Regionales Undo/Redo für Multi-User-Anwendungen auf großen interaktiven Oberflächen

Christian Rendl

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2011

© Copyright 2011 Christian Rendl

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung–NichtKommerziell–KeineBearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe http://creativecommons.org/licenses/by-nc-nd/3.0/at/.

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 24. Juni 2011

Christian Rendl

Inhaltsverzeichnis

Vorwort

Ich bedanke mich bei meiner Familie und meiner Freundin Michaela für die wertvolle Unterstützung während des gesamten Master-Studiums. Des Weiteren danke ich Thomas Seifried, der mich bei der gesamten Arbeit und vor allem bei der komplexen technischen Umsetzung großartig unterstützt und mit mir in unzähligen Stunden Sonderfälle des selektiven Undos ausgetestet hat. Bei Florian Perteneder möchte ich mich für seine Ausdauer bei der Beantwortung unzähliger Fragen bedanken. Ein besonderer Dank gilt meinem Betreuer Prof. (FH) Univ.-Doz. Dr. Michael Haller, der bei der Betreuung meiner Arbeit stets ein offenes Ohr für meine Anliegen hatte und mich mit zahlreichen Ideen und Verbesserungsvorschlägen ausgezeichnet unterstützt hat. Abschließend bedanke ich mich noch bei allen weiteren Personen, die zur Erstellung meiner Arbeit beigetragen haben.

Kurzfassung

Interaktive Whiteboards werden im Unternehmenseinsatz immer beliebter. Aus diesem Grund erhält auch die Entwicklung von maßgeschneiderten Softwarelösungen für den Einsatz auf Whiteboards eine zunehmende Bedeutung. Dazu gehört auch die Integration von scheinbar einfachen Programmfeatures wie Undo und Redo. Während das Verhalten von Undo und Redo im Desktopbereich weitestgehend standardisiert ist, stellt sich bei verschiedensten Konstellationen von Gruppen- und Einzelarbeit an einer großen interaktiven Fläche die Frage, wie sich Undo und Redo grundsätzlich verhalten sollten. Dazu wird in dieser Arbeit eine Studie präsentiert, die die Erwartungshaltung der Benutzer in Bezug auf das Verhalten von Undo in Multi-User-Anwendungen aufzeigt. Die Studie ergab, dass regionales Undo am häufigsten der Intention der Benutzer entspricht. Gruppen erwarten sich ein auf ihre Gruppe bezogenes Undo, während Einzelpersonen ein benutzerbezogenes Undo bevorzugen. Die größte Schwierigkeit bei der Umsetzung eines regionalen Undo-Verfahrens stellt daher die möglichst genaue Ermittlung der Arbeitsbereiche von Gruppen und Einzelpersonen dar. In der Arbeit werden verschiedene Möglichkeiten der Regionsbestimmung genutzt, um verschiedene Techniken vorzustellen, wie regionales Undo in einer Multi-User-Anwendung auf einer großen interaktiven Fläche realisiert werden kann. Darüber hinaus wird ein generisches Undo-Framework vorgestellt, das einem Anwendungsprogrammierer die umfangreiche technische Basis zur Verfügung stellt, die benötigt wird, um regionales Undo in einer Multi-User-Anwendung umzusetzen.

Abstract

Due to the increasing popularity of interactive whiteboards in companies, the development of customized whiteboard software is becoming more and more important. This also includes the integration of apparently simple features such as undo and redo. Although the use of undo and redo in desktop applications has been largely resolved, the behavior within different combinations of group and individual use on large interactive surfaces is still uncertain. For this purpose a study was performed that evaluates the expectations of the users within multi-user scenarios. The survey suggests that regional undo fulfills most users' expectations. While groups expect a group-based undo, individual users prefer an user-based undo. The challenge in the implementation of regional undo is to accurately detect the working areas of groups and individual users on the interactive surface. In this Thesis, we developed several techniques of regional undo within a multi-user scenario. Thereby, different methods of region detection are used. Furthermore, we developed a generic undo-framework that provides the technical basis to implement regional undo in a multi-user application.

Kapitel 1

Motivation

Digitale Whiteboards und andere große interaktive Flächen erlangen vor allem im Unternehmenseinsatz eine immer stärkere Beliebtheit. Mittlerweile sind die Systeme so ausgereift, dass die Unternehmen durch die Einbindung von interaktiven Flächen in Meeting- oder Besprechungsszenarien einen Mehrwert und eine damit verbundene Steigerung der Produktivität und Ökonomie erreichen. Während sich die Forschung und Industrie in den letzten Jahren vorwiegend auf die Hardwaregestaltung solcher Systeme konzentriert hat, erlangt die Entwicklung von benutzerfreundlicher Software zunehmend an Stellenwert.

Dazu gehört auch die Einbindung von scheinbar einfachen Funktionalitäten wie Undo/Redo. Gerade Undo/Redo gehört mittlerweile in beinahe jeder Anwendung mitunter zu den wichtigsten Programmfeatures und ist für die Benutzerfreundlichkeit und Benutzbarkeit der jeweiligen Software von zentraler Bedeutung [30]. Undo/Redo wird dabei von den Benutzern vor allem zum Beseitigen von etwaigen Fehlern, zum Wiederherstellen eines früheren Dokumentstatus oder zum fehlerfreien Erlernen und Austesten von Programmfeatures genutzt [7]. Wichtig ist in jedem Fall, dass das Verhalten von Undo immer der Intuition des Benutzers entsprechen soll [1].

Während das Verhalten von Undo/Redo in Single-User-Anwendungen weitestgehend abgeklärt und standardisiert ist, stellt das Aufkommen von Mehrbenutzersystemen die traditionellen und erprobten Konzepte des Single-User-Undo in Frage.

Diese Arbeit beschäftigt sich zunächst mit den Problemen, die Undo/Redo in echten Multi-User-Anwendungen (mehrere Personen zur gleichen Zeit am gleichen Ort) verursachen. Darüber hinaus wird eine Studie präsentiert, die die Erwartungen der Benutzer in Bezug auf ein mehrbenutzerfähiges Undo in einem derartigen Szenario aufzeigt. Aufgrund dieser Erkenntnisse werden im weiteren Verlauf Konzepte und Lösungen vorgestellt, anhand derer sich Undo/Redo in einem Mehrbenutzersystem in annähernd jeder Situation so verhält, wie das die Benutzer intuitiv erwarten würden.

1.1 Überblick über bestehende Undo-Methoden

Da heutzutage beinahe jede Anwendung Undo und Redo in irgendeiner Weise anbietet, haben sich in der Vergangenheit verschiedene Verfahren entwickelt.

In den meisten Anwendungen wird ein linearer Verlauf (eng. History) eingesetzt, in dem die Operationen chronologisch nach Ausführungszeitpunkt abgelegt werden. Wird eine Operation aus der Vergangenheit rückgängig gemacht, werden neben der rückgängig gemachten auch alle chronologisch nachfolgenden Operationen im Verlauf rückgängig gemacht und in die Liste der verfügbaren Redo-Operationen übernommen. Dieses Vorgehen wird in der Fachsprache als *globales oder lineares Undo* bezeichnet [28].

Einige Anwendungen ermöglichen neben dem globalen Undo auch noch die Möglichkeit selektiv Operationen rückgängig zu machen. *Selektives Undo* [3, 18, 31] bedeutet, dass eine beliebige Operation aus dem Verlauf entfernt werden kann, ohne dass die nachfolgenden Operationen ebenso rückgängig gemacht werden müssen. Diese Technik stellt im Gegensatz zu einem linearen Verlauf einen erheblichen technischen Mehraufwand in der Implementierung dar. Außerdem muss die Anwendung eine entsprechende Visualisierung zur Verfügung stellen, mit Hilfe derer die zu rückgängig machende Operation möglichst bequem ausgewählt werden kann. Häufig wird für eine solche Visualisierung eine textuelle Beschreibung der Operationen gewählt, welche voraussetzt, dass der Benutzer die zu rückgängig machende Operation eindeutig identifizieren kann und aufgrund der textuellen Beschreibung in der Liste identifizieren kann. Ein derartiges Verhalten ist für den Benutzer nicht sonderlich intuitiv [14, 15].

Bedingt durch die stärkere Verbreitung von verteilten Anwendungen, die ein gleichzeitiges Arbeiten mehrerer Benutzer an einem gemeinsamen Dokument ermöglichen, werden die Anwendungsentwickler gezwungen, die bewährten Single-User-Konzepte auf den Einsatz in Multi-User-Anwendungen zu erweitern. Dabei bieten verteilte Anwendungen heutzutage meist ein *dokumentenbasiertes* (vgl. globales Undo, z. B. twiddla[1]) oder ein *benutzerbezogenes Undo* (z. B. Google Docs[2]) an. Im Gegensatz zu einem globalen Undo sind bei einem benutzerbezogenen Undo nur Operationen des jeweiligen Benutzers betroffen. Die große Schwierigkeit in der Entwicklung von Multi-User-Undo ist die stark steigende Komplexität, die durch das Auftreten von Konflikten und Fehlerfällen durch das gleichzeitige Arbeiten von mehreren Benutzern bedingt ist. Neben den Konflikten in der Kollaboration entsteht bei verteilten Anwendungen zusätzlich das Problem, dass wegen der Verteilung über das Netzwerk Latenzzeiten entstehen können und die richtige Reihenfolge und Synchronisierung der Operationen des Verlaufes jedes Clients gewährleistet werden muss. Mit der Lösung dieser Probleme beschäftigen

[1] http://www.twiddla.com/
[2] http://docs.google.com/

sich einige technische Lösungen [18, 21, 24, 27], welche in Kapitel 2 genauer beschrieben werden.

1.2 Ausgangslage

Diese Arbeit beschäftigt sich grundsätzlich mit einer ähnlichen Problemstellung wie verteilte Anwendungen, verfügt aber über eine gänzlich andere Ausgangslage. In den kommenden Abschnitten wird ebenso wie bei verteilten Anwendungen von Multi-User-Szenarien und -Anwendungen gesprochen. Jedoch ist dabei im Gegensatz zu verteilten Anwendungen eine tatsächliche Multi-User-Umgebung gemeint. Das bedeutet, dass nicht mehrere Benutzer verteilt mit der gleichen Anwendung (also von verschiedenen Clients aus) arbeiten, sondern dass vielmehr mehrere Benutzer zur gleichen Zeit am gleichen Ort mit der gleichen Anwendung arbeiten. Damit rücken Probleme wie die Netzwerksynchronisierung des Verlaufes in den Hintergrund, während der Undo-Methodik eine viel größere Bedeutung beigemessen werden muss.

Dabei beschäftigt sich diese Arbeit in erster Linie mit Gruppen- oder Einzelarbeit auf einer großen interaktiven Fläche im Rahmen von Meeting- oder Besprechungsszenarien und der bestmöglichen Undo-Methodik für solche Tätigkeiten. Ein Beispiel für eine derartige Tätigkeit wäre unter anderem das Bearbeiten oder Überzeichnen von großflächigen Dokumenten wie Plänen, Diagrammen oder Kartenmaterial auf einem digitalen Whiteboard (siehe Abbildung 1.1). In entsprechenden Situationen oder Szenarien werden von den Teilnehmern eines Meetings verschiedene Rollen eingenommen. Dabei

Abbildung 1.1: Die zentrale Fragestellung ist, wie sich eine Lösung für Undo und Redo auf einer großen interaktiven Fläche verhalten muss, um bei verschiedenster Zusammensetzung von Gruppen- und Einzelarbeit immer möglichst der Intuition und Erwartung der Benutzer zu entsprechen.

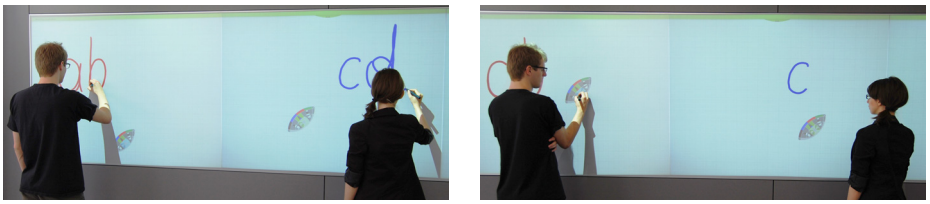
sind aktive und passive Teilnehmer möglich, wobei es vorkommen kann, dass passive Teilnehmer zu aktiven Teilnehmern und umgekehrt werden. Die Rollenverteilung der Meeting-Teilnehmer ist also während eines Meetings nicht fix vorgegeben, sondern ändert sich während eines Meetings mitunter mehrere Male. Ein Meeting läuft üblicherweise in mehreren Phasen ab, die nicht fix vorgegeben sind und von Meeting zu Meeting variieren. Häufig werden nach einer Diskussionsphase, die tendenziell von einer Person geleitet wird, die Aufgaben oder Diskussion in Gruppen- oder Einzelarbeit aufgeteilt. Somit ergeben sich verschiedenste Szenarien, indem die Diskussionsteilnehmer in verschiedener Zusammensetzung, in Gruppen- oder Einzelarbeit, an einem Problem oder einer Aufgabe arbeiten. Beispiele für häufige Zusammensetzungen auf einem interaktiven Whiteboard sind die gemeinsame Arbeit von mehreren Personen in einer Gruppe, die getrennte Arbeit von Einzelpersonen oder die gleichzeitige Arbeit einer Gruppe und einer Einzelperson (siehe Abbildung 1.1). Ein weiteres häufiges Szenario ist, dass zunächst mit Einzelarbeit begonnen wird, sich aber mit Fortdauer Abhängigkeiten und Zusammenhänge in den Aufgaben ergeben, die dazu führen, dass Einzelarbeit zur Gruppenarbeit wird.

1.3 Problemstellung

Aufgrund der verschiedenen Möglichkeiten der Kollaboration in einem solchen Szenario verhalten sich die in Abschnitt 1.1 erläuterten Undo-Ansätze nicht immer intuitiv für die Benutzer. Vielmehr muss für die verschiedensten Zusammensetzungen der Teilnehmer eines Meetings eine eigene Undo-Methodik entwickelt werden, die in jeder Zusammensetzung möglichst der Intuition und Erwartung der Teilnehmer entspricht. Die möglichen Probleme der bestehenden Undo-Methoden und die daraus resultierende Schlussfolgerung für eine neue Undo-Methodik werden in den nächsten Abschnitten erläutert.

1.3.1 Globales Undo

Die Anwendung eines *globalen oder dokumentenbasierten Undo* führt auf einer großen interaktiven Fläche dann zu Problemen, wenn mehrere Personen getrennt voneinander arbeiten und die Arbeit der anderen Personen nicht bewusst wahrnehmen. Angenommen eine Person möchte einen möglicherweise fehlerhaften Schritt rückgängig machen und eine andere Person tätigt währenddessen bereits weitere Arbeitsschritte. Zur Verwirrung beider Personen würde bei einem Undo ein Arbeitsschritt jener Person rückgängig gemacht, die ursprünglich gar nicht Undo aufgerufen hat (siehe Abbildung 1.2). Die Arbeit jener Person, die eigentlich das Undo veranlasst hat, bleibt hingegen unberührt. Dieses Verhalten ist für die Benutzer nicht intuitiv und würde nicht der Erwartung beider Personen entsprechen.



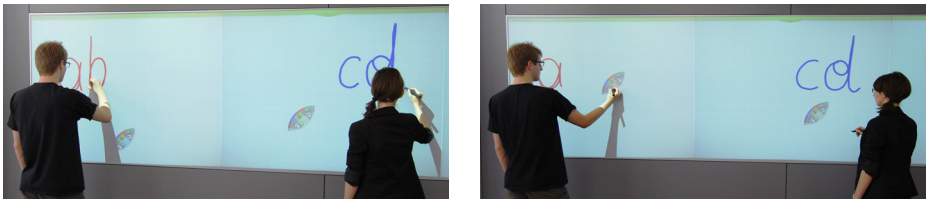
(a) (b)

Abbildung 1.2: Globales Undo führt in einem Multi-User-Szenario mit getrennt voneinander arbeitenden Personen (a) unweigerlich zu Problemen. Der linke Benutzer würde nicht erwarten, dass bei einem von ihm angestoßenen Undo-Vorgang ein Arbeitsschritt des rechten Benutzers (Element *d*) rückgängig gemacht wird (b).

