siehe auch `http://www.imapping.info/`
[18] Verschachtelte Kärtchen werden auch beim *iMapping*-Verfahren verwendet [Hal11]

Kärtchen auf dem Kärtchen des entsprechenden Objektes liegen[19].
Es wurden fünf verschiedene matte Hintergrundfarben für die Kärtchen verwendet. Für jede Ebene gibt es eine eigene Farbe bzw. nach fünf Ebenen wird wieder die erste Farbe verwendet. Dies soll helfen, dass der Benutzer die Kärtchen korrekt der richtigen Ebene zuordnen kann und beim Anzeigen bzw. Ausblenden von Objekten die Übersicht behält. Insbesondere sollen die Hintergrundfarben dem Benutzer helfen zu sehen, an welcher Stelle neue Objekte eingefügt werden.
In Abschnitt 2.4.1 wurde beschrieben, wie das Einfügen von Details beim Fließ-Layout funktioniert. Es stellt sich die Frage, ob das Objekt, zu dem der Benutzer Details sehen möchte, in eine eigene Zeile rutschen soll oder ob es besser ist, wenn es an seiner ursprünglichen Position bleibt. Wenn das Objekt in eine eigene Zeile rutscht, ist die Darstellung insgesamt konsistenter: Das übergeordnete Objekt zu Details befindet sich immer oben links. Der Nachteil ist jedoch, dass ausgerechnet das Objekt, für das der Benutzer sich interessiert, verschoben wird. Außerdem wird dann mehr Platz in der Vertikalen verbraucht (eine Zeile). Für *Links in Texten*[20] wurde die zweite Variante verwendet, siehe Abbildung 4. Es bleibt zu untersuchen, inwiefern durch das Verrutschen ein wertvoller visueller Ankerpunkt[21] verloren geht bzw. welche Variante vorzuziehen ist.
Angelehnt an die typischen Einrückungen in Quellcodes (siehe zum Beispiel auch Abbildung 5) bzw. die Einrückungen bei der *Baumansicht* (siehe Abbildung 6), wurden die seitlichen Abstände etwas größer gewählt (Siehe zum Beispiel Abbildung 4)
Die Analogie zu verschachtelten Kärtchen und die *Einrückungen* sollen die *Detail*-Beziehungen intuitiv verständlich machen.

3.2 Umsetzung des DiKo-Konzepts

Das Ziel von DiKo ist es, *kontexterhaltendes rekursives semantisches Skalieren* von Objekten zu ermöglichen[22].
Abbildung 4 zeigt einen Screenshot der Implementierung. Sie zeigt, wie (verlinkte) Dokumente (und somit auch Gliederung, Links und Hypertexte) dargestellt werden.
Eine Beobachtung ist die *beschränkte* Rekursionstiefe. Die Beschränkung wird dadurch verursacht, dass die enthaltenen Objekte etwas weniger Platz in der Horizontalen haben[23].
Die tapetenartige Erscheinung am unteren mittleren linken bzw. rechten Rand wirkt nicht elegant. Dort befindet sich ungenutzter Platz. Wie bereits erwähnt, wurde das visuelle De-

[19] Damit genügend Platz für die Detail-Kärtchen ist, müssen die übergeordneten Kärtchen vergrößert werden. Das Vergrößern von Kärtchen entspricht eigentlich nicht der Realität.

[20] Siehe auch nachfolgender Abschnitt 3.2

[21] Siehe Abschnitt 1.1.2

[22] Siehe auch entsprechende Abschnitte in der Einleitung.

[23] Es stellt sich die Frage, welche Rekursionstiefe gewöhnlich auftritt. Wenn jedes Objekt zehn Kind-Objekte hat, kann man mit zwölfmaligem rekursiven Hochskalieren *eine Billion* (= 1 000 000 000 000) verschiedene Objekte erreichen. Dies nur als Zahlenbeispiel, der Sachverhalt wurde nicht tiefer untersucht.

sign nicht tiefergehend untersucht (siehe Abschnitt 3.1). Eventuell würde es Sinn machen, die seitlichen Abstände zu verringern (Dadurch könnte auch die mögliche Rekursionstiefe erhöht werden).
Die Abbildung zeigt die Umsetzung von *Links in Texten*. *Nussartig* ist ein Link-Wort. Der verlinkte Inhalt (*Nussfrucht*) wird als DiKo-Objekt direkt darunter angezeigt. Das Prinzip aus Abschnitt 2.4.1 wurde also auf *Links in Texten* übertragen. Dies ist möglich, weil ein Fließtext als Liste von Wörtern – angeordnet im Fließ-Layout – aufgefasst werden kann. Wie bereits erwähnt, wurde bei *Links in Texten* das Prinzip dahingehend abgewandelt, dass das skalierte Objekt *nicht* in eine eigene Zeile rutscht, sondern seine Position beibehält. Siehe auch den vorhergehenden Abschnitt 3.1.
Die Darstellung mit DiKo kann prinzipiell in einer beliebigen Breite erfolgen. Wird DiKo jedoch auf einer relativ breiten Fläche dargestellt, ist die Platzausnutzung insgesamt schlechter (der ungenutzte Platz hinter *Blüten* würde beispielsweise noch größer). Fließtexte sind außerdem nur bis zu einer bestimmten Breite angenehm zu lesen. Das Thema Platzausnutzung wird auch in den Abschnitten *Fensterartige Systeme* 4.2 und *Verwendung des Fließ-Layouts* (Seite 21) beleuchtet.
Für Probleme, welche im Zusammenhang mit der *virtuellen Erweiterung* stehen, wird auf Abschnitt *Scrollen* 5.1 verwiesen.
In Abschnitt 4 wird untersucht, inwiefern DiKo das eingangs erwähnte Ziel im Vergleich zu anderen Konzepten und Werkzeugen erfüllt. Abschnitt 5 geht auf einzelne wichtige Aspekte bei der Implementierung ein.

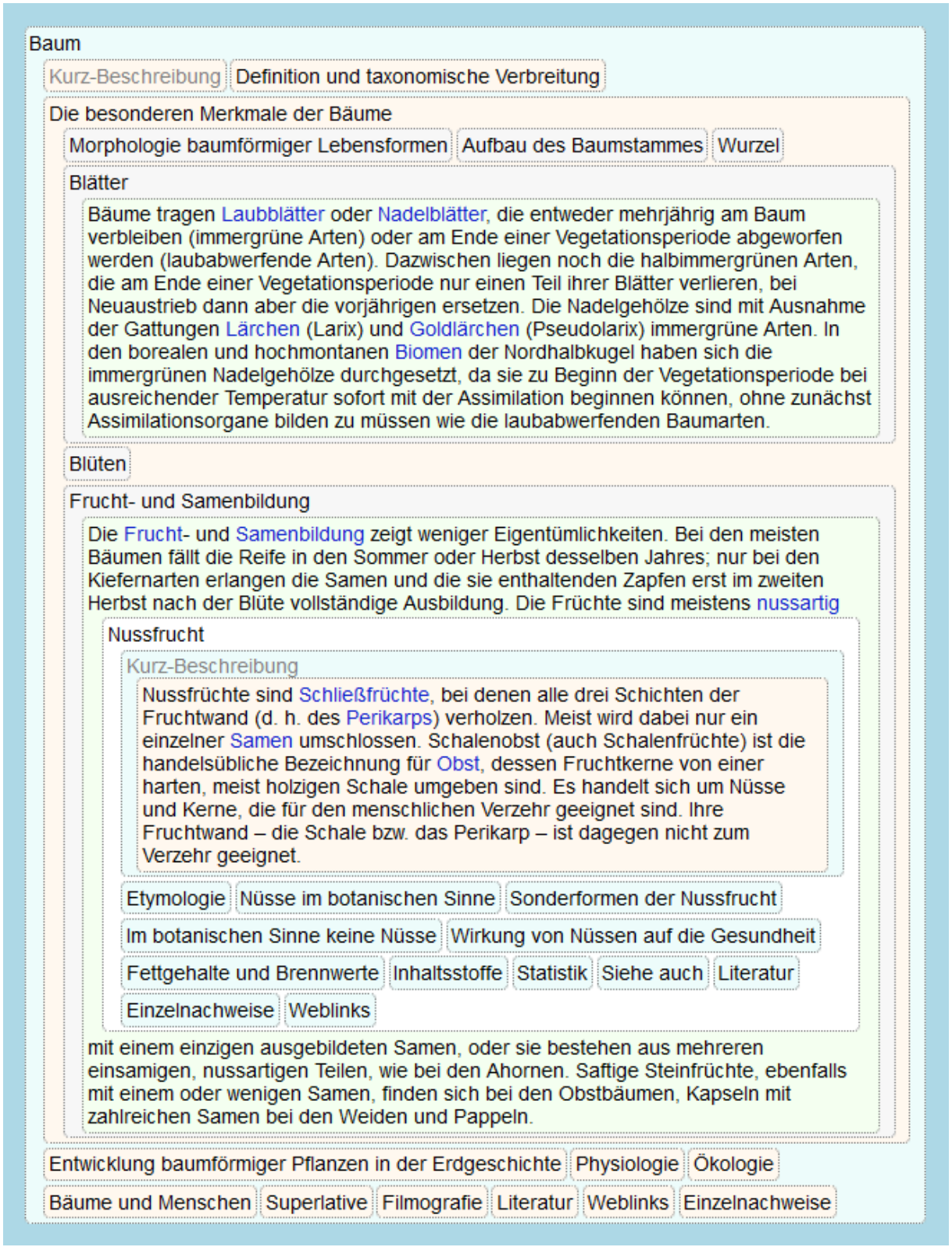

Abbildung 4: Darstellung von Wikipedia-Artikeln mit DiKo

4 Vergleiche

Grundlegende Möglichkeiten, um Details so anzuzeigen, dass der Kontext erhalten bleibt, werden im Folgenden untersucht und mit DiKo verglichen.
Die Details können als kleine virtuelle Kärtchen *auf* die Benutzeroberfläche gelegt werden. Diese Möglichkeit wird in Abschnitt *Pop-Ups* 4.1 untersucht.
Sie können in einem weiteren Fenster *daneben* angezeigt werden. Dies wird in Abschnitt 4.2 untersucht.
Außerdem können die Details wie bei DiKo in den Kontext *integriert* werden. Die *Baumansicht* stellt eine entsprechende, weit verbreitete Technik dar und wird in Abschnitt 4.3 untersucht. Weitere Techniken, um Details in den Kontext zu *integrieren* finden sich in Abschnitt *Focus+Context* 6.1.1.
Systeme, welche verschiedene dieser Techniken kombinieren, sind (je nach Betrachtungsweise) inkonsistent. Konsistenz ist jedoch gerade bei feingranularen Informationsstrukturen wichtig. Siehe auch *consistency and standards* in Abschnitt 6.2. Außerdem würde eine Kombination der Techniken wieder darauf hinauslaufen, dass mehre Fenster, oder zumindest mehrere abgetrennte Bereiche, verwendet werden. Fensterartige Systeme werden in Abschnitt 4.2 untersucht.