1.3.2 Benutzerbezogenes Undo

Die einfachste Lösung für dieses Problem wäre ein *benutzerbezogenes Undo*. Das bedeutet, dass jeder Benutzer, der auf der interaktiven Fläche arbeitet, über seinen eigenen Verlauf verfügt und jeweils nur seine eigenen getätigten Schritte rückgängig machen kann (siehe Abbildung 1.3). Dies würde weitestgehend der Erwartung von getrennt arbeitenden Personen entsprechen.

Nichtsdestoweniger könnte auch ein benutzerbezogenes Undo wieder zu Problemen führen. Nämlich genau dann, wenn Personen gemeinsam auf der interaktiven Fläche zusammenarbeiten und einen zusammengehörigen Inhalt erstellen. Würde eine Gruppe beispielsweise einen früheren Zustand eines Dokuments wiederherstellen wollen, müssten die Gruppenmitglieder abwechselnd ihre Arbeitsschritte rückgängig machen (siehe Abbildung 1.4). Dies würde voraussetzen, dass die Benutzer zu jedem Zeitpunkt genau wissen, welche Person welchen Arbeitsschritt getätigt hat. Dies ist bei wenigen Schritten vielleicht noch denkbar, bei einer Arbeit über einen längeren Zeitraum aber schwierig. Unmöglich würde dies werden, wenn beispielsweise ein



(a) (b)

Abbildung 1.3: Benutzerbasiertes Undo würde die Probleme eines globalen Undo bei getrennt arbeitenden Personen (a) lösen. Der linke Benutzer macht seine eigenen Schritte (Element *b*) rückgängig, ohne die Arbeit des anderen auf der gleichen Arbeitsfläche arbeitenden Benutzers zu beeinflussen (b).

(a) (b) (c)

Abbildung 1.4: Benutzerbezogenes Undo würde jedoch wiederum bei kollaborativer Arbeit einer Gruppe (a) Probleme verursachen. Die Gruppe müsste, um beispielsweise einen früheren Dokumentstatus wiederherzustellen, abwechselnd die eigenen Undo-Schritte rückgängig machen (b, c). Das bedeutet, dass der rechte Benutzer die Erstellung der Elemente *b* und *d* und der linke Benutzer die Erstellung von Element *c* rückgängig machen müsste.

Gruppenmitglied die Gruppe während der Kollaboration verlässt oder ein Benutzer den als Identifikationsmerkmal genutzten digitalen Stift wechselt. In solchen Fällen wäre der Zugriff auf die Arbeitsschritte dieses Mitglieds „verloren“. Ein Gruppenmitglied würde sich daher eher erwarten, dass jedes Mitglied alle Schritte des gemeinsamen Inhaltes rückgängig machen kann.

Somit wäre im speziellen Fall von Gruppenarbeit wiederum globales Undo besser geeignet. Bei einer Vermischung der beiden Szenarien, also gleichzeitiger Arbeit einer Gruppe und einer Einzelperson oder der Arbeit von mehreren Gruppen würde sich aber keines der beiden genannten Verfahren mehr so verhalten, wie das die Benutzer am ehesten erwarten würden.

1.3.3 Regionales Undo

Da beide Verfahren aufgrund der verschiedenen Möglichkeiten der Kollaboration in dem Szenario aus Abschnitt 1.2 nie ausnahmslos jene Ergebnisse liefern, die der Erwartung der Benutzer entsprechen, wäre eine Mischform beider Verfahren wünschenswert.

Innerhalb einer Gruppe soll das System mit einem globalen, auf die Gruppe bezogenem Undo reagieren, während Einzelpersonen weiterhin über ein benutzerbezogenes Undo verfügen sollen. Bei diffiziler Betrachtung dieser Anforderungen ergibt sich als neue Undo-Form ein globales Undo für jede Region, in der von einer Gruppe oder einer Einzelperson gearbeitet wird (siehe Abbildung 1.5). Diese Form von Undo wird als *regionales Undo* bezeichnet, mit Hilfe dessen sich das System für die Benutzer wieder intuitiv und transparent verhalten würde. Die Schwierigkeit einer solchen Lösung besteht neben dem technischen Mehraufwand vor allem darin, zweifelsfrei die Grenzen der Arbeitsbereiche von Gruppen oder Einzelpersonen zu erkennen.

Abbildung 1.5: Bei Betrachtung der verschiedenen Möglichkeiten der Kollaboration ergibt sich als optimaler Ansatz ein globales Undo für jede Region, in der von Benutzern entweder alleine oder kollaborativ gearbeitet wird. Diese Methodik wird als regionales Undo bezeichnet.

1.4 Studie zur Erwartungshaltung

Um zu ermitteln, ob regionales Undo bei Kollaboration mehrerer Personen auf einer großen interaktiven Fläche tatsächlich die bestmögliche Methode ist, wurde im Rahmen dieser Arbeit eine empirische Studie durchgeführt.

Abowd und Dix stellten in ihrer Arbeit fest, dass das Verhalten von Undo immer der Erwartung bzw. der Intention der Benutzer entsprechen sollte [1]. Der Aufruf von Undo in einer Anwendung ist in den meisten Fällen mit einer konkreten Erwartungshaltung seitens des aufrufenden Benutzers verbunden, was der Undo-Aufruf im Dokument bewirken wird. Genau diese Erwartungshaltung in verschiedensten Gruppen- und Einzelszenarien abzufragen, war Ziel dieser Studie. Aus diesen Erwartungshaltungen lässt sich dann jene Undo-Methode ableiten, deren Verhalten die *meisten Übereinstimmungen* mit den Erwartungen der Benutzer erzielt.

Nachfolgende Hypothesen wurden im Vorfeld der Studie in Bezug auf die Arbeit von verschiedenen Personen in verschiedener Zusammensetzung auf einer großen interaktiven Fläche getroffen:

- **Hypothese 1 (*H1*):** *Regionales Undo* ist auf einer großen interaktiven Fläche bei verschiedenen Konstellationen von Gruppen- und Einzelarbeit die von den Benutzern am häufigsten erwartete Undo-Methodik.
- **Hypothese 2 (*H2*):** Benutzer erwarten, dass sich Undo auf *jene Bereiche* auswirkt, in denen sie *gerade arbeiten.*
- **Hypothese 3 (*H3*):** Benutzer, die *unabhängig (getrennt)* von einem anderen Benutzer am selben Dokument arbeiten, nehmen dessen *Arbeit nicht aktiv wahr.*

1.4.1 Testumgebung

Die Studie wurde an einem interaktiven Whiteboard ausgeführt, welches mit digitalen Anoto-Stiften[3] beschrieben werden kann. Die Abmessungen der interaktiven Fläche betrugen $4,5 \times 1,2$ Meter. Die Projektionsfläche bildeten drei Projektoren vom Typ Hitachi AP-100 mit einer Auflösung von 3072×768 Pixel, wobei die Projektor-Kanten nicht überblendet wurden. Die Probanden bzw. die Helfer der Studie konnten bei ihren Aufgabenstellungen die gesamte interaktive Fläche ohne Einschränkungen nutzen.

1.4.2 Studienteilnehmer

An der Studie nahmen 23 Personen mit einem Altersschnitt von 24,3 Jahren ($SA = 3,5$ Jahre) teil. Die 23 Teilnehmer setzten sich aus 17 männlichen und 6 weiblichen Teilnehmern zusammen. Bei den Teilnehmern handelte es sich zum überwiegenden Teil um Studenten mit computertechnischer Ausbildung (20 Teilnehmer) und drei Teilnehmern ohne technischer Ausbildung. Die Teilnehmer sind überaus computererfahren (8,4 Stunden Computernutzung pro Tag, $SA = 2,5$ Stunden) und können überwiegend als technik-affin bezeichnet werden.

Die Teilnehmer der Studie wurden so gewählt, dass sie nur bedingt Erfahrung mit digitalen Whiteboards und der Arbeit mit diesen haben, um eine möglichst spontane Erwartungshaltung gegenüber Undo, ohne Beeinflussung durch bereits bekannte Programmfeatures oder Ähnlichem, zu erhalten. So arbeiteten 87% der Teilnehmer noch nie oder beinahe nie mit einem digitalen Whiteboard. Keine der teilnehmenden Personen hat ein digitales Whiteboard jemals für Gruppenarbeit genutzt.

Drei der teilnehmenden Probanden haben sich bereits einmal mit der Implementierung oder Umsetzung eines Undo-Mechanismus in einer eigenen Single-User-Anwendung beschäftigt, 15 Teilnehmer beschäftigten sich noch nie mit Undo und fünf Teilnehmer gaben an, keine Softwareentwickler zu sein.

1.4.3 Design

Pro Studiendurchgang nahm jeweils ein Proband mit ein bis zwei Helfern an der Studie teil. Sie erstellten gemäß einer Aufgabenstellung gerichtete Graphen auf einem digitalen Whiteboard (siehe Abbildung 1.6). Dabei spielte jeder Proband insgesamt sechs verschiedene Konstellationen von Einzel- und Gruppenarbeit (Szenarien) durch. Nach jedem Szenario wurde der teilnehmende Proband vom Untersuchungsleiter befragt, wie sich Undo in Bezug auf die in diesem Szenario erstellten Inhalte verhalten sollte.

Das erwartete Verhalten eines jeden Probanden kann dann mit dem

[3] http://www.anoto.com/

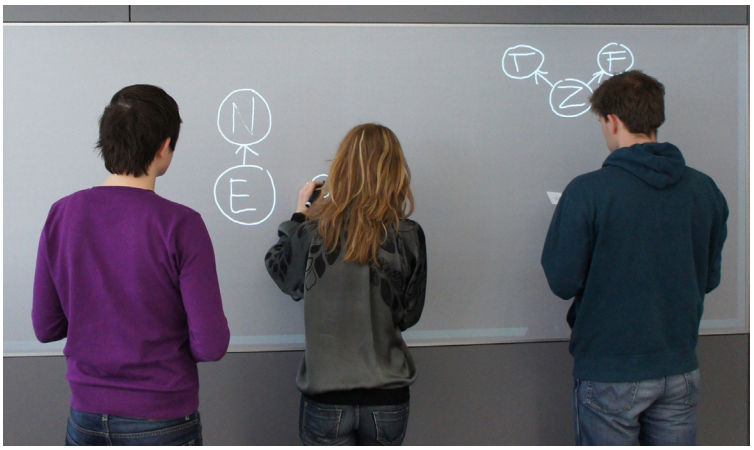


Abbildung 1.6: Der Proband (Mitte) erstellte in verschiedenen Einzel- oder Gruppenszenarien mit zwei Helfern der Studie gerichtete Graphen und wurde nach jedem Szenario vom Untersuchungsleiter zum erwarteten Verhalten von Undo befragt. Das erwartete Ergebnis der Probanden wurde dann mit dem Ergebnis von verschiedenen Undo-Techniken verglichen und daraus jene Technik ermittelt, die am ehesten den Erwartungen der Probanden entspricht.

Verhalten von verschiedenen Undo-Techniken verglichen werden und daraus Übereinstimmungen abgeleitet werden. Damit ergibt sich jene Undo-Technik, deren Ergebnis die meisten Übereinstimmungen mit dem von den Probanden erwarteten Ergebnis erzielt (bezogen auf jedes Szenario). *H1* wird dann bestätigt, wenn die Undo-Technik mit den meisten Übereinstimmungen in allen Szenarien einem regionalen Undo entspricht. Um die anderen Hypothesen zu bestätigen, füllten die Probanden nach jedem Szenario einen korrespondierenden Fragebogen aus.

Als Verfahren für die Studie wurde ein Laborexperiment mit jeweils einem Probanden ausgewählt, da durch das vollständige Durchspielen der Szenarien die natürlichsten Ergebnisse erwartet wurden. Eine Videobefragung oder Beobachtung durch die Probanden hätte von ihnen vorausgesetzt, dass sie sich in die Gruppen- und Einzelszenarien in einer ungewohnten Arbeitsumgebung (Multi-User-Whiteboard) hineinversetzen müssen.

Aufgabenstellung

Die verschiedenen Szenarien (insgesamt 6) wurden jeweils von einem Probanden und ein oder zwei Helfern der Studie durchgespielt. Dass es sich bei den Helfern um Mitarbeiter der Studie handelte, war für den Probanden nicht zu erkennen. Je nach Szenario und Aufgabenstellung mussten die Teilnehmer der Studie einen gerichteten Graphen erstellen. Ein gerichteter Graph bestand dabei aus Elementen (eingekreiste Buchstaben) sowie gerichteten

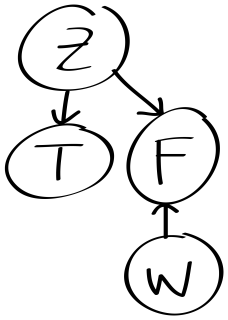


Abbildung 1.7: Die Aufgabe der Probanden war je nach Szenario die gemeinsame oder getrennte Erstellung von gerichteten Graphen. Ein gerichteter Graph bestand dabei aus Elementen (eingekreiste Buchstaben) sowie gerichteten Abhängigkeiten zwischen den Elementen (Pfeile).

Abhängigkeiten (Pfeile) zwischen den Elementen (siehe Abbildung 1.7). Innerhalb einer Gruppe wurde ein gemeinsamer Graph und von Einzelpersonen jeweils ein eigener Graph erstellt.

Dazu erhielt jeder Teilnehmer vor jedem Szenario eine persönliche Aufgabenstellung mit Teilaufgaben, die zur Erstellung der Aufgabe notwendig waren. Eine exemplarische Aufgabenstellung lautete beispielsweise:

```
- Beginnen Sie mit dem Element Z
- Zeichnen Sie Element T, Element Z ist von Element T abhängig
- Zeichnen Sie Element F, Element Z ist von Element F abhängig
- Zeichnen Sie Element W, Element W ist von Element F abhängig
```

Das Ergebnis dieser Aufgabenstellung ist in Abbildung 1.7 dargestellt. Die persönliche Aufgabenstellung von Gruppenmitgliedern enthielt jeweils nur einen Teil der Teilaufgaben, sodass die Gruppenmitglieder abwechselnd die Teilaufgaben lösen mussten, um zum richtigen (gemeinsamen) Ergebnis zu kommen.

Waren alle Aufgabenstellungen eines Szenarios erfüllt, begann der Untersuchungsleiter mit der Befragung des Probanden zum Verhalten von Undo. Der Proband sollte dabei möglichst spontan voraussagen, welche von den eben erstellten Inhalte beim *ersten* und *zweiten* Aufruf von Undo wieder verschwinden würden. Anschließend an die Befragung füllte der Benutzer einen Fragebogen zum jeweiligen Szenario aus.

Zusätzliche Maßnahmen

Um eine möglichst spontane Aussage über das Verhalten von Undo zu erhalten, wurde jeder Proband nach jedem Szenario direkt vor Ort durch den Untersuchungsleiter befragt. Deshalb nahm pro Szenario und Durchgang jeweils nur ein Proband teil, um eine mögliche gegenseitige Beeinflussung durch

die Befragung von mehreren, gleichzeitig teilnehmenden Personen zu verhindern. Die übrigen Teilnehmer in jedem Szenario wurden jeweils durch zwei Helfer der Studie gestellt, die nicht befragt wurden.

Die freie Platzwahl bei den Aufgabenstellungen wurde zur Steigerung der Kommunikation innerhalb der Gruppe eingeführt. Vor jedem Szenario wurde nur darauf hingewiesen, wer in Einzel- und wer in Gruppenarbeit arbeitet.

Um in der Studie trotz kurzer Aufgaben eine zusammengehörige Gruppenarbeit zu erreichen, wurde als Aufgabenstellung ein gerichteter Graph gewählt. Es wurde angenommen, dass dieser von den Gruppenmitgliedern durch die im Graphen enthaltenen Abhängigkeiten als stark zusammengehöriger Inhalt empfunden wird. Die Befürchtung war, dass die Probanden andere Inhalte, wie z. B. einfache Zeichnungen, nicht als gemeinsame Arbeit empfinden und somit die Ergebnisse in der Erwartungshaltung für Undo verfälscht würden. Zur Überprüfung, ob eine solche Aufgabenstellung tatsächlich andere Ergebnisse liefert, wurde in die Studiengestaltung ein wiederholtes Szenario mit einer einfachen Zeichnung (Haus und Baum) als Aufgabenstellung eingebaut (siehe Abbildung 1.8 (f)).