4.1 Pop-Ups

Pop-Ups sehen aus wie Kärtchen, welche *auf* die Benutzeroberfläche gelegt werden. Sie liefern Details zu einem Objekt in einem kleinen Fenster. Meist werden sie direkt unter dem entsprechenden Objekt positioniert. Abbildung 5 zeigt ein Pop-Up in einer Programmierumgebung[24].[25]
Pop-Ups bieten eine Möglichkeit, flexibel zu jedem Objekt auf der Benutzeroberfläche weitere Details anzuzeigen.
Sie sind ein gutes Beispiel dafür, wie Details ohne einen Kontext-Wechsel angezeigt werden können. Sie verdecken jedoch einen Teil des unmittelbaren Kontext des Objektes für das man sich interessiert.
Im Allgemeinen hat der unmittelbare Kontext einen erhöhten Informationswert. In Abbildung 5 verdeckt das Pop-Up-Fenster Inhalte der Methode, welche der Benutzer gerade bearbeitet, während die Inhalte anderer Methoden zu sehen sind. Der Inhalt der Methode,

[24]Die Abbildung zeigt eine sogenannte *Intelligente Codevervollständigung*. Es handelt sich dabei um ein Pop-Up, welches beim Schreiben von Quellcode (direkt unter der Schreibposition) angezeigt werden kann.

[25]Der Begriff *Pop-Up* scheint keine allgemein feste Definition zu haben. In Wikipedia wird er in erster Linie als Technik um (störende) Werbung anzuzeigen erklärt. In [Shn96] erscheint er jedoch als *details-on-demand* Technik, wird also im gleichen Sinne, wie in dieser Arbeit, verwendet. Für den Zusammenhang von DiKo zu *details-on-demand* bzw. zum *Visual Information Seeking Mantra* siehe Abschnitt 6.1.2.

an der man gerade arbeitet hat jedoch im Allgemeinen einen höheren Informationswert, als der Inhalt anderer Methoden. Bei DiKo werden beim Anzeigen von Details keine Objekte verdeckt. Dies stellt einen Vorteil gegenüber Pop-Ups dar.
Meist sind Pop-Ups so implementiert, dass sie verschwinden, sobald man woanders hinklickt. Dies macht es aufwendig, die Details mehrerer Objekte zu vergleichen. Außerdem ist ein rekursives Anzeigen von Details meist nicht (oder nur sehr beschränkt) möglich. Dies stellt ebenfalls einen Nachteil gegenüber DiKo dar, bei dem man unabhängig jedes Objekt seinem aktuellen Informationsbedarf entsprechend skalieren kann. Auch das *mehrfache Skalieren* (siehe Abschnitt 5.2) ist bei Pop-Ups nicht vorgesehen.

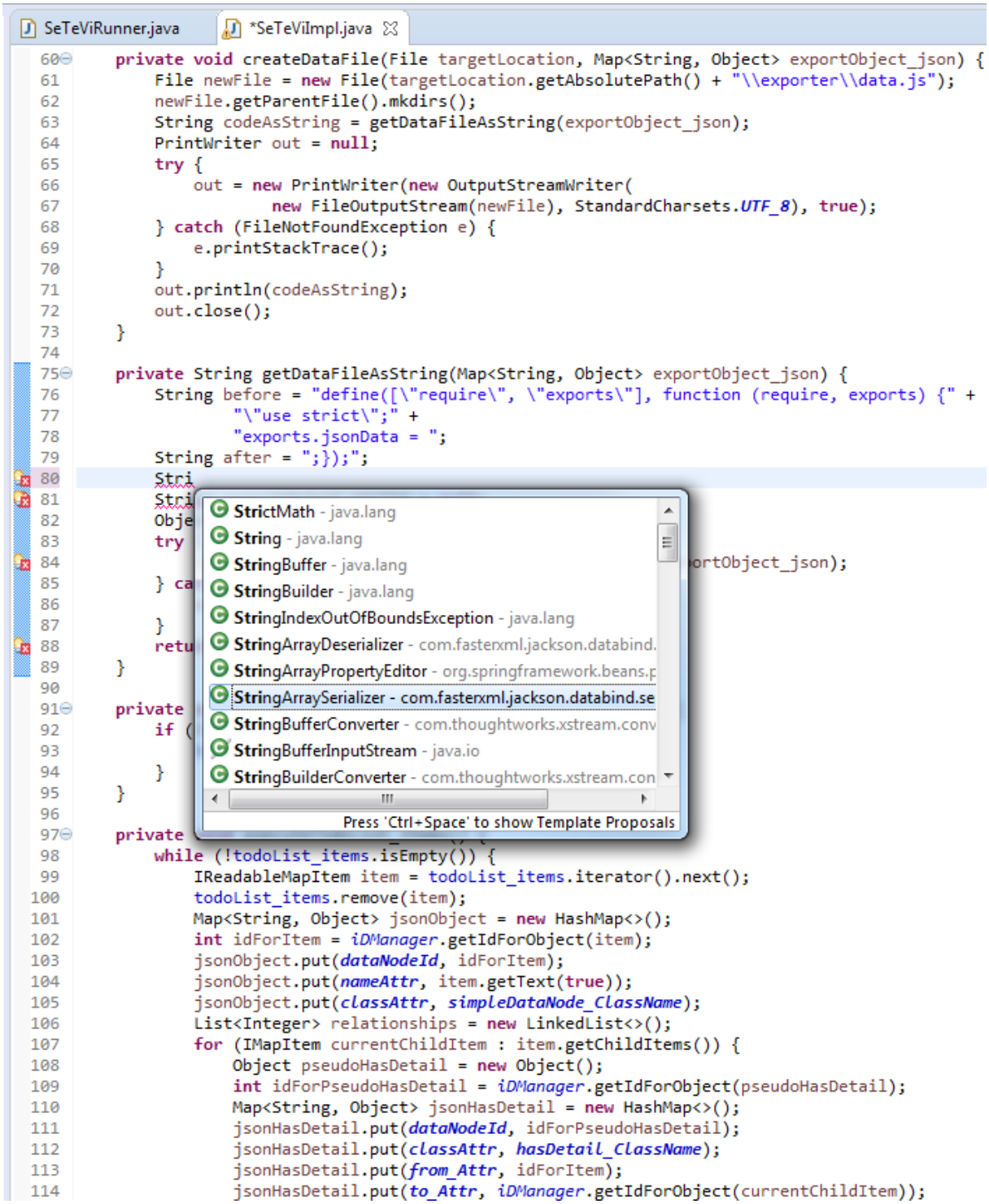

Abbildung 5: Intelligente Codevervollständigung bei *Eclipse*

4.2 Fensterartige Systeme

Als *fensterartige Systeme* werden hier grafische Benutzeroberflächen-Systeme bezeichnet, bei denen *Details* in einem extra Fenster, welches manuell positioniert werden kann, angezeigt werden. Mit Web-Browsern ist es beispielsweise möglich, *Links* in einem neuen

Fenster zu öffnen.
Diese Technik ermöglicht ein flexibles Anzeigen weiterer Details zu beliebigen Objekten auf der Benutzeroberfläche. Das *rekursive* Anzeigen von Details (siehe Abschnitt 1.1.1) ist prinzipiell unbeschränkt möglich.
Je nach Implementierung und Rahmenbedingung (Bildschirmgröße etc.) kann das neue Fenster neben dem aktuellen Fenster angezeigt werden. In diesem Fall ist das Anzeigen ohne Kontext-Wechsel möglich[26]. Es ist jedoch ein mentaler Aufwand notwendig, um die Beziehung zwischen Detail-Fenster und entsprechendem Objekt herzustellen.[27]
Der Benutzer muss also die verschiedenen Fenster auf einer logischen Ebene manuell zusammensetzen.
Dies bedeutet bei *feingranularem* semantischen Skalieren einen besonderen mentalen Aufwand. Wenn die Details sowieso eine große Fläche einnehmen, ist das manuelle Zusammensetzen verkraftbar. Wenn hingegen jede Ebene einer Gliederung in einem eigenen Fenster angezeigt würde, entstünde ein hoher mentaler Aufwand, um die Darstellung zu verstehen.
Eine DiKo-Darstellung bei sehr breiter Fläche macht nur beschränkt Sinn. Das Leseverständnis leidet zum Beispiel, wenn Fließtexte in sehr breiter Fläche dargestellt werden. Wenn sehr viel Platz in der Horizontalen zur Verfügung ist, kann man diesen mit einer DiKo-Ansicht schlecht nutzen. Mit einem fensterartigen System kann man den Platz jedoch sehr flexibel nutzen. Es bleibt zu untersuchen, inwiefern man diesen Vorteil eines fensterartigen Systems auf DiKo übertragen kann. Der Benutzer könnte dann flexibel per Drag-and-Drop[28] ein Objekt aus der ersten DiKo-Ansicht herausziehen und in einer weiteren DiKo-Ansicht daneben anzeigen. Es bleibt zu untersuchen, ob dies weitere Vorteile mit sich bringt, zum Beispiel, dass man zwei größere Bereiche miteinander vergleichen kann.[29]

4.3 Baumansicht

Die *Baumansicht* ist ein grafisches Benutzeroberflächen-Konzept zum Anzeigen von hierarchischen Strukturen.
Die Elemente der Baumansicht können aus- bzw. eingeklappt werden. Sie werden vertikal

[26] *Kontexterhaltung* wird in Abschnitt 1.1.2 behandelt.