Um eine Beeinflussung der Probanden durch vordefinierte Regionen (z. B. Projektor-Kanten) auszuschließen, wurde als Hintergrund schwarz gewählt, während mit den digitalen Stiften weiß gezeichnet wurde (siehe Abbildung 1.6). Die Teilnehmer wurden außerdem darauf hingewiesen, dass es sich beim Whiteboard um eine durchgängige Arbeitsfläche handelt und es keine Einschränkung in der Platzwahl gibt. Dass diese Ansätze in der Studie gut funktioniert haben zeigen die Auswertungen der Interaktionsbereiche, die von den Untersuchungsteilnehmern in den verschiedenen Szenarien willkürlich gewählt wurden.

Um eine schlechte Vergleichbarkeit der Ergebnisse durch verschiedene Auffassungen der Probanden in Bezug auf die Granularität von Teilschritten auszuschließen, wurden die Benutzer explizit darauf hingewiesen, bei der Befragung zum Verhalten von Undo nicht einzelne Striche (z. B. Pfeilspitze) sondern immer einen Buchstaben (also einen gesamten Teilschritt der Aufgabenstellung) zu nennen. Nichtsdestoweniger stellt die Wahl der Granularität von Undo-Schritten ein Problem dar (siehe Abschnitt 3.4.2).

Des Weiteren wurde durch den unabhängig vom Probanden arbeitenden Helfer der Studie versucht, dass der Abstand zum Probanden möglichst gering ist. Damit sollte sichergestellt werden, dass sich der getrennt arbeitende Helfer der Studie im Sichtfeld des Probanden befindet und somit genaue Aussagen über die Wahrnehmung der Arbeit von anderen Personen erzielt werden können.

Ablauf

Zunächst wurde der Proband über die Formalitäten und den groben Ablauf der Studie aufgeklärt. Nach der Einführung füllte der Proband einen

Fragebogen mit allgemeinen Fragen zu Undo/Redo aus. Anschließend wurde ihm anhand eines einfachen Beispiels die kommenden Aufgabenstellungen erläutert. Der Proband wurde vor der Studie darauf hingewiesen, dass das Whiteboard eine durchgängige interaktive Fläche darstellt und die Platzwahl in jedem Szenario freigestellt ist. Außerdem wurde explizit auf das im oberen Abschnitt erläuterte Granularitätsproblem hingewiesen. Jedes Szenario wurde mit dem Ausgeben der Aufgabenstellungen und dem Hinweis, welche der Teilnehmer des jeweiligen Szenarios (Proband und Helfer) in Einzel- oder Gruppenarbeit arbeiten, eingeleitet. Daraufhin wählten die Teilnehmer die Plätze am Whiteboard und begannen mit der Erstellung der Inhalte gemäß den ausgegebenen Aufgabenstellungen. Waren alle Aufgabenstellungen von allen Teilnehmern erfüllt, begann der Untersuchungsleiter mit der Befragung des Probanden zum erwarteten Verhalten beim ersten sowie beim zweiten Undo-Aufruf. Anschließend füllte der Proband einen zum Szenario korrespondierenden Fragebogen aus. Die sechs Szenarien wurden jeweils in zufälliger Reihenfolge durchgespielt.

Szenarien

Jeder Proband absolvierte während der Studie insgesamt sechs verschiedene Szenarien. Jedes Szenario stellte eine bestimmte Konstellation von Gruppen- und Einzelarbeit (siehe Abbildung 1.8) dar:

Szenario 1 (Gruppenarbeit): Ein gemeinsamer Graph wurde durch den Probanden und einen Helfer der Studie erstellt.

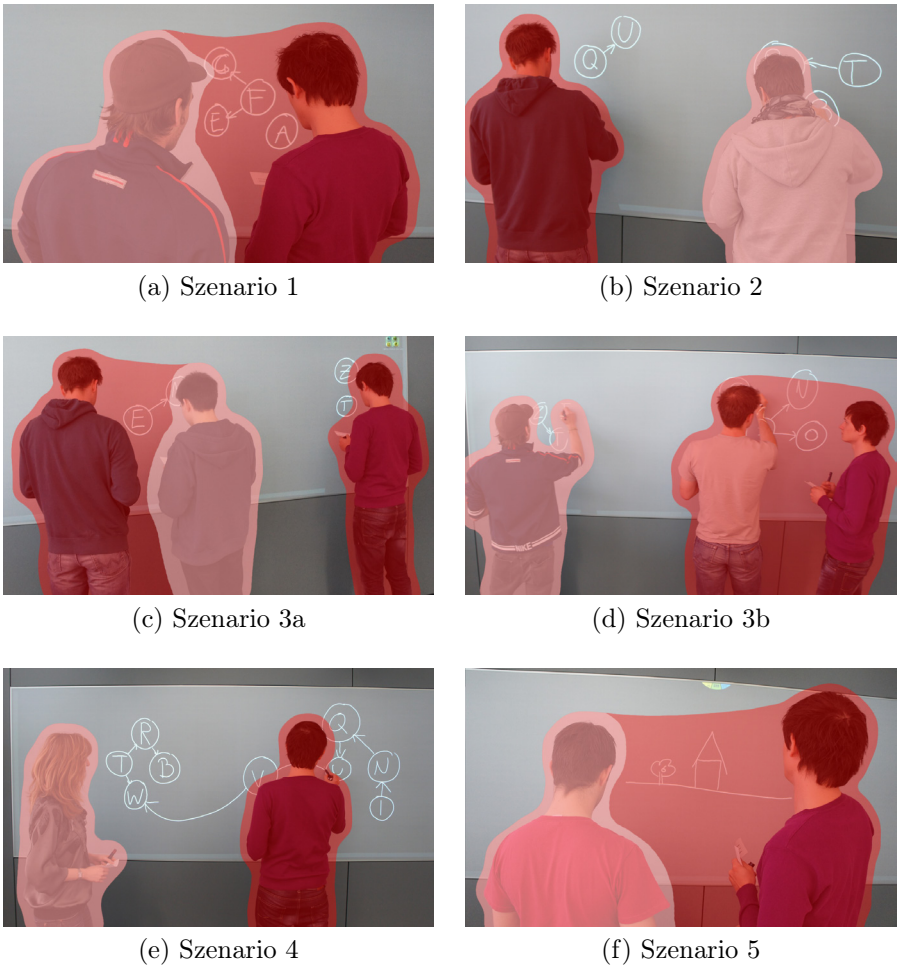
Szenario 2 (Einzelarbeit): Der Proband und ein Helfer der Studie erstellten jeweils unabhängig voneinander einen Graphen.

Szenario 3a (Gruppenarbeit und gleichzeitige Einzelarbeit): Ein Graph wurde gemeinsam von dem Probanden und einen Helfer der Studie erstellt, während ein weiterer Helfer unabhängig von der Gruppe einen eigenen Graphen erstellte.

Szenario 3b (Einzelarbeit und gleichzeitige Gruppenarbeit): Zwei Helfer der Studie erstellten jeweils einen gemeinsamen Graphen, während der Proband unabhängig von der Gruppe einen eigenen Graphen erstellte.

Szenario 4 (Einzelarbeit, die zur Gruppenarbeit wird): Zunächst erstellten der Proband und ein Helfer der Studie unabhängig voneinander einen Graphen, ehe die beiden Graphen durch ein neues Element des Helfers verbunden wurden.

Szenario 5 (Gruppenarbeit): Der Proband und ein Helfer der Studie erstellten in Gruppenarbeit eine einfache Zeichnung (Haus und Baum). Gegenszenario zu Szenario 1 (siehe Abschnitt 1.4.3).



(a) Szenario 1 (b) Szenario 2

(c) Szenario 3a (d) Szenario 3b

(e) Szenario 4 (f) Szenario 5

Abbildung 1.8: Folgende Zusammenstellungen mussten die Teilnehmer der Studie absolvieren. Gruppenarbeit (a), Einzelarbeit (b), Gruppenarbeit bei gleichzeitiger Einzelarbeit (c), Einzelarbeit bei gleichzeitiger Gruppenarbeit (d), Einzelarbeit, die mit Fortdauer zur Gruppenarbeit wird (e) sowie Gruppenarbeit mit einer einfachen Zeichnung als Aufgabenstellung (f). Die Zusammenstellungen der Teilnehmer ist dunkelrot und der jeweilige Proband zusätzlich hellrot markiert.

1.4.4 Ergebnisse

Allgemeine Angaben zur Nutzung von Undo/Redo

78% der Probanden erachten Undo als eine sehr wichtige und die restlichen 22% der Probanden für eine wichtige Funktion in einem Programm. 83% der Befragten nutzen Undo in ihrer täglichen Arbeit häufig, die restlichen 17% nutzen Undo mittelmäßig oft. 91% der Befragten benutzten bereits einmal einen erweiterten Verlauf mit eigener Visualisierung (z. B. Photoshop),

47% davon nutzen einen solchen erweiterten Verlauf häufig. 100% der Probanden nutzen Undo um Fehler zu korrigieren, 78% um zu einem früheren Dokumentstatus zu gelangen, 43% um neue Programmfeatures auszutesten und 30% auch noch für andere Gründe (u.a. für Try&Error, als eine Art Snapshot-Funktion oder um Effekte an Bildern zu testen). 96% der Befragten rufen die Funktion Undo über einen Shortcut auf (z. B. Strg+Z), 43% nutzen eine Verlauf-Visualisierung, 30% benutzt eine Toolbar und 22% der Befragten rufen Undo über das Programmmenü auf.

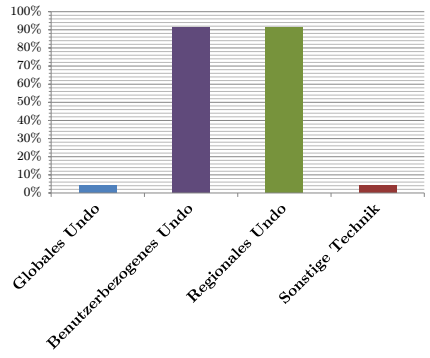
Szenarien

Die ermittelten Erwartungshaltungen der Probanden wurden in der Auswertung mit dem Verhalten von verschiedenen Undo-Techniken verglichen. Somit lassen sich für jedes Szenario und für jede Technik Übereinstimmungen ableiten, wie oft die Ergebnisse dieser Technik mit dem erwarteten Ergebnis des Probanden übereinstimmen. Verglichen wurde globales (dokumentenbasiertes) Undo, benutzerbezogenes Undo sowie regionales Undo (benutzer- und gruppenbezogen). Andere Undo-Techniken oder unbekannte Verhaltensweisen wurden als sonstige Technik zusammengefasst.

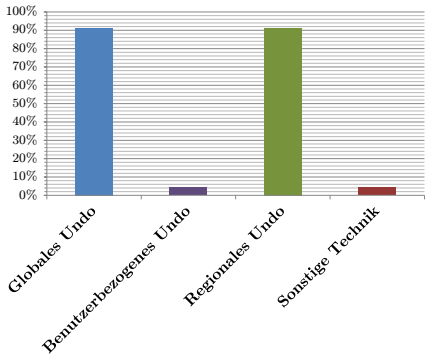
Gruppenarbeit In einem Gruppenszenario wie Szenario 1 erwarten sich 91% der Gruppe ein auf ihre Gruppe bezogenes Undo (siehe Abbildung 1.9 (a)). Ein Proband erwartete sich trotz Gruppenarbeit ein benutzerbezogenes Undo und eine Person erwartete eine sonstige Technik. Die Annahme, dass Probanden einfache Zeichnungen unter Umständen als weniger zusammengehörig empfinden als den gerichteten Graphen, bewahrheitete sich nicht. In Szenario 5, das im Wesentlichen die Wiederholung von Szenario 1 mit einer anderen Aufgabenstellung (Haus und Baum) war, erwarteten sich die Probanden sogar zu 100% ein auf ihre Gruppe bezogenes Undo (siehe Abbildung 1.9 (f)). Auch die Auswertung der Zusammengehörigkeit der erstellten Inhalte zeigte, dass die einfache Zeichnung mit Haus und Baum sogar tendenziell als zusammengehöriger empfunden wurden als der gerichtete Graph.

Gruppen- und Einzelarbeit Die Erwartung eines auf die Gruppe bezogenen Undos bleibt innerhalb einer Gruppe auch dann weitestgehend bestehen (83%), wenn wie in Szenario 3a unabhängig von der Gruppe auch noch eine weitere Person am Whiteboard arbeitet (siehe Abbildung 1.9 (c)). Dieses Verhalten kann aufgrund der gleichzeitigen Arbeit von Gruppe und Einzelperson nur durch ein regionales Undo gewährleistet werden. Eine Person erwartete sich in Szenario 3a trotz Gruppenarbeit ein benutzerbezogenes Undo und einige Personen (13%) eine sonstige Technik.

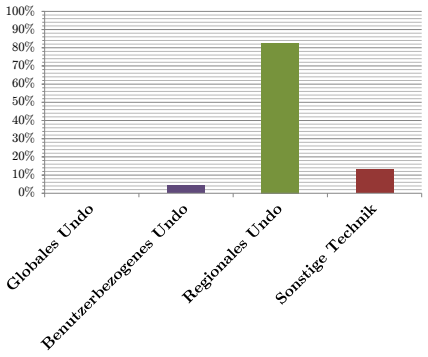
Einzelarbeit Wie in Abbildung 1.9 (b) ersichtlich, erwarteten sich die Probanden bei zwei unabhängig arbeitenden Einzelpersonen (Szenario 2) zum



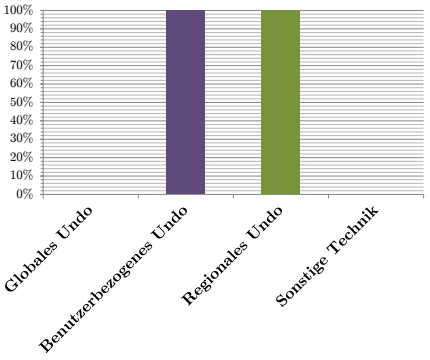
(a) Szenario 1



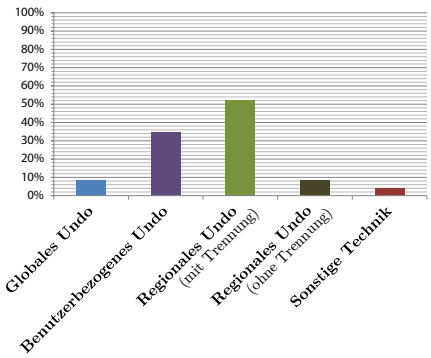
(b) Szenario 2



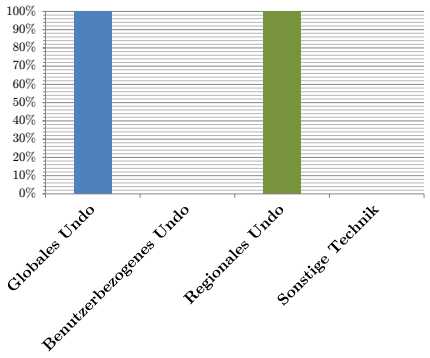
(c) Szenario 3a



(d) Szenario 3b



(e) Szenario 4

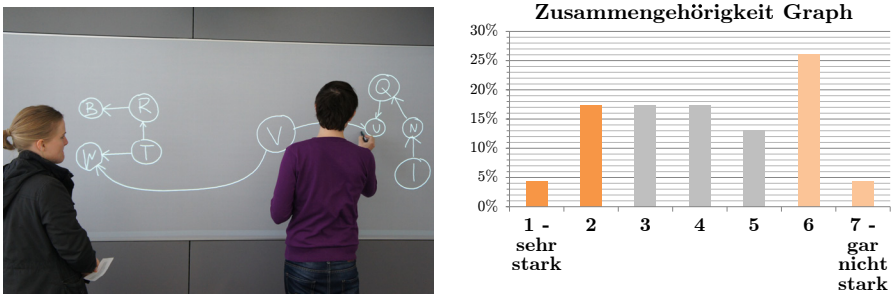


(f) Szenario 5

Abbildung 1.9: Die Diagramme zeigen, wie oft eine bestimmte Undo-Technik in einem Szenario genau jenes Ergebnis liefert bzw. jenem Verhalten entspricht, das von den Probanden erwartet wurde. Allgemein lässt sich ableiten, dass Gruppen ein auf ihr Gruppe bezogenes Undo erwarten, während Einzelpersonen ein benutzerbezogenes Undo erwarten.