[27] Folgendes Zitat aus [CKB09] sei hier ergänzend wiedergegeben, welches ebenfalls das Problem des kognitiven Zusammensetzens thematisiert: „Spatial separation demands that users assimilate the relationship between the concurrent displays of focus and context. Evidence that this assimilation process hinders interaction is apparent in the experiments of [[HBP02]], who note that 'switching between the detail and the overview window required mental effort and time'. [[BGBS02]] made similar observations of costs associated with split attention with overview+detail interfaces."

[28] Siehe zum Beispiel `https://de.wikipedia.org/wiki/Drag_and_Drop`

[29] Die Nutzung des Platzes in der Horizontalen scheint besonders wichtig, da PC-Bildschirme meist deutlich breiter als hoch sind.

angeordnet. Wenn der Platz in der Vertikalen nicht mehr ausreicht, wird dieser virtuell erweitert. Die *virtuelle Erweiterung* wird in Abschnitt *Scrollen* 5.1 erklärt.[30]
Abbildung 6 zeigt eine Baumansicht auf ein Dateisystem.
Die Baumansicht ermöglicht insbesondere das *rekursive* Anzeigen von Details, wie es in Abschnitt *Bedarf an Details* thematisiert wurde. Als *Details* eines Objektes werden hier dessen Kind-Objekte aufgefasst. Das heißt, der Benutzer kann sich (je nach Datenmenge und Platz auf der Benutzeroberfläche) beliebig tief in die hierarchische Struktur hineinklicken.
Meist wird die Baumansicht für Gliederungen verwendet, es ist jedoch auch möglich *Texte* in die Baumansicht zu integrieren. Die sogenannte Code-Faltung[31] kann als Baumansicht betrachtet werden und stellt somit ein entsprechendes Beispiel dar. Ein weiteres Beispiel ist der Gliederungseditor *Workflowy*. Dieser werde auch dazu benutzt, ganze Bücher darzustellen[32].
Beim Anzeigen von Details bleibt der Kontext (dem Prinzip nach) erhalten. Im Gegensatz zu Pop-Ups[33] werden keine umliegenden Objekte verdeckt. Statt dessen werden die Objekte, welche sich unter dem auszuklappenden Objekt befinden, nach unten verschoben, um Platz für die Details (die Kind-Objekte) zu machen.
Im Folgenden werden Probleme der Baumansicht aufgezeigt und es wird ein Vergleich mit DiKo angestellt.

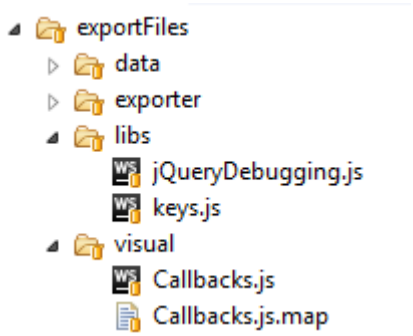

Abbildung 6: Baumansicht auf ein Dateisystem

DiKo liefert eine kompaktere Darstellung als die Baumansicht (siehe unten). Dadurch muss man bei DiKo *weniger scrollen* und die Probleme, welche in Abschnitt *Virtuelle Erweiterung in der Vertikalen* 5.1 beschrieben werden treten *seltener* auf.
Das in Abschnitt 5.2 beschriebene *mehrfache Skalieren* ist bei Baumansichten meist nicht implementiert. (Beim mehrfachen Skalieren eines Objektes wird die Skalieroperation auf alle Kind-Objekte übertragen.)

[30] Siehe auch `https://de.wikipedia.org/wiki/Baum_(Steuerelement)`
[31] Siehe auch `https://de.wikipedia.org/wiki/Code-Faltung`
[32] Siehe auch `https://blog.workflowy.com/2016/07/07/books-in-workflowy` Abgerufen am 10.03.16
[33] Siehe Abschnitt 4.1

Der wesentliche konzeptionelle Unterschied zwischen der Baumansicht und DiKo ist die Verwendung des *Fließ-Layouts*. Dieses wird im Folgenden untersucht.
In Kapitel 5 werden Lösungen für konkrete Probleme vorgestellt, welche nicht nur für DiKo, sondern auch für die Baumansicht relevant sind.

Verwendung des Fließ-Layouts

Bei der Baumansicht werden die Objekte nur vertikal angeordnet. Bei DiKo werden die Objekte dank Fließ-Layout auch horizontal angeordnet. Dadurch entsteht eine kompaktere Darstellung und der Platz in der Horizontalen wird besser genutzt. Abbildung 8 zeigt den Inhalt von Abbildung 7 *ohne* Verwendung des Fließ-Layouts. Die Nutzung des Platzes in der Horizontalen ist besonders wichtig, da Bildschirme häufig deutlich breiter als hoch sind. Wenn die Darstellung breite Objekte enthält und man deshalb die Fensterbreite entsprechend groß wählt, ist eine gute Platzausnutzung in der Horizontalen ebenfalls wichtig. Abbildung 4 und 9 zeigen jeweils die gleichen Informationen – einmal mit und einmal ohne Verwendung des Fließ-Layouts.
Das Konzept *Einfügen von Details beim Fließ-Layout* kann auch auf Fließtexte übertragen werden (Siehe Abschnitt 3.2). Dadurch wird es möglich, verlinkte Inhalte in die Ansicht zu integrieren und es ist kein Kontext-Wechsel notwendig.
In Umfragen für qualitatives Feedback wurde das *in place* Anzeigen von *verlinkten Inhalten* als besonders nützlich empfunden. Man müsse sich dadurch nicht neu orientieren. Die bessere Platzausnutzung von DiKo wurde ebenfalls bestätigt.

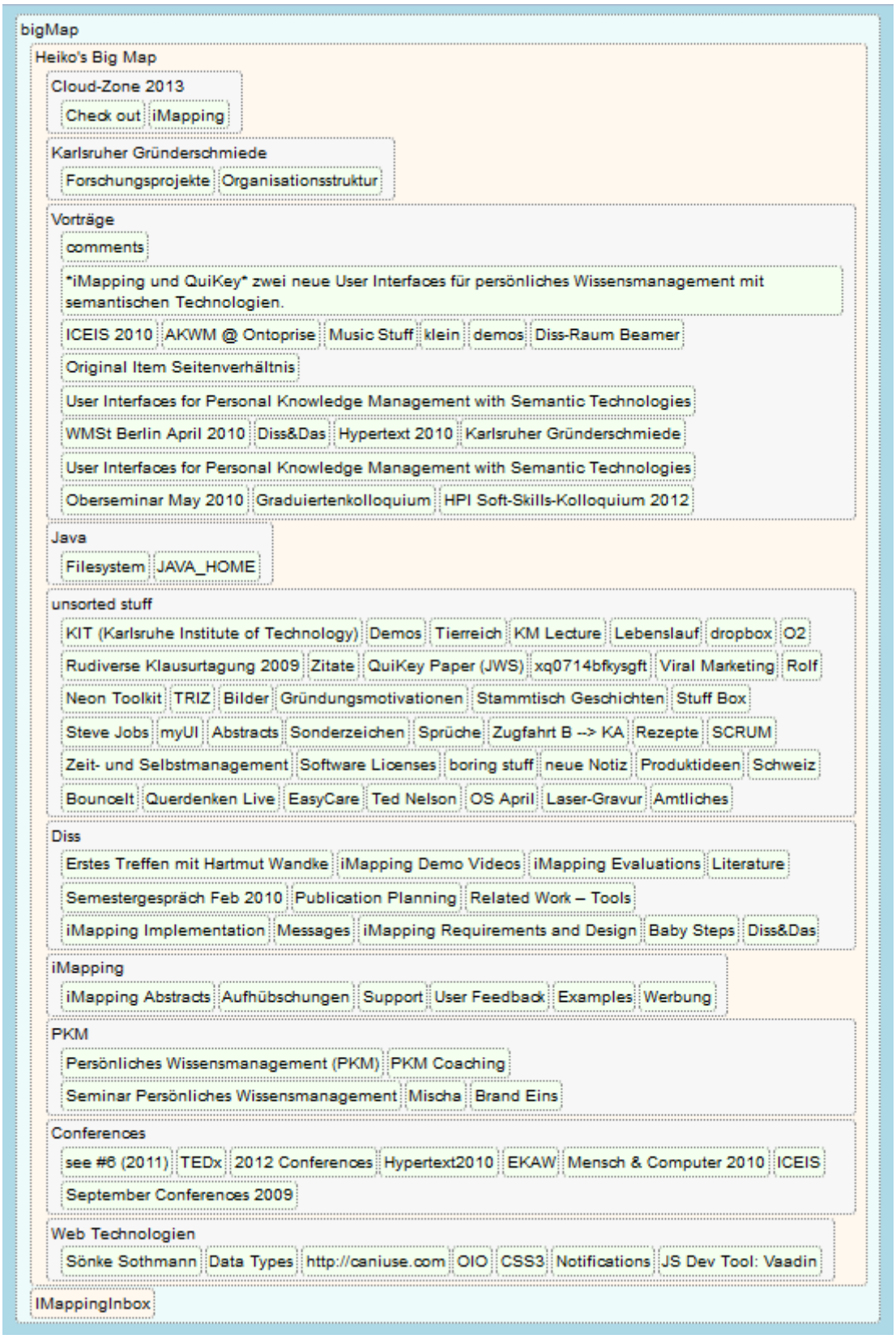

Abbildung 7: Kompakte Darstellung und Nutzung des Platzes in der Horizontalen[34]

[34]Die zugrundeliegenden Daten entstammen einer *iMap* von Dr. Heiko Haller. Als *iMaps* werden Wissenslandschaften bezeichnet, welche mit dem *iMapping*-Verfahren erstellt wurden. Siehe `http://www.imapping.info/`

Abbildung 8: Schlechte Nutzung des Platzes in der Horizontalen ohne Fließ-Layout (vgl. Abbildung 7).