überwiegenden Teil (91%) ein benutzerbezogenes Undo. Nur ein Proband erwartete sich in diesem Fall ein globales Undo und eine Person eine sonstige Technik. Die 91% Zustimmung für benutzerbezogenes Undo steigen sogar auf 100%, wenn eine Einzelperson wie in Szenario 3b unabhängig von einer ebenfalls am Whiteboard arbeitenden Gruppe alleine arbeitet (siehe Abbildung 1.9 (d)). Dies kann darauf zurückgeführt werden, dass ein Proband in einem solchen Fall noch stärker das Gefühl hat, dass es sich um „seinen“ und „deren“ (jenem der Gruppe) Inhalt handelt.

Vermischung Einzel- und Gruppenarbeit Interessante Ergebnisse lieferte auch Szenario 4, bei dem zwei getrennt erstellte Graphen nach Abschluss durch ein neues Element des Helfers verbunden wurden (siehe Abbildung 1.10 (a)). Die Auswertung der in diesem Fall anzuwendenden Undo-Technik zeigte Uneinigkeit bei den Probanden. 35% der Probanden erwarteten in diesem Szenario ein benutzerbezogenes Undo, 52% ein regionales Undo mit Auftrennung der Graphen (also im ersten Schritt ein gruppenbezogenes, im zweiten Schritt ein benutzerbezogenes Undo) und jeweils 9% der Probanden ein globales und gänzlich gruppenbasiertes Undo (siehe Abbildung 1.9 (e)). Diese Uneinigkeit hängt auch mit dem Umstand zusammen, dass die Meinungen der Probanden, ob es sich bei dem verbundenen Graphen um einen zusammengehörigen Inhalt handelt, weit auseinander gingen. Rund 40% der Probanden empfanden den verbundenen Graphen als zusammengehörig, während 43% der Probanden den Graphen als wenig zusammengehörig bewerteten (siehe Abbildung 1.10 (b)). In einigen Kommentaren gaben Probanden an, dass der Mitarbeiter der Studie den Probanden nicht gefragt hat, ob er die beiden Graphen verbinden dürfte. Dies deutet auf das Vorherrschen von sozialen Protokollen und Regeln hin, an die sich die auf der Interaktionsfläche arbeitenden Personen vor allem in Bezug auf Gruppenarbeit halten müssen.



(a) (b)

Abbildung 1.10: Die Probanden waren sich uneinig, ob es sich bei dem durch den Helfer verbundenen Graphen (a) in Szenario 4 um einen zusammengehörigen Inhalt handelt (b).

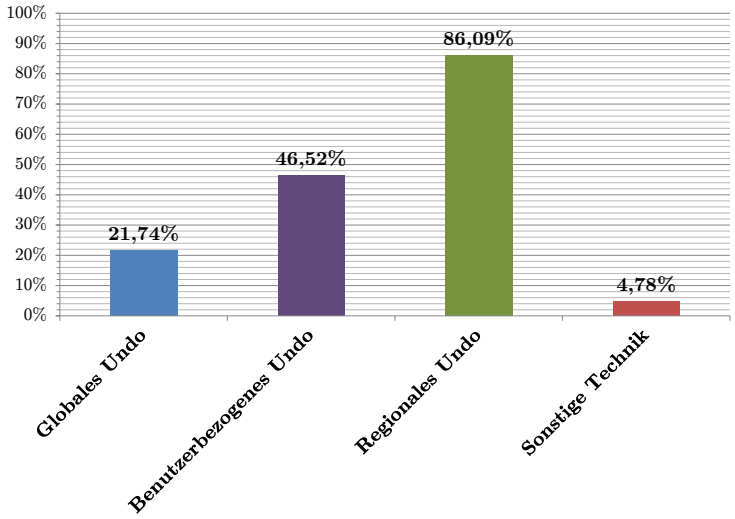


Abbildung 1.11: Über alle Szenarien gesehen liefert regionales Undo am häufigsten jenes Ergebnis, das der Erwartung der Probanden entspricht.

Szenarien gesamt Über alle Szenarien gesehen entspricht regionales Undo am häufigsten den Erwartungen der Benutzer. Dabei würde regionales Undo in 86% aller Fälle jenes Ergebnis liefern, welches auch von den Probanden in der Studie als am intuitivsten wahrgenommen wurde. Benutzerbezogenes Undo würde in 46% der Fälle das entsprechende Ergebnis liefern, globales Undo in 21% der Fälle (siehe Abbildung 1.11). Das duplizierte Gruppenszenario 5 mit der einfachen Zeichnung als Aufgabenstellung floss nicht in die Auswertung ein, damit reiner Gruppenarbeit kein zu starkes Gewicht zukommt. Da Undo in erster Linie der Intuition des Benutzers entsprechen soll und regionales Undo bei verschiedenen Konstellationen von Gruppen- und Einzelarbeit am häufigsten den Erwartungen der Probanden entsprach, wird Hypothese *H1* damit bestätigt.

Korrelation Benutzerposition und Undo-Aufruf

Eine weitere Hypothese (*H2*) der Studie war, dass ein Benutzer jeweils an jenem Ort eine Undo-Operation aufrufen möchte, an dem er gerade aktiv arbeitet. Diese Hypothese wird vor allem durch die Ergebnisse aus Abbildung 1.12 gestützt. Der überwiegende Teil der Probanden erwartet zunächst nicht, dass sie von ihrer aktuellen Position aus Arbeitsschritte einer anderen Person rückgängig machen könnten. Erst wenn der Proband sich zu der Person begeben würde, ändert sich auch die Erwartungshaltung. Bei reiner Beobachtung der Person durch den Probanden ist die Erwartungshaltung noch relativ gespalten, während Mitdiskutieren oder das aktive Zusammenarbeiten mit der anderen Person dazu führt, dass sich die Erwartungshaltung dahingehend

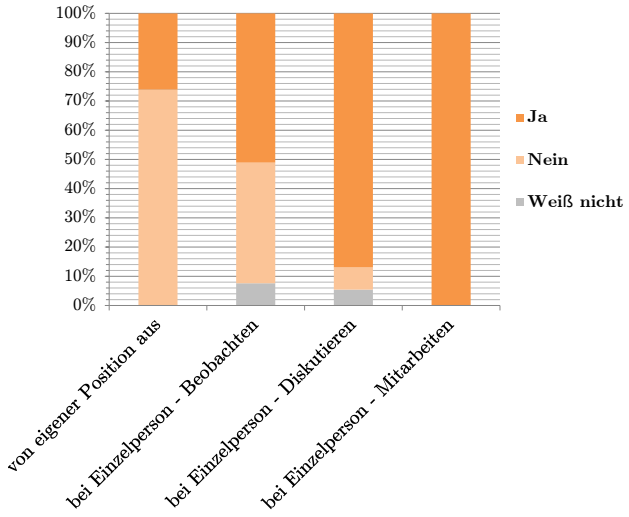


Abbildung 1.12: Dieses Diagramm zeigt die Erwartungshaltung der Probanden, ob diese auch Schritte von einer anderen Einzelperson/Gruppe rückgängig machen können - von der eigenen Position aus (a), wenn sich der Proband zur Person begibt und die Person beobachtet (b), mit ihr über dessen Arbeit diskutiert (c) oder aktiv mitarbeitet (d).

verschiebt, dass der Proband auch Arbeitsschritte der anderen Person rückgängig machen kann. Daraus lässt sich ableiten, dass ein Benutzer nicht von einer Position aus Arbeitsschritte an einer anderen (weiter entfernten) Position rückgängig machen möchte. Außerdem verdeutlichen auch diese Ergebnisse wieder ein Bestehen von unterbewussten, sozialen Protokollen. So ist offensichtlich das Beobachten der Arbeit einer anderen Person nicht ausreichend, um tatsächlich Zugriff oder Kontrolle über die Arbeit dieser anderen Person zu erlangen. Erst durch das Mitdiskutieren bzw. Mitarbeiten erlangt der Benutzer einen entsprechenden Zugriff.

Wahrnehmung

Des Weiteren wurde angenommen, dass Personen, die unabhängig von anderen Personen auf der gleichen interaktiven Fläche arbeiten, dessen Arbeit nicht aktiv wahrnehmen (*H3*). Dazu wurden die Probanden in den Szenario-Fragebögen befragt, ob sie die andere Person/Gruppe wahrnahmen und wie sehr sie diese in der eigenen Arbeit gestört haben. Dabei empfand der überwiegende Teil der Probanden, dass die Arbeit der anderen Person/Gruppe nicht sonderlich wahrgenommen wurde. Der Großteil von ihnen empfand die Arbeit anderer Personen als gar nicht störend. In den abschließenden Befragungen der Studie stellte sich heraus, dass die Probanden großteils bemerkten, dass auf der interaktiven Fläche noch jemand arbeitet. Sie wussten

jedoch in den meisten Fällen nicht, an welchem Inhalt die entsprechende Person genau arbeitet - sofern diese Arbeit nichts mit ihrer eigenen Aufgabenstellung zu tun hatte.

Mit der Wahrnehmung von Personen in verteilten Echtzeit-Arbeitsprozessen beschäftigten sich auch Gutwin und Greenberg [9]. Sie stellten in ihrer Arbeit ein beschreibendes Framework auf, das den Entwicklern von verteilten Anwendungen Regeln und Leitsätze zur Verfügung stellt, um verteilte Arbeitsprozesse und die Wahrnehmung dieser besser bewerten zu können. In ihrer Arbeit kamen sie zur Auffassung, dass eine vollständige *Situation Awareness* in einem verteilten Arbeitsprozess unter anderem von folgenden Elementen abhängt:

1. **Wer:** Gegenwart, Identität, Autorenschaft
2. **Wo:** Ort, Blickfeld, Erreichbarkeit
3. **Was:** Arbeit, Ziel, Produkt

Die Ergebnisse der Studie bezüglich Wahrnehmung wiedersprechen der Auffassung von Gutwin und Greenberg, dass in Face-to-Face-Umgebungen Awareness-Probleme ausgeschlossen sind. Eine vollständige Wahrnehmung des Arbeitsbereiches kann in den Szenarien der Studie schon deshalb nicht gegeben sein, weil getrennt und unabhängig voneinander arbeitende Benutzer die Arbeit der jeweils anderen Benutzer nicht aktiv wahrnehmen und während des Arbeitsprozesses auch nicht an der Zielsetzung oder der Intention der anderen Benutzer interessiert sind.

1.4.5 Diskussion

Die Studie ergab, dass regionales Undo am häufigsten jene Ergebnisse liefert, die die Probanden in den verschiedenen Szenarien erwartet haben (*H1*). Einzelpersonen erwarten ein benutzerbezogenes Undo, Gruppen ein auf ihre Gruppe bezogenes Undo. Dabei ist es völlig unerheblich, ob noch weitere Personen im näheren Umkreis auf der interaktiven Fläche arbeiten oder nicht. Zweifel treten nur dann auf, wenn sich aus Einzelarbeit eine zusammenhängende Gruppenarbeit entwickelt. Dieser Umstand hängt wohl damit zusammen, dass in solchen speziellen Szenarien für die Probanden konkrete Erfahrungswerte fehlen. Entscheidend ist in einem solchen Fall, wie zusammengehörig die Benutzer einen Inhalt empfinden. Desto zusammengehöriger das Empfinden, desto eher die Bevorzugung von regionalem Undo.

Außerdem erwarten Benutzer auf einer großen interaktiven Fläche, dass Undo immer dort ausgeführt werden sollte, wo sie gerade arbeiten bzw. wo gerade ihr Fokus liegt (*H2*). Benutzer erwarten nicht, dass sie von der eigenen Position aus Arbeitsschritte eines anderen Benutzers an einer anderen Position rückgängig machen können. Außerdem zeigt die Studie auf, dass es sehr wohl vorkommen kann, dass auf derselben interaktiven Fläche arbeitende Benutzer einander nicht aktiv wahrnehmen (*H3*). Erst mit Abschluss der

eigenen Arbeit bewegt sich der Fokus weg von der eigenen Arbeit hin zu der Arbeit von anderen Personen.

Obwohl in der Studie eine klare Tendenz hin zu regionalem Undo ermittelt werden konnte, verfügt die Studie auch über einige Einschränkungen. Eine Einschränkung der Studie ist, dass die Gruppenprozesse in den Szenarien weitestgehend konstruiert wurden und dass in einer echten kollaborativen Gruppenarbeit möglicherweise andere Regeln vorherrschen, die mit den relativ einfachen und kurzen Aufgabenstellungen der Studie nicht abgebildet werden konnten.

Darüber hinaus können in realen Arbeitsprozessen noch andere Szenarien oder Sonderfälle (wie beispielsweise Szenario 4) auftreten, die in der Studie nicht berücksichtigt wurden. Vor allem in Bezug darauf, wie sich Einzel- und Gruppenarbeit über einen längeren Zeitraum entwickelt und dass möglicherweise Zusammenhänge, Synthesen oder Aufspaltungen entstehen können. Das Problem hierbei ist, dass es in solchen speziellen Szenarien oder Sonderfällen aufgrund der fehlenden Erfahrungen der Benutzer mit solchen Arbeitsumgebungen keine eindeutige Erwartungshaltung mehr gibt. Damit kann auch keine Lösung konstruiert werden, die *immer* der Intuition des Benutzers entspricht. Einziges Ziel kann sein, ein Verfahren zu entwickeln, das *möglichst oft* der Erwartung der Benutzer entspricht.

Eine weitere Einschränkung ist, dass für die Studie bewusst nur Einfügeoperationen rückgängig gemacht wurden, um den Probanden das Voraussagen des erwarteten Undo-Verhaltens zu erleichtern. In der Praxis kommen natürlich eine Vielzahl von Operationen (z. B. Löschoperationen, Transformationen, etc.) vor, von denen aber angenommen wird, dass sie die Erwartung gegenüber dem Verhalten von Undo nicht wesentlich beeinflussen.

1.5 Ziel dieser Arbeit

In diesem Kapitel wurde aufgezeigt, dass sowohl benutzerbezogenes als auch globales Undo auf großen interaktiven Flächen nicht immer der Intention des Benutzers entspricht. Regionales Undo entspricht dabei noch am häufigsten der Erwartungshaltung der Benutzer. Generell kann abgeleitet werden, dass sich Gruppen ein gruppenbezogenes und Einzelpersonen ein benutzerbezogenes Undo erwarten.

Das Ziel dieser Arbeit ist folglich die Umsetzung und Konzeption von regionalem Undo für Anwendungen auf großen interaktiven Flächen, die das Bearbeiten bzw. Überzeichnen von großflächigen Dokumenten wie Plänen, Diagrammen oder Karten in verschiedenen Konstellationen von Gruppen- und Einzelarbeit erlaubt.

Kapitel 2

Verwandte Arbeiten

Undo mit all seinen Möglichkeiten ist ein viel erforschtes Thema im Bereich Mensch-Maschine-Interaktion (HCI) [1,6,11,18,20]. Grundsätzlich lässt sich die Forschung im Bereich Undo und History in zwei große Gebiete teilen. Einerseits kann in die technische Beschreibung von Undo-Modellen [2, 7, 18, 21, 24, 29, 30] und andererseits in die Visualisierung von Verläufen und verschiedenen Dokumentzuständen über die Zeit [8, 12, 13, 15, 16] unterteilt werden. Bei der technischen Beschreibung wird wiederum zwischen *Linearem* und für diese Arbeit relevanten *Non-Linearem* (Selektiven) Undo unterschieden.

2.1 Non-Lineares Single-User-Undo

Im Bereich von Non-Linearem Undo wurde viel Arbeit in die Beschreibung von formalen Modellen investiert. Modelle für Non-Lineares Single-User-Undo sind in den Arbeiten von Archer et al. [2], Vitter [29] oder Yang [30] zu finden. Das einflussreichste der genannten Modelle ist das Skript-Modell von Archer et al. [2]. In diesem Modell werden alle Aktionen eines Benutzers als ein Skript von Kommandos interpretiert. Das Modell erreicht, ausgehend von einem Anfangszustand, durch Ausführen aller Kommandos im Skript einen bestimmten Endzustand. Durch Änderung von bereits ausgeführten Kommandos und der kompletten Neuausführung des Skripts (*Re-Run-Strategy*) kann Undo und Redo bereitgestellt werden. Das Skript-Modell kann als Basis vieler Undo-Modelle im Single- und Multi-User-Bereich angesehen werden, die Non-Lineares bzw. Selektives Undo ermöglichen. So kann auch diese Arbeit als Erweiterung eines Skript-Modells gesehen werden.

2.2 Non-Lineares Multi-User-Undo

Abowd und Dix beschäftigten sich in ihrer Arbeit mit der formalen Beschreibung eines Multi-User-Undo-Modells. Sie betrachteten Undo vor allem aus

Sicht der Benutzer und stellten fest, dass Undo grundsätzlich immer der Intention von Benutzern entsprechen sollte. Darüber hinaus erläuterten sie die Anforderungen, die in Multi-User-Systemen an Undo und Redo gestellt werden. So zum Beispiel zeigten sie auf, dass die Wahl der Granularität von Aktionen in Bezug auf Undo ein wesentliches Problem darstellt. Darüber hinaus stellten sie neben dem formalen Modell auch einige Methoden wie „Locking“ oder „Rollen“ vor, die die Probleme von Undo in Zusammenhang mit Multi-User-Systemen beseitigen können [1]. Die Erkenntnisse aus der Arbeit von Abowd und Dix, vor allem jene in Bezug auf Granularität, flossen in die Entwicklung der Konzepte dieser Arbeit ein.

Hazemi und Macaulay beschäftigten sich in einer Studie mit den Erwartungen von Benutzern in verteilten Multi-User-Anwendungen. Daraus leiteten sie Anforderungen an die Undo-Funktionalität in verteilten Anwendungen ab und wie diese mit bekannten Methoden aus dem Bereich Computer Supported Cooperative Work (CSCW) gelöst werden können [11]. Diese Studie war Inspiration und Ausgangslage für die in dieser Arbeit durchgeführte Studie zur Erwartungshaltung von Undo der Benutzer in einem echten Multi-User-Szenario (siehe Abschnitt 1.4).

Selektives Undo

Entgegen dem eigentlichen Skript-Modell unterstützt die Arbeit von Cass und Fernandes semantische Abhängigkeiten von Operationen, deren Abhängigkeiten in sogenannten Task Models (Bäume) modelliert werden. Das Verfahren kann somit ermitteln, ob beim Rückgängigmachen einer bestimmten Operation eventuell auch semantisch abhängige Operationen rückgängig gemacht werden müssen [5]. Eine empirische Evaluierung verschiedenster Undo-Techniken zeigte, dass Benutzer am ehesten ein abhängiges selektives Undo, wie von Cass und Fernandes beschrieben, erwarten [6]. Die Erkenntnisse dieser Studie flossen auch in diese Arbeit ein, die mögliche semantische Abhängigkeiten von Operationen automatisch rückgängig macht (siehe Abschnitt 4.4.5).

Berlage präsentiert in seiner Arbeit eine Möglichkeit von selektivem Undo für grafische Editoren basierend auf History-Bäumen [3]. Dabei wird mit Bäumen und Verzweigungen gearbeitet. Bei jedem Undo und einem daraus resultierenden neuen Dokumentzustand wird eine neue Verzweigung im Baum begonnen, während nachfolgende Operationen einfach in die neue Verzweigung übertragen werden. So können im Dokument durch Aufbauen und Navigieren von verschiedenen Verzweigungen des Baumes selektiv Änderungen vorgenommen werden. In [4] zeigte Berlage einen auf dieser Arbeit basierenden Ansatz für Multi-User-Anwendungen.

Generell wurde die Implementierung von Multi-User-Undo hauptsächlich aus Sicht von verteilten Texteditoren betrachtet. Die wichtigsten Arbeiten in diesem Bereich sind *DistEdit* [18], *adOPTed* [20, 21] und *AnyUndo* [24] bzw.

COT [27]. Alle diese Arbeiten verwenden Transformationen von Operationen als Basis (Operational Transformation).

Die erste publizierte Lösung für Multi-User-Undo in verteilten Texteditoren war *DistEdit* [18], welches das Rückgängigmachen von Undo-Operationen in beliebiger Reihenfolge (Selektives Undo) ermöglichte. Jedoch kann eine Operation möglicherweise nicht rückgängig gemacht oder wiederhergestellt werden, wenn sie in Konflikt mit einer anderen ausgeführten Operation steht. Konflikte entstehen beispielsweise, wenn zwei Operationen auf den gleichen oder benachbarten Inhalt operieren.

adOPTed [20,21] benutzt dynamische Transformationen um Undo in kollaborativen Systemen zu ermöglichen. Dabei werden neben den eigentlichen auch künftig mögliche Transformationen in multi-dimensionalen Transformationsmatrizen in einer vorausschauenden Art und Weise abgelegt. Der Nachteil dieser Lösung ist, dass nur eine chronologische und keine selektive Undo-Abfolge möglich ist.

AnyUndo [24] stellt einen weiteren Ansatz für verteilte Texteditoren dar, der auf Operational Transformation basiert. *AnyUndo* ermöglicht ein selektives und konfliktfreies Undo und Redo von beliebigen Operationen zu jedem Zeitpunkt. Sun präsentierte darüber hinaus eine Abwandlung dieser Arbeit, die in objekt-basierten grafischen Editoren eingesetzt werden kann [25]. *COT* [27] stellt eine Weiterentwicklung von *AnyUndo* dar, die jede Operation um einen Kontext erweitert, der den jeweiligen Zustand des Dokuments zum Zeitpunkt der Ausführung der Operation enthält. Damit lassen sich Spezialfälle im Zusammenhang mit der kausalen Abfolge von Operationen einfacher behandeln und zugleich eine Effizienzsteigerung erreicht werden.

Als technische Grundlage dieser Arbeit wurde *AnyUndo* [24] ausgewählt. Einerseits wegen der Möglichkeit von konfliktfreien selektiven Undo und der einfachen Abbildung von Operationen in einem linearen Verlauf. Andererseits wegen einer möglichen Erweiterung der Konzepte dieser Arbeit auf verteilte Anwendungen.

2.3 Regionales Undo

Mit der Möglichkeit eines regionalen Undos beschäftigten sich noch relativ wenige Arbeiten. Prakash erwähnte, dass regionales Undo mit der von ihm beschriebenen Arbeit möglich ist. Jedoch stellte er fest, dass sich gerade in Texteditoren die Frage stellt, was Regionen sind und wie diese bestimmt werden [18]. Die Fragestellung, wie Regionen in einem grafischen Editor und bei mehreren gleichzeitig arbeitenden Benutzern bestimmt werden können, stellt auch einen zentralen und wesentlichen Punkt dieser Arbeit dar.

Die Arbeit von Li et al. zeigt ein Verfahren für Texteditoren, wie Undo basierend auf frei wählbaren Regionen oder aber dem gesamten Inhalt eines Textdokuments realisiert werden kann. Dabei wird die Schwierigkeit

der Wahl der einzelnen Operationen bei einem selektiven Undo auf die Wahl einer ungefähren Region des Dokuments verlagert. Für das selektive Undo werden dann alle darin enthaltenen Operationen herangezogen. Mit einem eigenen Verfahren namens „Region Transformation“ werden die Regionen aufgrund von Operationen mitangepasst [14]. Der Grundgedanke Operationen aufgrund von Regionen im Dokument auszuwählen, verhält sich ähnlich wie die regionalen Undo-Konzepte dieser Arbeit.

Sowohl Meng et al. [15] als auch Nakamura und Igarashi [17] präsentierten Konzepte, in denen der visualisierte Verlauf von Operationen aufgrund spezifischer Regionen im Dokument gefiltert werden können. Entgegen der Konzepte dieser Arbeit, wonach die Auswahl der Operationen für Undo und Redo direkt im Dokument erfolgen kann, ist bei den beiden genannten Arbeiten eine zusätzliche Visualisierung des Verlaufs notwendig.

2.4 Visualisierungen

Einige Arbeiten beschäftigen sich, wie das in Abschnitt 5.1.2 beschriebene „History Wheel“, mit der Darstellung früherer Dokumentzustände bzw. dem sogenannten „History Browsing“.

Kurlander und Feiner präsentierten in ihrer Arbeit grafische editierbare Verläufe. Der Verlauf besteht dabei aus einer Auflistung von Miniaturbildern, die wichtige Ereignisse in der Evolution eines Dokumentes zeigen [12, 13]. Auch die Arbeit von Meng et al. zeigt in visueller Form den Verlauf eines Dokumentes bzw. Teil eines Dokumentes [15]. Das Projekt Flatland ermöglicht das Anbinden eines Zeit-Sliders an beliebige Objekte an einem digitalen Whiteboard, mit Hilfe dessen ein Abspielen der Evolution eines Objektes möglich ist [16]. Das Projekt Chronicle von Grossman et al. ermöglicht ein System zur Exploration und Wiedergabe eines Dokument-Workflows. Dazu stellt das System eine Form von Miniaturbildern sowie eine umfangreiche Timeline zur Verfügung. Dabei können neben dem gesamten Dokument auch nur Teilbereiche betrachtet werden [8].

Einen ähnlichen Ansatz wie die eben beschriebenen Arbeiten verfolgt das in dieser Arbeit vorgestellte „History-Wheel“, das die visuelle und stufenlose Exploration der Entwicklung eines Bereiches direkt im Dokument erlaubt.

2.5 Rahmenbedingungen

Generell ist das im letzten Abschnitt beschriebene Projekt Flatland und seine Funktionalitäten der Ausgangslage bzw. dem Szenario dieser Arbeit sehr ähnlich. Im Unterschied zu dieser Arbeit handelt es sich bei Flatland jedoch um eine Single-User-Anwendung [16]. Einen Überblick über das in dieser Arbeit verwendete Setup findet sich in [10].

Kapitel 3

Regionale Undo-Techniken

Da regionales Undo in aktuellen Anwendungen wenig verbreitet ist und ein Multi-User-Szenario und große interaktive Flächen spezielle Anforderungen an die Einbindung von regionalem Undo mit sich bringen, müssen auch entsprechende Konzepte entwickelt werden, die genau auf diese Anforderungen eingehen. Nachfolgend werden die Designziele bei einem Konzept für regionales Undo sowie die verwendeten Rahmenbedingungen erläutert und im weiteren Verlauf entsprechende Konzepte vorgestellt.

3.1 Designziele

Ausgehend von den von Cloudhary und Dewan festgelegten Anforderungen an ein Multi-User-Undo-Modell [7] und den aus der Studie gewonnenen Erkenntnissen (Regionales Undo, fehlende Awareness, usw.) ergeben sich für Entwicklung regionaler Undo-Konzepte folgende Anforderungen:

Kompatibilität: Ein Multi-User-Undo-Konzept sollte sich, wenn möglich, immer wie ein Single-User-Undo bei einem Benutzer verhalten. Gerade im Fehlerfall ist der Benutzer unter Umständen gestresst und nicht bereit, neue Verhaltens- und Funktionsweisen zu erlernen.

Unmittelbarkeit: Ein entsprechendes Konzept soll für die Benutzer einer Multi-User-Anwendung möglichst wenig Mehraufwand in der Handhabung gegenüber herkömmlichen Single-User-Undo-Lösungen für einen Benutzer bedeuten. Von Fehlerfällen gestresste Benutzer wollen den ursprünglichen Zustand in erster Linie schnell wiederherstellen.

Reproduzierbarkeit: Die gleichen Schritte, die mit Undo rückgängig gemacht werden, müssen mit Redo auch wiederhergestellt werden können.

Unabhängigkeit und Regionalität: Andere Nutzer, die in anderen Bereichen arbeiten, sollten durch Undo-Aufrufe oder Tätigkeiten eines anderen Benutzers nicht gestört werden.

Kollaboration: Benutzer sollen auch Operationen von anderen Benutzern rückgängig machen können, sofern ihr Fokus auf diesen Operationen liegt. Das bedeutet, dass Benutzer einer Arbeitsgruppe alle Operationen der Gruppe rückgängig machen und wiederherstellen können. Dies soll auch gelten, wenn Benutzer ihren Fokus von einer eigenen Arbeit auf eine andere Arbeit verlagern.

3.2 Abgrenzung

Die Auffassung des Begriffes „Region“ hängt üblicherweise stark vom Typ der Anwendung ab. So wird bei einem grafischen Editor unter Region üblicherweise ein Teil der Zeichenfläche verstanden, während bei einem Texteditor als Region eher ein Teil eines Paragraphen im Text bezeichnet wird. Ein allgemein gültiges Konzept für verschiedenste Typen von Anwendungen ist deshalb schwierig und nicht ohne Einschränkungen möglich. Da jedoch digitale Whiteboards und andere große interaktive Flächen aufgrund ihrer Größe öfter für Skizzen und Handzeichnungen als für Textbearbeitung genutzt werden, ist in diesem Kontext eine Fokussierung auf grafische Editoren sinnvoll (siehe Ausgangslage in Abschnitt 1.2).

Die nachfolgenden Konzepte und Umsetzungen beziehen sich daher auf einen grafischen Editor, der speziell für den Einsatz auf großen interaktiven Whiteboards entwickelt wurde und das Zeichnen bzw. Überzeichnen von großen Dokumenten wie Plänen, Diagrammen oder Karten durch mehrere Benutzer ermöglicht (siehe Abbildung 3.1). Beliebig viele Benutzer können dabei gleichzeitig unter Verwendung von digitalen Stiften mit der Anwen-

Abbildung 3.1: Die Konzepte dieser Arbeit beziehen sich auf einen grafischen Editor, der auf einem interaktiven Whiteboard das Zeichnen bzw. Überzeichnen von großflächigen Dokumenten durch mehrere Benutzer erlaubt. Jeder Benutzer interagiert über einen digitalen Stift und verfügt über ein benutzerbezogenes Menü, welches üblicherweise in seiner näheren Arbeitsumgebung geöffnet ist.

dung interagieren. Jeder Benutzer verfügt über ein eigenes stiftbezogenes Benutzermenü, welches an jeder Position des Whiteboards geöffnet werden kann und in der Regel in der näheren Arbeitsumgebung des Benutzers geöffnet ist.

3.3 Techniken für Regionales Undo

Die Umsetzung bzw. Planung eines Konzeptes für regionales Undo beschäftigt sich zu großen Teilen einerseits mit der *Bestimmung der Regionen* und andererseits dem *visuellen Feedback* über die gebildeten Regionen. Die zentrale Fragestellung bei der Bestimmung der Regionen ist, wie die verschiedenen Arbeitsbereiche von gleichzeitig arbeitenden Gruppen und Einzelpersonen erkannt werden können. Dabei gibt es verschiedene Möglichkeiten, wie Regionen für Undo und Redo gebildet werden können:

Implizite Regionsbestimmung: Verwendung von vordefinierten Regionen für Undo/Redo (durch physikalische Gegebenheiten oder softwareseitige Abtrennungen, z. B. Seiten).

Explizite oder manuelle Regionsbestimmung: Die Benutzer definieren mithilfe von entsprechenden Werkzeugen selbst Regionen für Undo und Redo.

Automatische Regionsbestimmung: Die Regionen werden durch verschiedene Verfahren automatisch ermittelt. Dabei können folgende Arten unterschieden werden:

Inhaltsbezogen: Der Inhalt eines Dokumentes bestimmt verschiedene Regionen, von denen pro Benutzer jeweils eine für Undo/Redo ausgewählt wird.