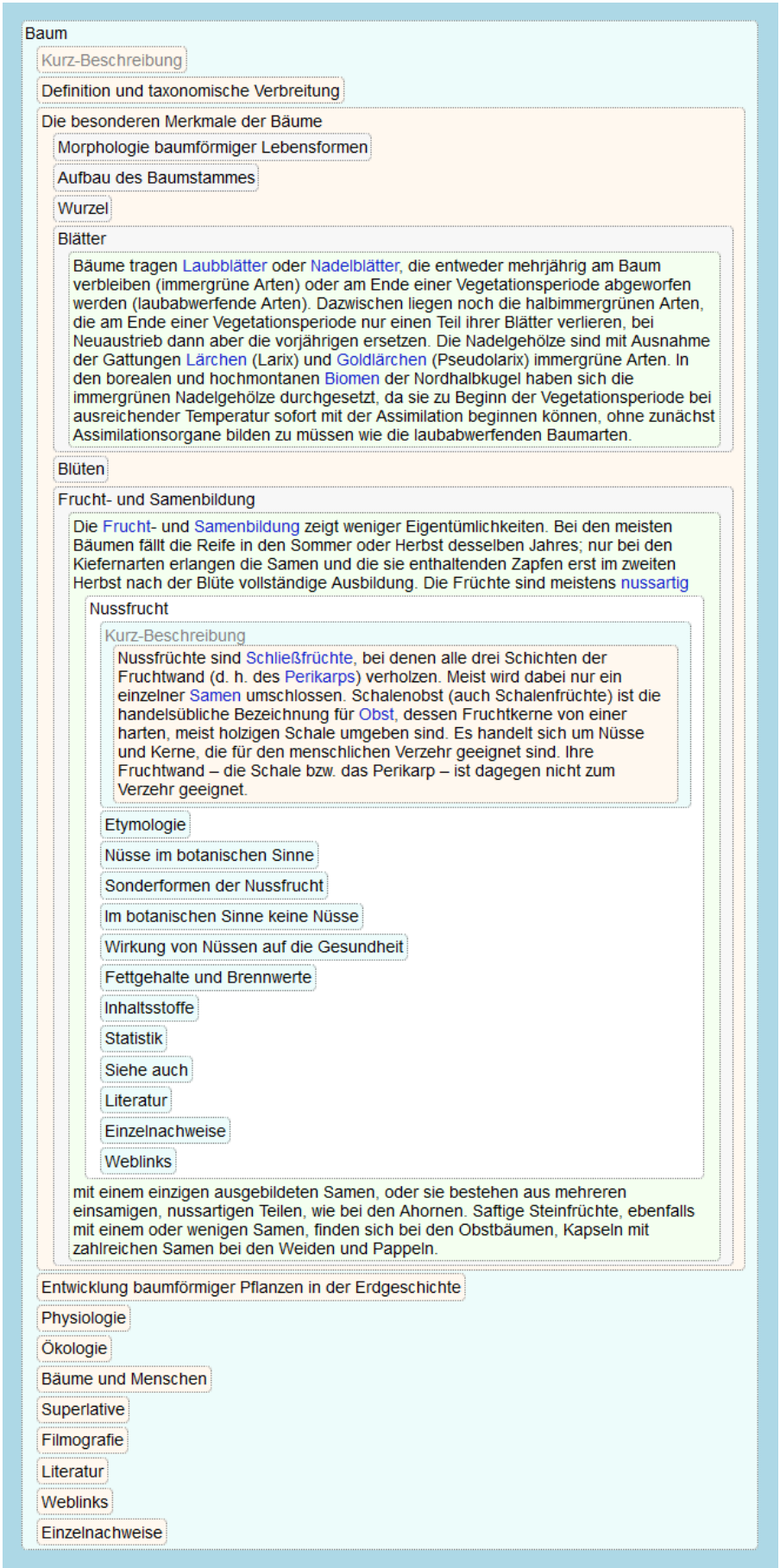

Abbildung 9: Schlechte Platzausnutzung (ohne Fließ-Layout), wenn die Ansicht breite Objekte (hier Fließtext) enthält.

5 Problemlösungen

Während der Entwicklung von DiKo wurden einzelne, wichtige Probleme gefunden. In diesem Abschnitt werden diese vorgestellt und es werden konkrete Lösungen gezeigt. Aufgrund der großen Ähnlichkeit mit DiKo, können diese Lösungen auch auf Baumansichten[35] angewandt werden.

5.1 Scrollen

Die *virtuelle Erweiterung* (Scrolling-Funktion) in der Vertikalen bedeutet, dass je nach Bedarf der Platz in der Vertikalen *virtuell* erweitert wird. Es erscheint ein Scrollbalken an der rechten Spalte, mit dem man den sogenannten *Viewport*[36], also den sichtbaren Ausschnitt, steuern kann. Meist ist der Viewport auch bequem per Mausrad zu bedienen.[37]
Mit dieser Methode wird es möglich, den verfügbaren vertikalen Platz auf der Benutzeroberfläche beliebig wachsen zu lassen. Es handelt sich um ein sehr weit verbreitetes Verfahren – beispielsweise verwendet fast jede Webseite diese Funktion[38].
Im Folgenden werden Probleme bezüglich der virtuellen Erweiterung gezeigt, welche sowohl bei der *Baumansicht*[39], als auch bei DiKo auftreten können. Es werden jeweils Lösungsmöglichkeiten vorgestellt (außer in Abschnitt 5.1.2, welcher lediglich auf eine potentielle Lösung im Abschnitt *Ausblick* 7.1 verweist).

5.1.1 Kontextverlust durch abruptes Verrutschen

Bei den meisten gängigen Implementierungen (der Baumansicht) tritt häufig das Problem auf, dass sichtbare Objekte unerwartet abrupt verrutschen und der Benutzer die Orientierung verliert. Dieses Problem tritt vor allem beim Herunterskalieren auf, kann jedoch auch beim Hochskalieren auftreten:
Wenn man ganz nach unten scrollt und ein Objekt herunterskaliert, verrutschen die oberen Objekte, anstatt – wie gewohnt – die unteren. Genauso verrutscht das skalierte Objekt.
Wenn beim Herunterskalieren die folgende Bedingung auftritt, verrutschen alle Objekte (die oberen, die unteren und das skalierte Objekt): Es wurde nach unten gescrollt, jedoch nicht bis zum unteren Rand, und die virtuelle Fläche unter dem Viewport ist kleiner, als der Platz in der Vertikalen, welcher durch das Herunterskalieren gewonnen wird.
In Abschnitt *Kontexterhaltende Interaktion* 1.1.2 wurde besprochen, dass es wichtig ist, dass hinreichend viele visuelle Ankerpunkte nicht (abrupt) verrutschen. Der Benutzer ist

[35] Siehe Abschnitt 4.3
[36] Siehe auch `https://de.wikipedia.org/wiki/Viewport`
[37] Siehe auch `https://de.wikipedia.org/wiki/Bildlauf`
[38] Einige (bekannte) Webseiten: `https://de.wikipedia.org/`, `https://dict.leo.org`, `http://www.infovis-wiki.net/`, `https://github.com/`
[39] Siehe Abschnitt 4.3

es gewohnt, dass beim Skalieren die oberen Objekte und das Objekt selber als visuelle Ankerpunkte erhalten bleiben. Es besteht die Gefahr des Kontext-Verlusts.
Normans *Gulf of Evaluation*-Theorie lehrt, dass die Differenz zwischen dem, was der Benutzer erwartet und dem tatsächlichen Ergebnis möglichst gering sein soll. Wenn plötzlich die oberen und das skalierte Objekt, anstatt der unteren Objekte verrutschen, bedeutet dies einen größeren *Gulf of Evaluation* [Nor88].
Es handelt sich um ein regelmäßig auftretendes Phänomen, weshalb eine Lösung wichtig ist.
Eine einfache *Lösung* ist das Einfügen einer *Dummy*-Fläche unter der dargestellten Struktur. Dann tritt die Situation, dass man zu nah an den unteren Rand scrollt, nicht auf. Mit animierten Übergängen beim Skalieren kann das Problem ebenfalls abgefedert werden. Bei dem kommerziellem Gliederungseditor *Workflowy*[40] wurde das Problem durch das Einfügen der *Dummy*-Fläche und animierte Übergänge gelöst.
Bei der DiKo-Implementierung trat dieses Problem anfangs ebenfalls auf, da der Scroll-Mechanismus nicht manuell programmiert, sondern vom Browser automatisch umgesetzt wurde. Deshalb wurde bei der Implementierung ebenfalls die *Dummy*-Fläche am unteren Rand eingefügt.

Bei entsprechender Implementierung kann das Problem des abrupten Verrutschens auch beim Hochskalieren auftreten. Dies ist der Fall, wenn die neuen Objekte nicht komplett im sichtbaren Bereich liegen würden und dadurch automatisch eine Änderung des Viewports ausgelöst wird. In [CS94] wird ebenfalls auf dieses Problem (bei der Baumansicht) aufmerksam gemacht: „If an expansion requires more screen space than is available below the item clicked, do not scroll or move the item clicked.“.

5.1.2 Sichtbarkeit übergeordneter Objekte

Als *übergeordnete Objekte* eines Objektes werden die *Vorfahren* dieses Objektes bezeichnet. (Rekursiv definiert: Das Objekt, über welches das Objekt erreicht wurde und die übergeordneten Objekte des übergeordneten Objektes.)
Die übergeordneten Objekte sind besonders wichtig, um die Einbettung in den größeren Kontext zu sehen. Verschwinden sie aus dem *Viewport*, ist die Forderung aus Abschnitt *Kontexterhaltende Interaktion* 1.1.2 zumindest auf visueller Ebene nicht erfüllt.
Manche oder sogar alle übergeordneten Objekte eines sichtbaren Objektes sind nicht zu sehen, wenn man nach unten scrollt.
Das Wurzel-Objekt verschwindet bereits aus der sichtbaren Fläche, wenn man nur wenig nach unten scrollt.
Aber auch der direkte *Vorfahr* eines Objektes kann relativ schnell aus dem Viewport verschwinden: Dies ist der Fall, wenn ein vorhergehendes Objekt auf gleicher Ebene viel

[40] Siehe `https://workflowy.com/`

Platz einnimmt, weil es *hochskaliert* wurde.
Im Abschnitt *Ausblick* 7.1 wird eine Lösung für dieses Problem vorgestellt.

5.2 Mehrfaches Skalieren

In Abschnitt 2.3.1.2 wurde gezeigt, wie ein Liste-mit-Name-Objekt *mehrfach* hochskaliert werden kann. Es wird gezeigt, warum das *mehrfache Skalieren* besonders wichtig ist.
Wenn man ein Objekt in Skalierungsstufe eins hochskaliert, bedeutet dies, dass eine Liste der enthaltenen Objekte erscheint, wobei die enthaltenen Objekte jeweils in Skalierungsstufe eins dargestellt werden. Bei weiterem Hochskalieren wird dieses auf die enthaltenen Objekte übertragen. Alle enthaltenen Objekte werden dann hochskaliert. Die Möglichkeit ein Objekt mehrfach hoch- (bzw. herunter) zu skalieren wird hier als *mehrfaches Skalieren* bezeichnet.
Der Vorteil des mehrfachen Skalierens ist, dass man alle enthaltenen Objekte mit einer einzigen Operation skalieren kann (anstatt alle einzeln skalieren zu müssen).
Das *mehrfache Skalieren* ist besonders dann wichtig, wenn die Informationsstruktur wenig Kind-Objekte pro Objekt bzw. viele Ebenen hat. Das folgende Beispiel veranschaulicht dies.
Es gehe darum, einen *formalen Satz* mit einer Baumansicht bzw. mit DiKo darzustellen. Dieser Satz *besteht* aus einem Namen, einer Behauptung, einem Beweis und einem Text (hier ein Platzhalter-Text), welcher die Verwendung erklärt. Als Beispiel dient der *Satz von Kuratowski*. Die folgende hierarchische Liste zeigt den Aufbau der Struktur:

- Satz von Kuratowski
  - Behauptung
    - Graph ist planar $\Leftrightarrow$ weder K_5, noch K_{33} sind Minor des Graphen
  - Beweis
    - $\Rightarrow$ (die eine Richtung des Beweises)
    - $\Leftarrow$ (die andere Richtung des Beweises)
  - Verwendung
    - Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Ohne *mehrfaches Skalieren* muss der Benutzer jedes Objekt einzeln *Ausklappen*. In diesem Beispiel sind das vier Skalieroperationen an unterschiedlichen Objekten (zuerst *Satz von Kuratowski* ausklappen, danach *Behauptung*, *Beweis* und *Verwendung*). Mit *mehrfachem*

Skalieren sind es zwei Skalieroperationen an *einem* Objekt (*Satz von Kuratowski* zweimal hochskalieren).
Eine Möglichkeit die Anzahl der notwendigen Skalieroperationen zu reduzieren, wäre eine *Struktur-Reduktion*. Der *Satz von Kuratowski* könnte auch mit weniger Struktur dargestellt werden:

- Satz von Kuratowski
  - Behauptung: Graph ist planar $\Leftrightarrow$ weder K_5, noch K_{33} sind Minor des Graphen
  - Beweis
    - $\Rightarrow$ (die eine Richtung des Beweises)
    - $\Leftarrow$ (die andere Richtung des Beweises)
  - Verwendung: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Ein Nachteil der Struktur-Reduktion ist, dass dadurch weniger Semantik abgebildet wird. Es scheint Paradox, dass ein *Mehr* an Semantik zu einer Verschlechterung der Benutzbarkeit führen kann.
Das *mehrfache Skalieren* ermöglicht es, die Feingranularität von Informationsstrukturen zu *nutzen* – es wird ein feingranulareres Skalieren möglich. Ohne *mehrfaches Skalieren* wird die Benutzung durch Feingranularität hingegen aufwendiger.
Bei *Baumansichten* ist *mehrfaches Skalieren* meist nicht implementiert, wäre konzeptionell jedoch genauso wie bei DiKo möglich. Beim Gliederungseditor *Workflowy*[41] ist beispielsweise ein ähnliches Prinzip implementiert.[42]

[41] Siehe `https://workflowy.com/`

[42] Bei Workflowy kann das Wurzel-Objekt mehrfach skaliert werden. Es gibt geringfügige Abweichungen vom hier erklärten Prinzip. Warum das mehrfache Skalieren nur beim Wurzel-Objekt möglich war, konnte nicht herausgefunden werden. Versuche mit der DiKo-Implementierung haben diesbezüglich keine Probleme gezeigt.

6 Theoretische Evaluation und verwandte Arbeiten

In diesem Kapitel wird DiKo in die wissenschaftliche Landschaft eingeordnet. In Abschnitt 6.1 wird der Zusammenhang zu verwandten Arbeiten und Publikationen im Bereich der *Informationsvisualisierung* gezeigt. In Abschnitt 6.2 folgt eine Untersuchung im Bereich der Mensch-Maschine-Interaktion.
Es wird gezeigt, dass DiKo einige Anforderungen erfüllt, welche vom Doug-Engelbart-Institut an eine moderne Benutzerschnittstelle für das World-Wide-Web gestellt werden (Abschnitt 6.3).

Eine psychologische Begründung für *Kontexterhaltung* liefert die Arbeit [Rei99], welche sich mit *Didaktischem Design*[43] auseinander setzt. Diese begründet, dass neue Inhalte möglichst im Kontext zu präsentieren sind. Weitere Details sollen erst in einem weiteren Schritt präsentiert werden.

6.1 Informationsvisualisierung

Da DiKo keine typischen zwei- oder dreidimensionale *Grafiken* enthält, wird zunächst untersucht, inwiefern DiKo trotzdem der *Informationsvisualisierung* zuzuordnen ist.
Bei der Informationsvisualisierung geht es um die schematische, interaktive Präsentation von Informationen.[44]. In der Arbeit [Wil03] werden auch einfachere Darstellungen, wie „tables“, der Informationsvisualisierung zugeordnet.
Bei DiKo werden rekursive Detail-Beziehungen durch Einrückungen und verschachtelte Kärtchen dargestellt. Dies kann als *schematische* Darstellung bezeichnet werden. Alle Objekte können hoch- bzw. herunterskaliert werden, weshalb DiKo als *interaktiv* bezeichnet werden kann. Das semantische Skalieren erinnert an das grafische Zoomen von Bildern. Außerdem kann DiKo auch als *interaktives Verzeichnis* (engl. *table*) bezeichnet werden. DiKo ist also in der Informationsvisualisierung zu verorten.
In Abschnitt 6.1.1 werden Publikationen, welche ebenfalls mit der Integration von Details in den Kontext zu tun haben, ausgewertet.
Der Zusammenhang zum *Visual Information Seeking Mantra* wird in Abschnitt 6.1.2 kritisch analysisiert.