Positionsbezogen: Die Position bzw. der Arbeitsbereich der Benutzer bestimmt die Region für Undo und Redo.

Blickfeldbezogen: Das Blickfeld des Benutzers definiert die Region für Undo/Redo.

Nach der Bildung der Regionen stellt sich die Frage, wie diese den Benutzern in einer möglichst subtilen Art und Weise angezeigt werden können. Jeder Benutzer sollte beim Aufruf von Undo zumindest wissen, welche Region betroffen und welcher der als nächstes ausgeführte Undo- bzw. Redo-Schritt ist. Wichtig ist, dass eine entsprechende Visualisierung weder den Benutzer noch andere gleichzeitig arbeitende Benutzer im normalen Arbeitsfluss stören darf.

In den nachfolgenden Abschnitten werden vier regionale Undo-Techniken vorgestellt. Die verschiedenen Verfahren wenden verschiedene der eben erläuterten Ansätze an, um die Designziele und Herausforderungen in Bezug auf die Entwicklung von regionalen Undo-Konzepten bestmöglich umzusetzen:

Selektionstechnik (Manuell): Die Benutzer bestimmen manuell unter Verwendung eines Selektionswerkzeuges Regionen für Undo/Redo.

Clustering-Technik (Inhaltsbezogen): Alle zusammengehörigen Inhalte eines Dokumentes werden aufgrund ihrer räumlichen Anordnung in Cluster zusammengefasst, welche wiederum als Regionen für Undo/Redo verwendet werden.

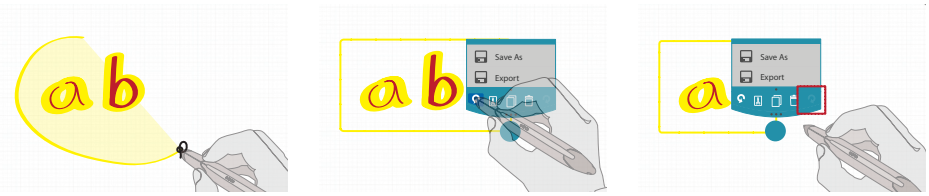
Guideline-Technik (Positionsbezogen): Der aktuelle Arbeitsbereich eines Benutzers bestimmt die aktuelle Region für Undo/Redo. Verlagert sich der Arbeitsbereich, verlagert sich auch seine Undo-Region mit.

Proximity-Technik (Blickfeldbezogen): Der wahrgenommene Bereich eines Benutzers wird als Region für Undo/Redo verwendet. Dazu wird ein angenähertes Blickfeld des Benutzers ermittelt.

3.3.1 Selektionstechnik

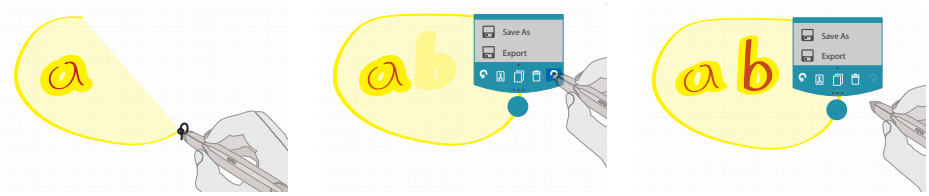
Eine sehr einfache Möglichkeit, um ein regionales Undo in einer Anwendung zu erreichen, ist eine *manuelle Regionsbestimmung* durch die Benutzer. Jeder Benutzer bestimmt im Falle eines Undo-Aufrufs jenen Bereich, auf den sich das auszuführende Undo beziehen soll. Damit ein Benutzer manuell Regionen bestimmen kann, benötigt er ein dafür geeignetes Werkzeug. Eine entsprechende Funktionalität bietet in vielen Fällen das *Selektionswerkzeug*, das in beinahe jedem grafischen Editor vorhanden ist. Aus diesem Grund kann die Funktionalität des Selektionswerkzeugs mit relativ wenig Aufwand um die Unterstützung von regionalem Undo erweitert werden. Wünscht ein Benutzer ein regionales Undo, kann er mit dem Selektionswerkzeug jene Dokumentteile selektieren, auf die sich das Undo beziehen soll. Über das Kontextmenü der Selektion kann das Undo anschließend ausgeführt werden.

Nichtsdestoweniger kann die Nutzung von selektierten Dokumentteilen auch zu Problemen führen. Werden Inhalte aus der Selektion rückgängig gemacht und somit gelöscht, verschwinden sie auch aus der Selektion. Der rück-



(a) (b) (c)

Abbildung 3.2: Der Benutzer selektiert Inhalte mit einem Lasso-Werkzeug (a). Anschließend führt der Benutzer ein Undo auf die selektierten Dokumentteile durch (b) und *b* wird gelöscht. Das Problem ist, dass der gelöschte Inhalt aus der Selektion verschwindet und nicht mehr wiederhergestellt werden kann (c).



(a) (b) (c)

Abbildung 3.3: Der Benutzer führt wieder eine Lasso-Selektion durch (a). Diesmal werden aber alle Operationen, die in die von dem Lasso aufgespannte Fläche fallen, für Undo und Redo berücksichtigt (b). Somit kann der Benutzer ohne weiteres den gelöschten Inhalt *b* wiederherstellen (c).

gängig gemachte Schritt kann anschließend aber nicht mehr wiederhergestellt werden, weil er sich nicht mehr in der Selektion befindet und üblicherweise auch nicht mehr selektiert werden kann (siehe Abbildung 3.2). Dies verstößt gegen das Designziel *Reproduzierbarkeit* und ist deshalb nicht zielführend.

Ein besserer Ansatz wäre daher statt der selektierten Inhalte die Selektionsfläche selbst zu nutzen. Nutzt der Benutzer beispielsweise ein Lasso-Werkzeug, so spannt er damit eine Fläche auf, die die zu rückgängig machenden Operationen beinhaltet. Diese Fläche kann dann als Region für Undo und Redo verwendet werden. Alle sichtbaren und nicht sichtbaren Inhalte, die sich innerhalb dieser Fläche befinden, können somit rückgängig gemacht und wiederhergestellt werden. Liegt z. B. in der aufgespannten Fläche eine gelöschte aber wiederherstellbare Operation, so kann diese mit dem Kontextmenü ohne weiteres wiederhergestellt werden (siehe Abbildung 3.3).

Stärken und Schwächen Selektionstechnik

Grundsätzlich kann mit der Selektionstechnik eine sehr einfache und flexible Einbindung von regionalem Undo in eine Anwendung erfolgen. Der Mehraufwand der Selektion bei jedem Undo-Aufruf stellt aber ein erhebliches Problem dar. Die Stärken und Schwächen der Selektionstechnik sind in Tabelle 3.1 angeführt.

Tabelle 3.1: Stärken und Schwächen der Selektionstechnik

Stärken	Schwächen
einfache Integration	erfordert hohe Merkfähigkeit der Benutzer
flexible Regionsbestimmung	Mehraufwand durch manuelle Regionsbestimmung

Stärken Die Stärke dieser Lösung ist einerseits die *einfache Integration* und andererseits die *Flexibilität*, mit der Regionen bestimmt werden können. Unabhängig vom Inhalt und von semantischen Zusammenhängen können mittels Selektionswerkzeug beliebige Regionen gebildet werden. Denkbar wäre daher, dass diese Lösung zusätzlich zu einem anderen regionalen Undo-Konzept in eine Anwendung integriert wird. Somit kann der Benutzer im normalen Arbeitsfluss schnell und ungestört arbeiten und hat, wenn notwendig, zusätzlich die Möglichkeit der Selektionsmethode zur diffizilen Regionsbestimmung (wie z. B. der Bearbeitung von einzelnen Dokumentteilen).

Schwächen Um nicht sichtbare und gelöschte Inhalte wiederherstellen zu können, muss sich der Benutzer *genau erinnern, wo* diese im Dokument platziert waren. Gerade in längeren Arbeitsprozessen stellt dies ein erhebliches Problem dar. Ein weiterer Nachteil dieser Lösung ist, dass der Benutzer bei jedem Undo oder Redo manuell die gewünschte Region bestimmen muss. Dies stellt einen *erheblichen Mehraufwand* für den Benutzer dar und stört seinen Arbeitsfluss. Gerade im Fehlerfall sind Benutzer oft gestresst und wollen ihre Fehler möglichst schnell und ohne Mehraufwand korrigieren. Dieser Umstand steht somit direkt in Konflikt mit den ausgegebenen Designzielen.

3.3.2 Clustering-Technik

Diese Technik versucht *Inhalte* aufgrund ihrer *räumlichen Anordnung* in sogenannte Cluster *zusammenzufassen*. Diese Cluster werden dann als Regionen für Undo und Redo verwendet (siehe Abbildung 3.4). Der Ansatz basiert auf dem Grundgedanken, dass getrennt arbeitende Personen nicht im glei-

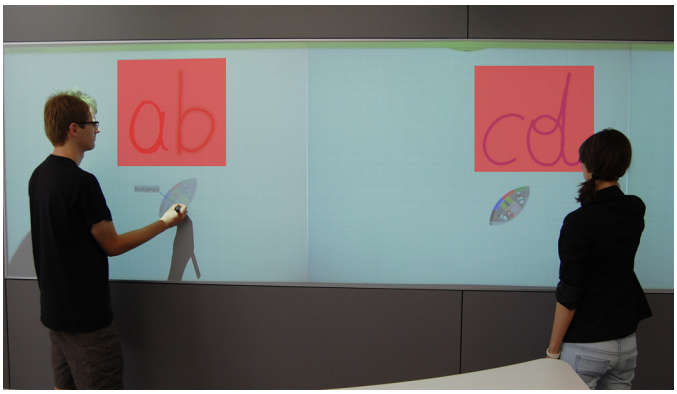


Abbildung 3.4: Bei der Clustering-Technik wird versucht, die zusammengehörigen Inhalte in einem Dokument zu Clustern zusammenzufassen, welche dann als Region für Undo/Redo verwendet werden.

chen Bereich der interaktiven Fläche arbeiten, während semantisch zusammengehörige Teile von Benutzern oder Gruppen auch räumlich nahe platziert werden. Bei einem etwaigen Aufruf von Undo oder Redo durch einen Benutzer wird dann eine der gebildeten Regionen (die vom Benutzer nächstgelegene) ausgewählt.

Bildung der Cluster

Die Bildung der inhaltlichen Cluster wird in einem mehrstufigen Verfahren bei jeder Dokumentänderung durchgeführt. Zunächst werden die Begrenzungen (eng. Bounds) eines jeden Inhaltes im Datenmodell gebildet. Diese werden dann abhängig von ihrer Gesamtgröße vergrößert. Anschließend werden alle vergrößerten Cluster gegeneinander abgeglichen und auf Überschneidungen überprüft. Überschneiden sich zwei Cluster, so werden diese vereinigt und zu einem neuen Cluster zusammengefasst. Dieses Verfahren wird rekursiv solange durchgeführt, bis keine neuen Cluster mehr gebildet

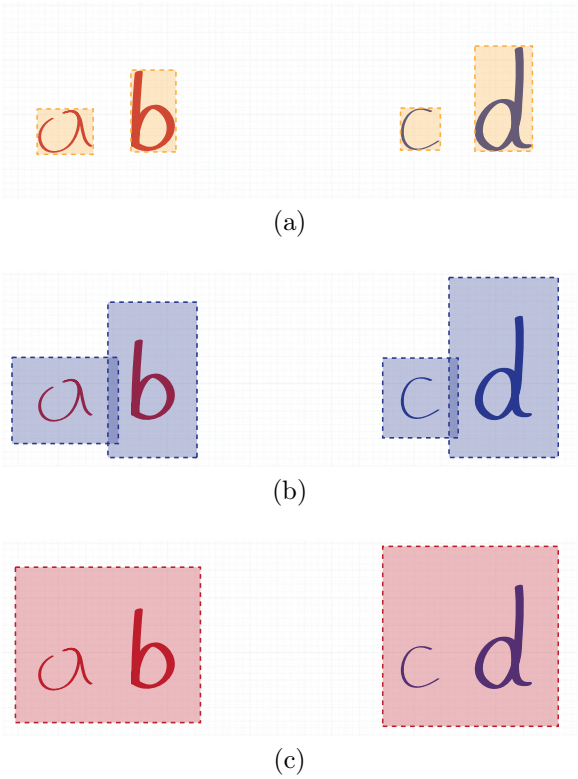


Abbildung 3.5: Die Cluster werden in einem mehrstufigen Verfahren gebildet. Zunächst werden die Begrenzungen der Inhalte gebildet (a), diese abhängig von deren Größe gleichmäßig vergrößert (b) und anschließend aufgrund von Überschneidungen rekursiv vereinigt (c).

werden können (siehe Abbildung 3.5). Durch inkrementelles Clustering kann das Verfahren in seiner Performance noch optimiert werden.

Für die Bildung der Cluster können sowohl Polygone (z. B. konvexe Hülle) als auch Rechtecke (z. B. Axis-Aligned Bounding-Box) verwendet werden. Da Überschneidungen von Polygonen aber wesentlich aufwendiger zu rechnen sind als einfache Bounding-Boxen sind rechteckige Cluster klar zu empfehlen. In dieser Arbeit wurden beide Verfahren realisiert, wobei gerade bei den polygonalen Clustern in großen Dokumenten und bei vielen Operationen Performanceeinbußen zu beobachten waren. Empfehlenswert ist daher, sofern dies aus Sicht des Interaktionsdesigns möglich ist, Rechtecke für das Clustering zu verwenden.

Auswahl eines Clusters

Bei jedem Aufruf von Undo oder Redo muss jene Region festgestellt werden, auf die sich die nächste Undo-Operation beziehen soll. Dafür kann jene Region ausgewählt werden, die am wenigsten weit vom Aufruf von Undo bzw. Redo entfernt ist. Daher ist es notwendig, die Position dieses Aufrufes zu ermitteln. Wird davon ausgegangen, dass der Benutzer Undo oder Redo über sein stiftbezogenes Benutzermenü aufruft und dass dieses in seiner näheren Arbeitsumgebung geöffnet ist, kann die Position des Benutzermenüs dazu verwendet werden. Durch einen einfachen Distanztest lässt sich dann jene Region bestimmen, die am wenigsten weit vom Benutzermenü entfernt ist (siehe Abbildung 3.6).

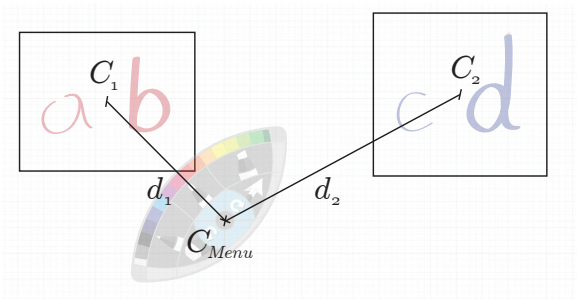
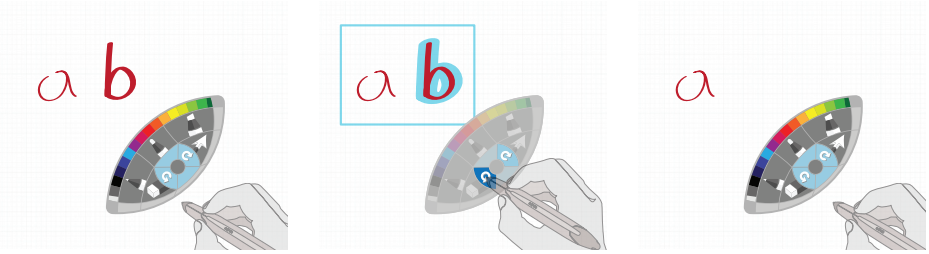


Abbildung 3.6: Als aktive Region eines Benutzers für Undo/Redo wird jene herangezogen, dessen Distanz d gegenüber dem Benutzermenü minimal ist. Bei C_1 und C_2 handelt es sich um geometrische Mittelpunkte der Cluster, wobei C_{Menu} den Mittelpunkt des stiftbezogenen Benutzermenüs darstellt.