[43] Siehe auch `https://de.wikipedia.org/wiki/Instruktionsdesign`

[44] Zwei entsprechende Zitate: „Information visualization (InfoVis) is the communication of abstract data through the use of interactive visual interfaces.“ [KMSZ06], „Visual representations of the semantics, or meaning, of information. In contrast to scientific visualization, information visualization typically deals with nonnumeric, nonspatial, and high-dimensional data.“ [Che05]

6.1.1 Focus+Context

Ausführliche Recherche- bzw. Forschungsergebnisse über Techniken zum Anzeigen von Kontext- und Detail-Objekten finden sich in [CKB09]. Dort werden drei Kategorien genannt: *Overview+Detail*, *Zooming* und *Focus+Context*. Diese scheinen als die wichtigsten Kategorien wahrgenommen zu werden. In jeder Kategorie werden Vor- und Nachteile gefunden, es gibt am Ende keinen *Favoriten*. Deshalb ist es interessant, dass DiKo eine Focus+Context-Technologie ist, welche jedoch keine der dargestellten Nachteile hat. Dies wird im Folgenden gezeigt.
Der Zweck von Focus+Context-Techniken ist die *Integration von Details in den Kontext*. Dadurch soll die räumliche bzw. zeitliche Trennung von Kontext- und Detailansicht aufgehoben werden. (Bei *Overview+Detail* gibt es die *räumliche* Trennung, beim *Zoomen* die *zeitliche* Trennung).
In der Arbeit werden vor allem Fischaugenansichten und *verallgemeinerte* Fischaugenansichten untersucht.[45] Die Verzerrung erschwere dabei die Navigation in der Informationsstruktur (das „targeting“ spiele dabei eine Rolle). Außerdem könne der Benutzer räumliche Eigenschaften nicht mehr richtig einschätzen (was offensichtlich auf die Verzerrung zurückzuführen ist).[46]
DiKo erfüllt ebenfalls den Zweck, Details in den Kontext zu integrieren, hat jedoch keine der Nachteile, welche mit der Verzerrung bzw. verzerrungsähnlichen Effekten zu tun haben. Bei DiKo hat der Benutzer die direkte Kontrolle darüber, welche Objekte angezeigt werden.[47] Bei (verallgemeinerten) Fischaugenansichten berechnet jedoch der Computer, welche Objekte zu sehen sind, bzw. wie groß diese Objekte dargestellt werden.
Bei DiKo kann ein Objekt prinzipiell beliebig *vergrößert* werden. Bei den verzerrungsbasierten Techniken wird die Darstellung hingegen desto verzerrter, je mehr *vergrößert* wird.
In der Arbeit ([CKB09]) werden Eigenschaften erwähnt, welche helfen, Focus+Context-Technologien zu klassifizieren. DiKo kann mit folgenden Eigenschaften klassifiziert werden:

[45] Die Arbeit [Fur86] wird als theoretischer Unterbau für Focus+Context-Techniken genannt. Darin wird die *verallgemeinerte* Fischaugenansicht vorgestellt. Tatsächlich wird in [Fur86] als Hauptbeispiel ein hierarchisch-organisierter Quellcode mit einer *verallgemeinerten* Fischaugenansicht dargestellt. 20 Jahre später hat Furnas eine anknüpfende Arbeit [Fur06] geschrieben, welche deutlich hervorhebt, dass die *verallgemeinerte* Fischaugenansicht nichts mit dem grafischen Fischaugeneffekt zu tun hat.

[46] „Even in non-spatial domains, evaluations of distortion-based techniques have largely failed to provide evidence of performance enhancements, and several studies have shown that fisheye views can damage fundamental components of interaction such as target acquisition.“ Das Zitat zeigt insbesondere, dass es in dieser Arbeit auch um *non-spatial domains* geht.

[47] Die Kontrolle durch den Benutzer wird auch in Nielsens bekannten *10 Usability-Heuristiken* hervorgehoben, siehe Abschnitt 6.2).

selective presentation:
Bei DiKo werden die unteren Objekte nach unten verschoben, wodurch Objekte am unteren Rand aus dem sichtbaren Bereich verschwinden können. Diese Eigenschaft steht im Gegensatz zu verzerrungsbasierten Techniken, bei denen das Platz-Problem durch eine verzerrte Darstellung der umliegenden Objekte gelöst wird, jedoch keine Objekte verschwinden.

human controlled:
Genauso, wie bei der *Baumansicht*, kann der Benutzer direkt kontrollieren, welche Objekte angezeigt werden und welche nicht. Es gibt keine *degree-of-interest*-Funktion zur *Berechnung*, welche Objekte auf der Benutzeroberfläche zu sehen sein sollen.

multiple focus:
Es können unabhängig voneinander Details zu verschiedenen Objekten angezeigt werden.[48]

non-spatial domain:
Die Datenbasis von DiKo ist diskret und keine 2D-Landschaft.

6.1.1.1 Hierarchische Strukturen

In [CS94] (Chimera, Shneiderman) werden drei *Interfaces* (Benutzerschnittstellen) für hierarchische Informationsstrukturen verglichen. Eines der *Interfaces* ist die Baumansicht, welche große Ähnlichkeiten zu DiKo hat (siehe Abschnitt 4.3).
Die Arbeit betont die Wichtigkeit von animierten Übergängen beim *expand/contract*-Vorgang, was dem Hoch- bzw. Herunterskalieren bei DiKo entspricht. Wegen der großen Ähnlichkeiten ist zu vermuten, dass diese Beobachtung auch für DiKo gilt. Es wird argumentiert, dass die Animationen dem Benutzer helfen, zu sehen, wo neue Inhalte auftauchen. Ohne Animationen verliere der Benutzer komplett die Orientierung. „We conjecture that if an expansion (contraction) was performed with an entire screen repaint, it would require complete reorientation to identify where the new information appeared.“ Diese extreme Sichtweise konnte bei Untersuchungen von entsprechenden Baumansichten jedoch nicht bestätigt werden.
Außerdem wird ähnlich wie in Abschnitt *Kontexterhaltende Interaktion* 1.1.2 argumentiert, dass es wichtig sei, dass der Benutzer die übergeordnete Struktur im Blick behält. Man könne schnell vergessen, wo man sich gerade befindet, wenn der Kontext nicht mehr zu sehen ist. „When viewing a section that is longer than one screen, users can quickly forget what part of the document is being viewed.“ Die Arbeit betont ebenfalls die Wich-

[48] Es können auch Details zu Objekten, welche selber Details eines anderen Objektes sind, angezeigt werden (Rekursives Prinzip, siehe Abschnitt 1.1.1).

tigkeit von (visuellen) Ankerpunkten, um die Orientierung zu behalten.[49]
Die Arbeit [PGB02] stellt eine besonders aufwendig animierte, interaktive Darstellung von hierarchischen Gliederungen vor. Es werden drei vergleichbare Anwendungen untersucht. Sie kommen zu dem Schluss, dass Testpersonen vor allem die animierte Darstellung gefällt (insbesondere im Vergleich zur Baumansicht, welche DiKo ähnelt.) Dies motiviert auch bei DiKo das *visuelle Design* zu verbessern (siehe auch Abschnitt 3.1) und animierte Übergänge zu implementieren.

6.1.2 Analyse: *Visual Information Seeking Mantra*

Es soll Shneidermans vielzitiertes *Visual Information Seeking Mantra* [Shn96] analysiert und ein interessanter Zusammenhang mit DiKo gezeigt werden.
Die Arbeit [CC05] kritisiert, dass das *Visual Information Seeking Mantra* häufig falsch angewendet würde. Deshalb soll zunächst geklärt werden, worum es sich bei dem Prinzip überhaupt handelt.
Im Folgenden wird gezeigt, dass das „Mantra“ eher eine Ideensammlung und Inspirationsquelle und keine fundierte wissenschaftliche Anleitung für Informationsvisualisierung darstellt:
Das *Visual Information Seeking Mantra* lautet: *Overview first, zoom and filter, then details-on-demand*. Die Verwendung des Begriffs *Zoom* lässt die Frage aufkommen, ob das Prinzip überhaupt auf nicht-grafische Darstellungen wie DiKo anwendbar ist. In [Shn96] wird das Prinzip jedoch als allgemein gültig dargestellt. Die untersuchten Daten-Typen wären eine Abstraktion der Realität. „These seven data types reflect [sic] are an abstraction of the reality.“[50]
Die Arbeit sei eine Möglichkeit, seine *Ideen* zu präsentieren, wie Shneiderman im Nachwort erklärt.
Die *Tasks* (darunter auch die Teile des „Mantras“) werden mit den Worten eingeleitet: „Some *idea* of missed opportunities emerges in looking at the tasks and data types in depth“ (Hervorhebung nachträglich).
Das „Mantra“ wird mit zehn gleichen Zeilen *Overview first, zoom and filter, then details-on-demand* eingeleitet. Es wird erklärt: „Each line represents one project in which I found myself rediscovering this principle and therefore wrote it down it [sic] as a reminder.“ Diese ungewöhnliche Ausdrucksweise verleiht der Arbeit ebenfalls den Charakter einer Ideensammlung. Die ersten zwei Sätze der Arbeit enthalten einen Hinweis darauf, dass diese einen „starting point“ darstellt. Auch dies lädt dazu ein, die Arbeit als Ideensammlung

[49] [...] The expand/contract [interface provides] an additional reinforcement and anchor point (users are cognizant of the [...] item and hopefully its location because they clicked on it) which contributes to users' awareness of their location within the document.

[50] In der Arbeit wird aber stellenweise implizit angenommen, dass die Informationen in einer 2D-Landschaft dargestellt werden (So auch im *overview*-Abschnitt).

zu betrachten.[51]
Die Arbeit [CC05] präsentiert eine kritische Untersuchung des „Mantra“. Es sei unklar, für welche Probleme die jeweiligen Teile des „Mantras“ verwendet werden sollen und ob alle Teile notwendig für ein Informationsvisualisierungs-System sind. Es wird dort kritisiert (Hervorhebungen im Nachhinein):

> While the methodological suggestions are generally accepted as valid, they have not been subjected to adequate and rigorous assessment. The Mantra appears to act as an *inspiration* and guideline for practitioners *rather than a scientifically studied methodological approach.*

Das Publikationsjahr 1996 lässt die Frage aufkommen, ob die Arbeit auf dem Gebiet der Informationsvisualisierung noch zeitgemäß ist. Die Arbeit wurde zu einem Zeitpunkt geschrieben, in der grafische Informationsvisualisierungen erst am Entstehen waren („three dimensions are still novel“). Die Beschreibung des Zoomens *ohne* Mausrad wirkt veraltet. Die vorangehenden Überlegungen zeigen, dass das *Visual Information Seeking Mantra* nicht einschränkend wirken soll und nicht als Rechtfertigung für das Design herhalten darf.[52] Als *Inspirationsquelle* ist es relativ frei interpretierbar. Darauf aufbauend wird im Folgenden ein interessanter Zusammenhang mit DiKo gezeigt.