Feedback Clustering-Technik

Beim Aufruf von Undo ist der Benutzer möglicherweise nicht sicher, auf welche Region sich der nächste Undo- bzw. Redo-Schritt bezieht. Deshalb ist es sinnvoll, die aktuell ausgewählte Region anzuzeigen. Eine Möglichkeit ist



(a) (b) (c)

Abbildung 3.7: Einem Benutzer wird die aktuelle Region eingeblendet, solange er den Undo-Button drückt (a,b). Hebt der Benutzer den Stift hoch, wird die Undo-Aktion durchgeführt und die Region wieder ausgeblendet (c).

einem Benutzer beim Drücken des Undo-Buttons die aktuell ausgewählte Region anzuzeigen (siehe Abbildung 3.7). Diese kann solange eingeblendet werden, bis der Benutzer den Stift aufhebt und die Undo-Aktion bestätigt. Im Falle einer falsch ausgewählten Region hat der Benutzer immer noch die Möglichkeit die Aktion abzubrechen, indem er den Stift außerhalb des Undo- oder Redo-Buttons aufhebt. Ein derartiges Verhalten sind die Benutzer von herkömmlichen User Interfaces weitestgehend gewöhnt. Im Zuge der Entwicklung dieses Verfahrens wurden auch Visualisierungen getestet, die dem Benutzer laufend die aktuelle Region anzeigen, sodass dieser bereits vor dem Undo-Aufruf zweifelsfrei weiß, auf welche Region sich der nächste Undo- bzw. Redo-Schritt beziehen wird. Die Benutzer empfanden aber eine dauerhafte Anzeige der Undo-Regionen im normalen Arbeitsfluss als zu störend.

Stärken und Schwächen Clustering-Technik

Die Clustering-Technik erfordert durch das inhaltsbezogene Clustering kaum Mehraufwand der Benutzer in Bezug auf Undo/Redo. Jedoch ist die Technik aufgrund der fixen Cluster nicht besonders flexibel. Die Stärken und Schwächen der Clustering-Technik sind in Tabelle 3.2 angeführt.

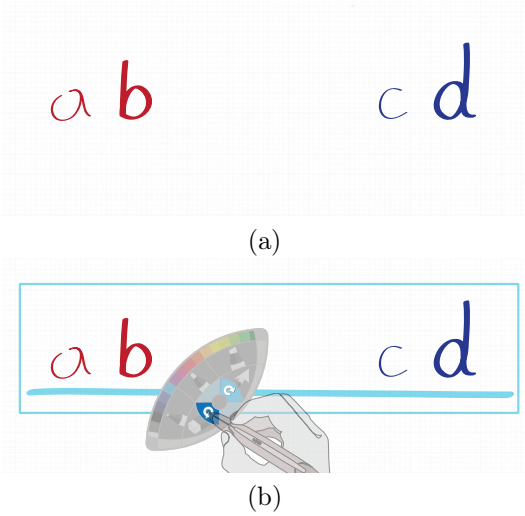
Tabelle 3.2: Stärken und Schwächen der Clustering-Technik

Stärken	Schwächen
automatische Regionsbestimmung	fixe Regionen, wenig Flexibilität
kaum Mehraufwand für die Benutzer	mögliche Diskrepanz zwischen vermuteten und gebildeten Regionen
	kontinuierliches Wachstum der gebildeten Regionen

Stärken Das Clustering-Konzept ist für den Benutzer insofern angenehm, als dass es sich dabei um eine vollautomatische Lösung handelt, die vom Benutzer kaum Mehraufwand verlangt (außer der Platzierung des Menüs, die aber meistens implizit erfolgt). Wie gut das Konzept in der Praxis funktioniert, hängt davon ab, wie gut das Verfahren zusammenhängende Inhalte von Benutzern erkennt. Generell ist das genannte Verfahren eine relativ einfache und effiziente Methode um regionales Undo in einem großflächigen Dokument zu ermöglichen.

Schwächen Vor allem bei umfangreichen Dokumenten und längeren Arbeitsprozessen können im Zusammenhang mit dem genannten Konzept Probleme auftreten. Ein großer Nachteil dieser Lösung ist, dass die gebildeten Regionen im weitesten Sinne *fix* sind und *nicht mehr getrennt* werden können. Das Konzept fasst die Inhalte aufgrund räumlicher Nähe zusammen und daher ist bei Fehlern in der Zusammenfassung ein Korrigieren dieser per se nicht möglich. Eine mögliche Lösung wäre ein zusätzliches Werkzeug zur Verfügung zu stellen, welches bei etwaigen Problemen ein Auftrennen von Regionen oder das manuelle Bilden von Regionen erlaubt. In diesem Zuge wäre die Kombination mit der Nutzung eines Selektionswerkzeuges (siehe Abschnitt 3.3.1) für die Bestimmung von diffizilen Regionen denkbar.

Ein weiteres wesentliches Problem ist, dass bei einem additiven Verfahren wie diesem die gebildeten Regionen mit der Anzahl an Operationen *kontinu-*



(a)

(b)

Abbildung 3.8: Dass visuelle Regionen nicht immer den für Undo/Redo gebildeten Regionen entsprechen, führt bei den Benutzern unter Umständen zu Verwirrung. Auch wenn für den Benutzer zwei visuelle Regionen im Dokument zu erkennen sind (a), können bereits gelöschte und somit nicht sichtbare Inhalte (blau markiert) trotzdem die Regionen verbinden (b).

ierlich wachsen. Werden immer *alle* Operationen in die Bildung der Cluster miteinbezogen, so ergeben sich mit Fortdauer des Arbeitsprozesses derart große Regionen, dass ein Unterschied gegenüber globalem bzw. dokumentenbasierten Undo nicht mehr gegeben ist. Eine Lösung hierfür wäre, dass der Verlauf begrenzt wird und beispielsweise nur die letzten 30 Schritte in die Bildung der Cluster miteinbezogen werden.

Schwierig ist für Benutzer auch die *gebildeten Regionen* von den *sichtbaren Inhalten* im Dokument (und daraus resultierenden visuellen Regionen) *zu unterscheiden*. Sind für einen Benutzer beispielsweise aufgrund des Inhaltes zwei visuelle Regionen im Dokument zu erkennen, kann es trotzdem vorkommen, dass ein bereits gelöschter und somit nicht sichtbarer Inhalt die beiden Regionen verbindet (siehe Abbildung 3.8). Dieser Umstand kann bei Benutzern, die mit dem Umgang dieser Technik wenig Erfahrung haben, zu Verwirrungen führen. Ein Ansatz hierbei wäre, dass zum Beispiel nur die letzten 10 Undo-Schritte, welche nicht mehr sichtbare Operationen betreffen, in die Bildung der Regionen miteinbezogen werden. Somit korrespondieren die sichtbaren Inhalte mehr mit den gebildeten Regionen und sind für den Benutzer insgesamt durchschaubarer.

3.3.3 Guideline-Technik

Die Guideline-Technik basiert auf dem Grundgedanken, dass *jeder Benutzer* über einen *gewissen Bereich* am Whiteboard verfügt, der seinem Arbeitsbereich entspricht (siehe Abbildung 3.9). Alle Operationen, die in diesen Bereich fallen, werden für Undo und Redo herangezogen. Der Bereich reicht jeweils über die gesamte Whiteboard-Höhe, da Benutzer an einer vertikalen Fläche in der Regel nicht unter- bzw. übereinander arbeiten.

Abbildung 3.9: Jeder Benutzer verfügt über einen gewissen Bereich, der seinem Arbeitsbereich entspricht und über die gesamte Höhe des Whiteboards reicht. Alle Operationen, die in diesen Bereich fallen, werden für Undo und Redo herangezogen.

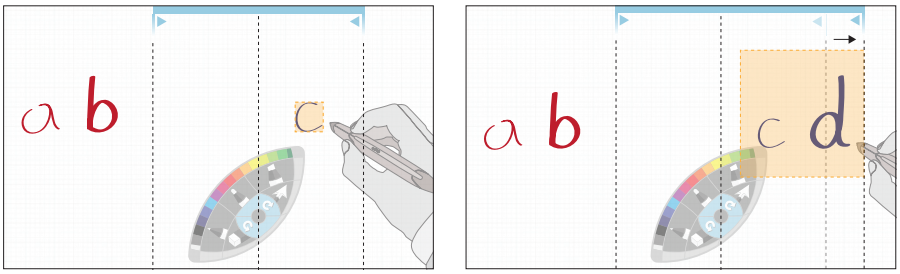


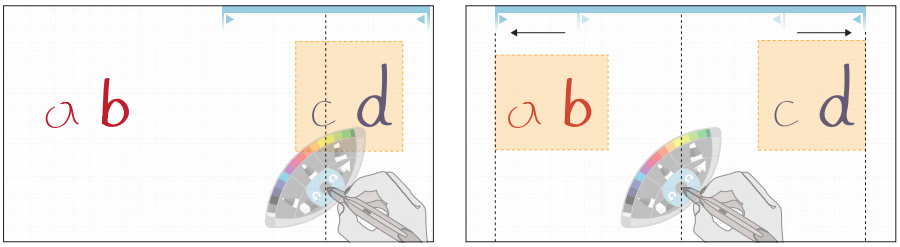
Abbildung 3.10: Die verschiedenen Bereiche werden laufend aufgrund der in der Clustering-Technik gebildeten Cluster angepasst. Jeder Bereich wird immer so angepasst, dass er alle angrenzenden Cluster miteinschließt.

Als Metapher zur Beschreibung dieser Technik könnte eine über dem Benutzer schwebende Wolke herangezogen werden, die anzeigt, in welchem Bereich des Whiteboards der Benutzer gerade aktiv ist. Alle Operationen und Inhalte die in diesen Bereich fallen, sind von den Undo- und Redo-Aufrufen des Benutzers betroffen. Der Name „Guideline-Technik“ leitet sich daraus ab, dass der Bereich eines Benutzers jeweils von zwei imaginären Hilfslinien eingeschlossen wird.

Im Gegensatz zur Clustering-Technik bestimmt also nicht der Inhalt selbst die Regionen, sondern die Benutzer und deren Arbeitsbereiche. Jeder Bereich eines Benutzers verfügt zunächst über eine initiale Ausbreitung, die üblicherweise dem Bereich entspricht, den ein Benutzer bearbeiten und erreichen kann, ohne die Position zu wechseln. Verlagert ein Benutzer seinen Arbeitsbereich, so verlagert sich auch der Bereich entsprechend mit. Darüber hinaus nutzt die Guideline-Technik die gebildeten Cluster aus der Clustering-Technik zur intelligenten Anpassung des Arbeitsbereiches des Benutzers. Grenzen Inhalte bzw. Cluster an den aktuellen Bereich eines Benutzers an, so wird der Bereich immer so angepasst, dass der Cluster in den Bereich miteingeschlossen wird (siehe Abbildung 3.10). Durch das Miteinbeziehen der Cluster wächst ein Bereich auch automatisch mit den von einem Benutzer erstellten Inhalten mit.

Bestimmung des Guideline-Bereiches

Wie bei der Clustering-Technik wird angenommen, dass ein Benutzer sein stiftbezogenes Benutzermenü beim Aufruf von Undo- oder Redo in der näheren Arbeitsumgebung geöffnet hat. Deshalb ist die Position dieses benutzerbezogenen Menüs auch wieder der Ausgangspunkt für die Positionierung des Bereiches. Der Mittelpunkt des Benutzermenüs stellt somit auch das Zentrum des initialen Bereiches dar. Bei jeder Änderung der Inhalte oder jeder Positionsänderung des Benutzermenüs wird der Bereich mitbewegt und aufgrund der gebildeten Cluster angepasst. Bewegt ein Benutzer sein stift-



(a) (b)

Abbildung 3.11: Für die Positionierung der Bereiche wird wie bei der Clustering-Technik die Position des stiftbezogenen Benutzermenüs genutzt. Ist das Benutzermenü nahe bei einem Inhalt so wird aufgrund der automatischen Anpassung nur dieser Cluster in den Bereich eingeschlossen (a). Befindet sich das Menü zwischen zwei Inhalten, so werden beide Cluster in den aktuellen Bereich miteinbezogen (b).

bezogenes Menü, bewegt sich auch der Bereich entsprechend mit. Trifft der Bereich auf gebildete Cluster werden diese automatisch miteingeschlossen.

Befindet sich z. B. ein Menü nahe einem Inhalt (Cluster) so schließt der Bereich nur diesen Cluster ein (siehe Abbildung 3.11 (a)). Befindet sich das Menü aber zwischen zwei Inhalten so werden beide Inhalte in den Bereich eingeschlossen (siehe Abbildung 3.11 (b)).

Feedback Guideline-Technik

Zunächst wurde die Guideline-Technik so visualisiert, dass jeder Bereich von zwei visuellen Hilfslinien eingeschlossen war. Erste Tests zeigten aber, dass die Benutzer durch die sich bei jeder Verschiebung eines Menüs mitbewegenden Hilfslinien zu stark gestört werden. Deshalb wurde ein subtilerer Ansatz gewählt, der den Bereich eines Benutzers als halbtransparenten Streifen an der Oberseite der Anwendung darstellt. Jeder Streifen ist farbkodiert in der Farbe des Benutzers, welche sich auch im stiftbezogenen Benutzermenü wiederfindet. Darüber hinaus wird die Visualisierung noch links und rechts von Pfeilen nach innen unterstützt, welche die Übersichtlichkeit steigern. Gerade bei sich überlagernden Bereichen (z. B. von Benutzern einer Gruppe) lassen sich durch die Farbkodierung und den visuellen Elementen die Bereiche jedes Benutzers trotzdem noch gut erkennen. Die Darstellung von sich überlagernden Bereichen von zwei Benutzern ist in Abbildung 3.12 abgebildet.

Da eine unmittelbare Bewegung der Bereiche oder die unmittelbare Ausdehnung der Bereiche für Benutzer irritierend wirken kann, werden die Bewegungen und Ausdehnungen des Bereiches geglättet. Somit wirken auch unmittelbare und plötzliche Änderungen des Bereiches (z. B. Miteinbeziehen eines neuen Clusters) weich und angenehm.

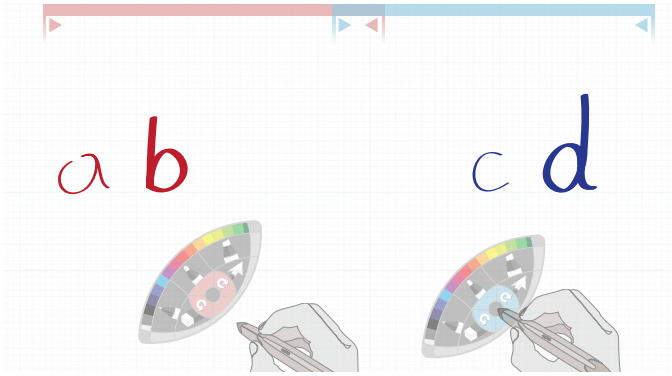
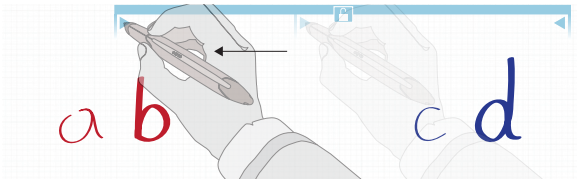


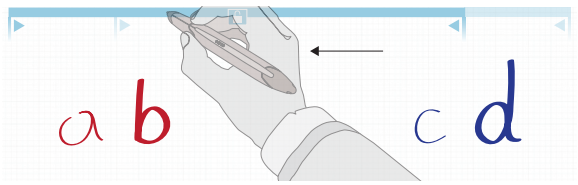
Abbildung 3.12: Die Bereiche werden durch halbtransparente Streifen in der Farbe des Benutzers an der Oberkante der Anwendung dargestellt. Pfeile nach innen unterstützen die Übersichtlichkeit bei überlappenden Bereichen von beispielsweise Gruppenmitgliedern.