Overview first:
Das rekursive semantische Skalieren ermöglicht, dass der Benutzer sich zu einem Objekt zuerst die Übersicht anzeigen lassen kann. Beispiel: Bei einer Gliederung sieht er in Skalierungsstufe eins die erste Ebene.

Zoom:
Die Motivation des Zoomens ist, dass der Benutzer diejenigen Objekte vergrößern kann, für die er sich interessiert. Das semantische Skalieren dient dem gleichen Zweck. In [CC05] wird auf die subtile Frage hingewiesen, ob das Zoomen so zu verstehen ist, dass dabei automatisch die Objekte am Rand den sichtbaren Bereich verlassen. Da die Arbeit [Shn96] auch das Fisheye-Zoomen thematisiert, bleibt dies im „Mantra“ unklar.

Filter:
Das *Herunterskalieren* kann als Herausfiltern uninteressanter Objekte gesehen werden.

Details-on-demand:
Dieses Prinzip beschreibt *scheinbar* das Anzeigen von Details ohne Kontext-Wechsel (wird

[51] Auch das folgende Zitat weist auf einen entsprechenden Charakter der Arbeit [Shn96] hin: „This taxonomy is useful only if it facilitates discussion and leads to useful discoveries.“

[52] Auch in [CC05] wird festgetellt, dass das *Visual Information Seeking Mantra* nicht als „justification“ zu verwenden ist.

gleich erklärt). Zunächst scheint es das Prinzip *Details im Kontext anzeigen* ziemlich genau zu treffen, wie auch folgendes Zitat aus [Hin06] zeigt: „With this method the user can acquire detailed information with a simple action and *without changing the representational context.*"(Hervorhebung nachträglich). Auch in der kritischen Auseinandersetzung mit dem „Mantra" in [CC05] wird auf die Kontexterhaltung in diesem Zusammenhang hingewiesen.
Trotzdem wurde in dieser Arbeit bewusst darauf verzichtet, diesen Begriff zu verwenden. In der ursprünglichen Arbeit [Shn96] wird die Kontexterhaltung überhaupt nicht erwähnt. Das heißt die Eigenschaft den Kontext zu erhalten geht nicht aus der ursprünglichen Definition hervor und wird lediglich *implizit* mit dem Begriff verbunden.
Außerdem wirkt das *details-on-demand* als letzter Schritt des inhaltlichen Vertiefens. Das Besondere an DiKo ist hingegen das *rekursive* semantische Skalieren. Interessant ist jedoch, dass darauf hingewiesen wird, dass die Details wiederum Hypertext, mit Links auf weitere Informationen, enthalten können[53].
Bei DiKo werden alle Teile des „Mantras" als gleichartig angesehen und mit *einem* Prinzip realisiert.
DiKo kann also teilweise gut mit den Begriffen des „Mantras" beschrieben werden – gleichzeitig zeigen sich jedoch bedeutende Unterschiede. Trotzdem wird das „Mantra" erfüllt.

6.2 Mensch-Maschine-Interaktion

Die *Mensch-Maschine-Interaktion* ist das der *Informationsvisualisierung* übergeordnete Forschungsgebiet. Die *10 Usability Heuristics for User Interface Design* ([Nie95]) stellen grundlegende Erkenntnisse aus diesem Bereich dar[54]. Deshalb soll untersucht werden, inwiefern DiKo diesen Heuristiken entspricht.
Es werden nur diejenigen Heuristiken erwähnt, welche einen direkten Zusammenhang mit DiKo haben.

Match between system and the real world:
Es wäre wichtig, dass das System Konzepte verwendet, welche dem Benutzer vertraut sind. DiKo wird durch *Hoch-* und *Herunterskalieren* bedient. Dies entspricht dem vertrauten Konzept des grafischen Vergrößerns bzw. Verkleinern, welches zum Beispiel auch beim grafischen Zoomen eines Bildes oder eines Dokumentes verwendet wird.
In Umfragen für qualitatives Feedback wurde das System jedoch teilweise als ungewohnt

[53] In Abschnitt 1.1.1 wurde gezeigt, dass es wichtig ist, dass auch Details zu Details angezeigt werden können (*Rekursives* Prinzip).

[54] Die Vorlesung *Mensch-Maschine-Interaktion* am *Karlsruher Institut für Technologie* empfiehlt die sogenannten *10 Usability Heuristics for User Interface Design* (siehe auch [Nie94]), als Richtlinien beim Design von Mensch-Maschine-Systemen.

beschrieben. Die Befragten waren eine normale Baumansicht[55] gewohnt.

User control and freedom:
Bei DiKo hat der Benutzer viel Kontrolle, er kann sich flexibel jedes Objekt entsprechend seinem Informationsbedarf zurecht skalieren.
Die Objekte werden jedoch automatisch in einer hierarchischen Struktur angeordnet. Dies bedeutet ein Defizit an Freiheit und motiviert die Erweiterung von DiKo: Es sollte möglich sein, beliebige Objekte per Drag-and-Drop aus der DiKo-Ansicht heraus zu ziehen und in einer eigenen DiKo-Ansicht darzustellen. Siehe auch Abschnitt 4.2.
„Users often choose system functions by mistake and will need a clearly marked ‚emergency exit' to leave the unwanted state without having to go through an extended dialogue.":
Wenn der Benutzer sich „verklickt" und aus Versehen das falsche Objekt hochskaliert, ist dies nicht weiter tragisch. Der Kontext bleibt erhalten (siehe Abschnitt 1.1.2). Dadurch behält der Benutzer auch bei einem Fehler die Übersicht. (Das *Hochskalieren* kann auch leicht durch die gegenteilige Operation *Herunterskalieren* rückgängig gemacht werden.)
Dieser Sachverhalt konnte auch in Umfragen für qualitatives Feedback beobachtet werden: Es wurde das Problem gesehen, dass man bei Wikipedia-ähnlichen Systemen die Orientierung verliert, wenn man im Inhaltsverzeichnis aus Versehen auf den falschen Abschnitt klickt. Bei DiKo trete das Problem nicht auf.

Consistency and standards:
Details – im verallgemeinerten Sinne[56] – werden immer gleich dargestellt: als relativ kleine Fläche direkt unter dem skalierten Objekt. Es sei hervorgehoben, dass der Begriff *Detail* in einem sehr allgemeinen Sinn verwendet wird und somit ein großer Teil der Interaktion mit der Benutzeroberfläche mit *einem konsistentem* Prinzip abgedeckt ist. Die Bedienung geschieht immer mit den gleichen Operationen: Hoch- bzw. Herunterskalieren.

Recognition rather than recall:
Bei DiKo kann ein Objekt zu unterschiedlichen Zeitpunkten sehr unterschiedlich aussehen: Jedes enthaltene Objekt kann in unterschiedlichen Skalierungsstufen dargestellt sein. Objekte welche zu einem Zeitpunkt direkt nebeneinander positioniert waren, können zu einem anderen Zeitpunkt weit voneinander entfernt liegen (wenn das vorhergehende Objekt hochskaliert wurde). Testpersonen wiesen darauf hin, dass das Fließ-Layout den Nachteil habe, dass Objekte nicht immer die gleiche vertikale Position haben. Zu einem Zeitpunkt könne ein Objekt am Anfang einer Zeile sein und zu einem anderen Zeitpunkt in der Mitte der Zeile. Es wäre zu untersuchen, inwiefern darunter die *Wiedererkennung* leidet.
Es wird aber auch gefordert, dass der Benutzer sich (zum Beispiel) nicht die Informationen von zwei verschiedenen Teilen eines Dialoges merken muss. Der Gedanke von DiKo

[55] Siehe Abschnitt 4.3
[56] Siehe Abschnitt 1.1.1

ist, dass alle Objekte unabhängig voneinander zurecht skaliert werden können. Dadurch kann der Benutzer sich flexibel diejenigen Objekte gleichzeitig anzeigen lassen, an denen er interessiert ist. In Umfragen für qualitatives Feedback wurde es als besonders nützlich empfunden, wenn man sich in einem wissenschaftlichen Dokument genau die Stellen *hochskalieren* kann, für die man sich interessiert und dadurch die entsprechenden Abschnitte direkt miteinander vergleichen kann (zum Beispiel verschiedene Definitionen).

6.3 Wissenssysteme

In [ohs] (Doug Engelbart Institute) wird beobachtet, dass Wissenssysteme aus feingranularen Bausteinen aufgebaut sind (bzw. sein sollten). Insbesondere wird das World-Wide-Web als solch ein Wissenssystem aufgefasst. Es stellt sich die Frage, ob DiKo eine geeignete Benutzerschnittstelle für diese Systeme ist. Im Artikel ist von *fine-grained browsing* die Rede. Es wird insbesondere auch das flexible Anzeigen von Objekten in unterschiedlichen Detaillierungsstufen gefordert. Das folgende Zitat soll dies verdeutlichen (Hervorhebungen nachträglich):

> We'll need more facile ways to traverse our knowledge domains, as if we are *flying* around in an information space. We need to be able to quickly skim across the landscapes, and *dive down* into whatever detail suits our needs in the moment, *zooming in and out of detail* as desired. Basic [...] requirements include birds-eye views that scale, and *fine-grained* addressability of all objects and media types, [...] providing flexible and facile ways of viewing, skimming, traversing [...] the knowledge that our work depends on.

DiKo erfüllt diese Anforderungen: Mit *mehrfachem* Skalieren kann der Benutzer sich bequem größere Datenmengen anzeigen lassen, welche er dann mit *Scrollen* „überfliegen" kann. Das *semantische Skalieren* ermöglicht dem Benutzer den Detaillierungsgrad jedes Objektes flexibel zu verändern. Mit *rekursivem* Skalieren, kann der Benutzer beliebig tief in eine Struktur „eintauchen". Das *mehrfache Skalieren* ermöglicht die effiziente Visualisierung von *feingranularen* Strukturen[57].

[57] Siehe Abschnitt *mehrfaches Skalieren* 5.2

7 Schlussbetrachtung

Das Ziel dieser Arbeit war es, ein Konzept zu entwickeln und zu untersuchen, welches die Anforderungen aus Abschnitt *Bedarf an Details* 1.1.1 und *Kontexterhaltende Interaktion* 1.1.2 erfüllt. Die Anforderungen waren:

- Es muss möglich sein, flexibel zu Objekten auf der Benutzeroberfläche weitere Inhalte anzuzeigen. Die Inhalte, welche über eine Objekt erreicht werden, wurden dabei als dessen *Details* bezeichnet.
- Das Anzeigen der *Details* soll *kontexterhaltend* geschehen. Der Benutzer soll sowohl auf visueller Ebene, als auch auf logischer Ebene, den Kontext nicht verlieren.
- Es soll auch möglich sein, Details zu Objekten, welche selber Details sind, anzuzeigen. Dies wurde als *rekursives* Prinzip bezeichnet.

Es wurde gezeigt, dass das neu entwickelte Konzept, genannt DiKo, für viele Informationsstrukturen, darunter Gliederungen, Hypertexte, Dokumente und semantische Daten verwendet werden kann. Die Anwendbarkeit auf Tabellen und Bilder wurde nicht untersucht.
Mit DiKo können zu allen Objekten (auf der Benutzeroberfläche) Details auf kontexterhaltende Weise angezeigt werden. Auch der *verlinkte Inhalt* eines *Links* in einem Text kann in den *Text* integriert werden. Abbildung 4 zeigt, wie ein verlinktes Dokument direkt unter dem Link-Wort eingefügt wurde.
Das Prinzip *Einfügen von Details beim Fließ-Layout* aus Abschnitt 2.4.1 ermöglicht die Verwendung des Fließ-Layouts. Dadurch kann DiKo den zweidimensionalen Raum nutzen, während die weitverbreitete Baumansicht[58] seine Objekte lediglich in einer Dimension, nämlich der Vertikalen, anordnet.[59]
Es ist auch möglich Details zu Details auf kontexterhaltende Weise anzuzeigen (rekursives Prinzip), jedoch nur bis zu einer beschränkten Rekursionstiefe.[60]
Es wurde gezeigt, dass DiKo Vorteile gegenüber vergleichbaren Techniken hat. Pop-Ups[61] verdecken einen Teil der bestehenden Ansicht. Im Vergleich zu fensterartigen Systemen[62] werden bei DiKo die Beziehungen zwischen Kontext und Detail auf intuitive und konsistente Weise dargestellt. DiKo nutzt den verfügbaren Platz in der Horizontalen wesentlich besser als die Baumansicht. Eine Kombination der bestehenden Konzepte hat gegenüber DiKo den Nachteil, dass Details nicht auf *konsistente* Weise angezeigt werden[63].

[58] Siehe Abschnitt 4.3
[59] Die Platzausnutzung von DiKo wurde in Abschnitt 4.3 untersucht.
[60] Siehe Abschnitt 3.2
[61] Siehe Abschnitt 4.1
[62] Siehe Abschnitt 4.2
[63] Siehe Abschnitt 4

Bei der Untersuchung der Baumansicht und DiKo wurden einzelne konkrete Probleme gefunden. Für diese wurden Lösungen erarbeitet, welche in Abschnitt 5 vorgestellt wurden. Diese sind auch für die Baumansicht relevant. Das semantische Skalieren diente zunächst als Vorüberlegung, um die Anforderungen zu erfüllen. Abschnitt *mehrfaches Skalieren* 5.2 zeigte jedoch, dass dieses besonders wichtig ist, um mit feingranularen Strukturen bequem arbeiten zu können.

7.1 Ausblick: Kontexterhaltendes Scrollen

In Abschnitt 5.1.2 wurde das Problem erklärt, dass übergeordnete Objekte aus dem sichtbaren Bereich verschwinden, wenn man nach unten scrollt. Hier wird eine Lösung vorgestellt, welche im Rahmen dieser Arbeit jedoch nicht tiefergehend untersucht wurde.
Eine einfache Lösung ist die Folgende: Am oberen Rand wird eine sogenannte Brotkrümel-Navigationsleiste angezeigt. Diese enthält genau diejenigen übergeordneten Objekte, welche durch das Scrollen nach unten nicht mehr zu sehen sind. In der Brotkrümel-Navigationsleiste sind die Objekte mit dem Fließ-Layout angeordnet.
Diese einfache Lösung bedeutet, dass ein zweiter Abschnitt im DiKo-Fenster zu sehen ist. Die übergeordneten Objekte wären *wechselweise* in der normalen DiKo-Ansicht und in der Brotkrümel-Navigationsleiste zu sehen. Insgesamt würde dies einige mentale Anstrengung für den Benutzer bedeuten.
Deshalb wird hier eine verbesserte Lösung vorgestellt, welche prinzipiell ähnlich funktioniert.
Hierbei wird das Scrollen *semantisch umgedeutet*. Das *semantische Scrollen* nach unten bedeutet: Am oberen Rand werden geeignete Objekte komprimiert dargestellt. Gleichzeitig werden am unteren Rand entsprechend Objekte expandiert, welche zuvor komprimiert waren und die Objekte in der Mitte, werden nach oben verschoben.
Mit *geeignete Objekte* sind Objekte gemeint, welche keine übergeordneten Objekte sind.
Das semantische Scrollen *nach oben* verhält sich analog.[64]
Es müsste genauso *gleichmäßig* funktionieren, wie das grafische Scrollen.
Das semantische Scrollen würde also das Problem, dass die übergeordneten Objekte wechselweise in einem Extra-Abschnitt angezeigt werden, verhindern.
Es bleibt zu untersuchen, inwiefern sich das neue Scroll-Konzept umsetzen lässt, dadurch der übergeordnete Kontext beim Scrollen nicht verloren geht und die Benutzbarkeit insgesamt verbessert wird.

[64] Es würde sich also um eine vertikale (verallgemeinerte) Fischaugenansicht auf eine DiKo-Darstellung handeln, bei der die „Verzerrung“ am oberen und unteren Rand auftritt. Siehe auch [Fur86].

Literatur

[BGBS02] Baudisch, Patrick, Nathaniel Good, Victoria Bellotti und Pamela Schraedley: *Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, Seiten 259–266, New York, NY, USA, 2002. ACM.

[CC05] Craft, B. und P. Cairns: *Beyond guidelines: what can we learn from the visual information seeking mantra?* In: *Ninth International Conference on Information Visualisation (IV'05)*, Seiten 110–118, July 2005.

[Che05] Chen, C.: *Top 10 unsolved information visualization problems.* IEEE Computer Graphics and Applications, 25(4):12–16, July 2005.

[CKB09] Cockburn, Andy, Amy Karlson und Benjamin B. Bederson: *A Review of Overview+Detail, Zooming, and Focus+Context Interfaces.* ACM Comput. Surv., 41(1):2:1–2:31, Januar 2009.

[CS94] Chimera, Richard und Ben Shneiderman: *An Exploratory Evaluation of Three Interfaces for Browsing Large Hierarchical Tables of Contents.* ACM Trans. Inf. Syst., 12(4):383–406, Oktober 1994.

[Fur86] Furnas, G. W.: *Generalized Fisheye Views.* SIGCHI Bull., 17(4):16–23, April 1986.

[Fur06] Furnas, George W.: *A Fisheye Follow-up: Further Reflections on Focus + Context.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, Seiten 999–1008, New York, NY, USA, 2006. ACM.

[Hal11] Haller, Heiko: *User Interfaces for Personal Knowledge Management with Semantic Technologies.* Dissertation, Karlsruher Institut für Technologie, 2011.

[HBP02] Hornbæk, Kasper, Benjamin B. Bederson und Catherine Plaisant: *Navigation Patterns and Usability of Zoomable User Interfaces with and Without an Overview.* ACM Trans. Comput.-Hum. Interact., 9(4):362–389, Dezember 2002.

[Hin06] Hinum, Klaus: *Gravi++ An Interactive Information Visualization for High Dimensional, Temporal Data.* 2006.

[KMSZ06] Keim, D. A., F. Mansmann, J. Schneidewind und H. Ziegler: *Challenges in Visual Data Analysis.* In: *Tenth International Conference on Information Visualisation (IV'06)*, Seiten 9–16, July 2006.

[Nie94] Nielsen, Jakob: *Usability Inspection Methods.* In: *Conference Companion on Human Factors in Computing Systems*, CHI '94, Seiten 413–414, New York, NY, USA, 1994. ACM.

[Nie95] Nielsen, Jakob. `https://www.nngroup.com/articles/ten-usability-heuristics/`, 1995. [Online; Zugang: 05.03.17].

[Nor88] Norman, Donald: *The Design of Everyday Things.* Basic Books, 1988.

[ohs] `http://www.dougengelbart.org/about/ohs.html`. [Online; Zugang: 05.03.17].

[PGB02] Plaisant, Catherine, Jesse Grosjean und Benjamin B. Bederson: *SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation.* In: *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, Seiten 57–, Washington, DC, USA, 2002. IEEE Computer Society.

[Rei99] Reigeluth, C. M.: *The elaboration theory: Guidance for scope and sequences decisions.* Reigeluth, C. M., editor, Instructional-design theories and models II: A new paradigm of instructional theory, Seiten 425–454, 1999.

[Shn96] Shneiderman, Ben: *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations.* In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, Seiten 336–, Washington, DC, USA, 1996. IEEE Computer Society.

[Wil03] Wilkins, Barry: *MELD: A Pattern Supported Methodology for Visualization Design.* 2003.