Flexibilität Guideline-Technik

Im Gegensatz zu den fixen Regionen der Clustering-Technik erlaubt diese Technik den Bereich eines Benutzers frei zu adaptieren. Dazu kann nicht nur die gesamte Region verschoben werden, sondern auch die Begrenzungen (die Hilfslinien) links und rechts angepasst werden (siehe Abbildung 3.13). Dabei kann immer nur der Benutzer selbst seinen Bereich verändern, während



(a)



(b)

Abbildung 3.13: Um mehr Flexibilität zu erlangen kann jeder Bereich durch den Benutzer frei adaptiert werden. Es können sowohl die Begrenzungen links und rechts angepasst werden (a), als auch der ganze Bereich verschoben werden (b). Dies ermöglicht dem Benutzer eine diffizile Regionsbestimmung.

andere Benutzer nicht in der Lage sind, fremde Bereiche zu adaptieren.

Wird ein Bereich von einem Benutzer adaptiert, so wird dieser gesperrt und als solcher gekennzeichnet (Schloss-Symbol). Ist ein Bereich gesperrt, so ist die automatische Positionierung und Anpassung bei neuen Inhalten deaktiviert. Erst wenn der Benutzer den Bereich wieder entsperrt (durch Drücken des angezeigten Schloss-Symbols), werden diese Mechanismen wieder aktiviert und der Bereich wird wieder automatisch verwaltet.

Mit dieser Funktionalität erlangt die Guideline-Technik viel Flexibilität. Im normalen Arbeitsfluss können Benutzer auf eine vollautomatische Lösung zurückgreifen, die sich aufgrund der automatischen Anpassung sehr intuitiv verhält. Benötigt der Benutzer aber beispielsweise ein Undo auf einen Teilbereich der Region, kann der Bereich einfach entsprechend adaptiert werden.

Stärken und Schwächen Guideline-Technik

Die Guideline-Technik stellt eine sehr flexible Technik dar. Im normalen Arbeitsfluss arbeitet das Verfahren weitestgehend vollautomatisch, während aber durch die Adaptierung der Bereiche auch eine diffizile Regionsbestimmung möglich ist. Die Stärken und Schwächen der Clustering-Technik sind in Tabelle 3.3 angeführt.

Tabelle 3.3: Stärken und Schwächen der Guideline-Technik

Stärken	Schwächen
automatische Regionsbestimmung	kontinuierliches Wachsen der gebildeten Regionen
kaum Mehraufwand	
hohe Flexibilität	

Stärken Entgegen der relativ unflexiblen Funktionsweise der Clustering-Technik zeichnet sich diese Lösung vor allem durch seine *Flexibilität* aus. Arbeitet ein Benutzer im normalen Arbeitsfluss, so erledigt das Verfahren seine Dienste weitestgehend vollautomatisch. Wird jedoch in speziellen Fällen eine diffizile Regionsbestimmung benötigt, so kann der Benutzer seinen Bereich einfach adaptieren und so anpassen, wie er ihn gerade benötigt. Kehrt der Benutzer wieder zur eigentlichen Arbeit zurück, kann durch Drücken eines einzigen Buttons die vollautomatische Funktionsweise wiederhergestellt werden. Die Flexibilität dieses Konzeptes könnte noch weiter gesteigert werden, wenn die Benutzer ihre Bereiche nicht nur vertikal sondern auch horizontal einschränken könnten.

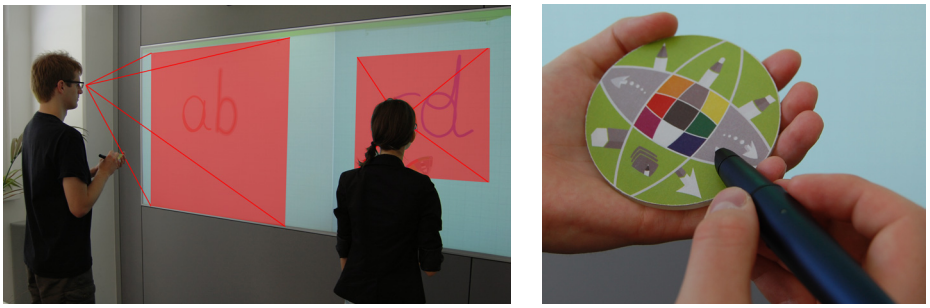
Erfahrungen aus Tests der verschiedenen Konzepte zeigten, dass es Benutzer in der Regel leichter fällt, ihren eigenen Arbeitsbereich zu verwalten

als die inhaltlichen Cluster zu erkennen und den richtigen auszuwählen. Dabei empfanden es die meisten Benutzer als völlig intuitiv, dass bei einem Undo-Aufruf alle in „ihrem“ Bereich befindlichen Operationen betroffen sind.

Schwächen Wie auch bei der Clustering-Technik bleiben auch bei diesem Verfahren durch die Miteinbeziehung der Cluster die Probleme mit dem *kontinuierlichen Wachstum* der Cluster bzw. Regionen bestehen. Werden auch hier alle Operationen zur Cluster-Bildung berücksichtigt, wachsen auch bei diesem Verfahren die Bereiche der Nutzer mit Fortdauer so stark an, dass sich das regionale Undo kaum anders als ein globales oder dokumentenbasiertes Undo verhält. Auch hier beseitigt die Beschränkung der Anzahl der Operationen zur Cluster-Bildung das Problem weitestgehend.

3.3.4 Proximity-Technik

Sowohl die Clustering- als auch die Guideline-Technik schließen von der Position des stiftbezogenen Benutzermenüs auf die Position bzw. den Arbeitsbereich des Benutzers. Aufgrund dieser Position wird eine Region für Undo und Redo ausgewählt. Diese Verallgemeinerung funktioniert in der Regel gut, führt aber zu Problemen wenn nicht garantiert ist, dass diese Position auch tatsächlich der Position des Benutzers entspricht. Ein mögliches Beispiel hierfür wäre die Benutzung von physischen Werkzeug-Paletten anstelle des Benutzermenüs (siehe Abbildung 3.14 (b)). Paletten werden an interaktiven Whiteboards eingesetzt, um in einer äußerst schnellen und bequemen Art und Weise Werkzeuge umzuschalten. Wird von Benutzern eine solche physische Palette (auch für Undo und Redo) genutzt, so lässt sich über das digitale Benutzermenü nicht mehr rückschließen, wo sich der Benutzer un-



(a) (b)

Abbildung 3.14: Die Proximity-Technik basiert auf dem Ansatz, den aktuell wahrgenommenen Bereich des Benutzers, angenähert durch sein Blickfeld, als Undo- bzw. Redo-Region zu verwenden (a). Dies ist vor allem dann notwendig, wenn die Position des Benutzers in der Anwendung nicht ermittelt werden kann, z. B. bei der Nutzung von Werkzeugpaletten (b).

gefähr befindet. Deshalb braucht es ein weiteres Konzept, welches auch ein derartiges Nutzerverhalten einschließt.

Wie in der Studie zur Erwartungshaltung in Abschnitt 1.4 ermittelt wurde, wollen Benutzer üblicherweise nicht von ihrer Position aus Operationen aus einem ganz anderen Bereich des Whiteboards rückgängig machen. Dies wird erst möglich, wenn die Benutzer ihren Standort wechseln und einen anderen Bereich fokussieren (siehe Abschnitt 1.4.4). Daraus kann abgeleitet werden, dass sich ein Undo oder Redo eines Benutzers immer auf *jenen Bereich* beziehen soll, den er gerade *wahrnimmt*. Der aktuell wahrgenommene Bereich eines Benutzers kann durch Ermittlung seines *Blickfeldes* angenähert werden (siehe Abbildung 3.14 (a)).

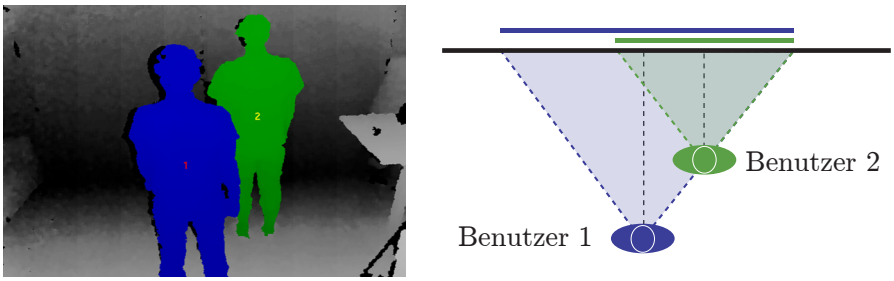
Der vorgestellte Ansatz basiert also auf der Idee, alle sich vor dem Whiteboard befindlichen Personen zu erkennen und deren Distanz zum Whiteboard zu ermitteln. Aus der Position und der Distanz zum Whiteboard lässt sich dann das ungefähre Blickfeld des Benutzers errechnen, welches wiederum als Region für Undo- bzw. Redo-Operationen herangezogen wird (siehe Abbildung 3.15). Die Benutzer bestimmen also mit ihrer Position und Positionswechsel den Geltungsbereich für Undo und Redo. Dazu ist ein Tracking-Verfahren notwendig, dass die Erkennung und Verfolgung von Personen zuverlässig ermöglicht.

Tracking-Verfahren

Zur Ermittlung des Blickfeldes einer Person gibt es auf dem Markt eine Reihe von Tracking-Technologien, die aber allesamt zusätzliches mobiles Equipment erfordern und somit für den Einsatz an einem digitalen Whiteboard nur bedingt geeignet sind. Ein Whiteboard soll vor allem spontane und kreative Arbeit von verschiedensten Personen ermöglichen, ohne dass die Benutzer umständliches Tracking-Equipment (z. B. Head-Mounted-Tracker) anlegen müssen. Das wäre für die Nutzer umständlich und würde eine wesentliche Einschränkung ihrer Kreativität darstellen. Deshalb braucht es für das Whiteboard eine Technologie, die ohne zusätzliches Equipment am Benutzer auskommt und eine gute Balance zwischen Genauigkeit und Kosten bietet.

Alle diese Anforderungen kann das für die Spielkonsole Xbox 360® konzipierte Tracking-System Microsoft Kinect[1] erfüllen. Das Tracking-System verfügt über eine Kamera und zwei Infarot-Tiefensensoren, die für jeden Kameraframe auch das zugehörige Tiefenbild ermitteln. Das Tracking der Sensoren ist so präzise, dass damit ohne weiteres Personen und deren Bewegungen (z. B. Gesten) erkannt werden. Durch Veröffentlichung der Treiber und einer entsprechenden Softwarebibliothek kann die Kinect auch für andere Tracking-Projekte verwendet werden.

[1] http://www.xbox.com/kinect



(a) (b)

Abbildung 3.15: Eine Microsoft Kinect wird für das Erkennen und Verfolgen von Personen vor dem Whiteboard verwendet (a). Dabei wird von einer eigens entwickelten Trackingapplikation aufgrund der Sensordaten die Position und die Entfernung der Benutzer zum Whiteboard ermittelt (b).

Das vom Kinect-Sensor-Hersteller PrimeSense veröffentlichte Software Development Kit (SDK) bietet neben dem Zugriff auf den Kinect-Sensor und das erzeugte Tiefenbild auch eine Vielzahl von Anwendungsbeispielen[2]. Ein Anwendungsbeispiel davon ist das Erkennen und Verfolgen von Personen im Sichtfeld der Kinect, welches für das folgende Konzept als Basis verwendet wurde.

Konzeptionelle Umsetzung Proximity-Technik

Jede Person im Sichtfeld der Kinect bekommt eine eigene Identifikationsnummer, die sie auch behält, wenn mehrere Personen sich im Raum kreuzen oder kurz aus der Kinect-Reichweite verschwinden. Verschwindet eine Person, so erwartet das Programm, dass sie in einer gewissen Zeit wieder an der gleichen Position erscheint. In solch einem Fall wird dieser Person wieder die gleiche Identifikationsnummer zugewiesen. Somit lassen sich mit diesem Programm sehr robust Personen erkennen und verfolgen.

Die Kinect wird so montiert, dass sie von hinten den gesamten Bereich vor dem Whiteboard einsehen kann (siehe Abbildung 3.15 (a)). Somit liefert das Programm in Echtzeit die Koordinaten aller Personen, die sich vor dem Whiteboard befinden. Diese Koordinaten können anhand entsprechender Kalibrierungsdaten auf die Position und die Entfernung zu dem Whiteboard umgerechnet werden (siehe Abbildung 3.15 (b)).

Jede von der Kinect ermittelte Person muss zunächst in einem ersten Schritt auf einen korrespondierenden Benutzer in der Anwendung abgebildet werden. Dieses „Mapping“ kann immer dann durchgeführt werden, wenn ein noch nicht abgebildeter Whiteboard-Benutzer die Whiteboard-Fläche berührt. Die von der Kinect ermittelten X-, Y- und Z-Koordinaten werden so

[2] http://www.openni.org/

umgerechnet, dass sie mit dem Stift-Berührungspunkt des Benutzers in Bildschirmkoordinaten verglichen werden können. Vereinfacht gesagt muss der von der Anwendung ermittelte Stift-Berührungspunkt eines Benutzers mit den von der Kinect ermittelten Daten übereinstimmen.

Dazu wird im ersten Schritt die vom Kinect-Tracking ermittelte Koordinate in X-Richtung in den Pixelraum der Anwendung umgerechnet. Mittels Kalibrierungsdaten wird die X-Koordinate zunächst in einen Wertebereich $[0, 1]$ normalisiert. Ein Wert von $0,5$ würde somit bedeuten, dass sich ein Benutzer genau in der Mitte vor dem Whiteboard befindet. Durch die Anzahl der horizontalen Pixel des Whiteboards ergibt sich dann die Pixelposition im Koordinatensystem der Anwendung. Für das in dieser Arbeit verwendete Setup mit einer Auflösung von 3072×768 Pixel und dem Wert $0,5$ ergibt sich als errechnete Pixelposition:

$$x_{Kinect} = x_{abs} \cdot w = 0,5 \cdot 3072 = 1536 \; . \tag{3.1}$$

Im nächsten Schritt wird das „Mapping“ einer von der Kinect getrackten Person $U_{Kinect} = (x_{Kinect}, z_{Kinect})$ und einem Anwendungsbenutzer $U_{Pen} = (x_{Pen}, y_{Pen})$ durchgeführt. Eine Übereinstimmung ist durch

$$\begin{aligned} \text{IsSameUser}(U_{Kinect}, U_{Pen}) := \neg\, &\text{IsAlreadyMapped}(U_{Kinect}) \wedge \\ &|x_{Kinect} - x_{Pen}| < \tau_{Pixel} \wedge \\ &z_{Kinect} < \tau_{Distance} \end{aligned} \tag{3.2}$$

gegeben, wobei es sich bei x_{Kinect}, x_{Pen}, y_{Pen} um Bildschirmkoordinaten und bei z_{Kinect} um die Entfernung zum Whiteboard in Millimetern handelt. Die Schwellwerte müssen für jedes Setup bestimmt werden. Anhand von Messungen mit dem in dieser Arbeit beschriebenen Whiteboard-Setup (Abmessungen: $4,5 \times 1,2$ Meter, Auflösung: 3072×768 Pixel) wurden folgende Schwellwerte für Formel 3.2 festgelegt. Die maximal zulässige Abweichung der X-Koordinaten τ_{Pixel} beträgt 200 Pixel und die maximale Whiteboard-Distanz einer interagierenden Person $\tau_{Distance}$ beträgt 700 Millimeter.

Anschließend kann aufgrund der ermittelten Daten der Kinect auf das Blickfeld der jeweiligen Benutzer (bezogen auf die Anwendung) zurückgerechnet werden. Dabei wird das Blickfeld nur in x-Richtung ermittelt, da wie bei der Guideline-Technik verallgemeinert wird, dass jeder Bereich eines Benutzers über die gesamte Höhe des Whiteboards reicht.

Zunächst muss zur Umrechnung des Blickfeldes von Millimeter auf Pixel ermittelt werden, wie viele Millimeter ein Pixel auf der projizierten Fläche (Whiteboard) misst. Mit dieser Größe kann dann mithilfe von z_{Kinect} das halbe Blickfeld in Pixel errechnet werden. Zur Errechnung der projizierten Pixelgröße in Millimeter ergibt sich aufgrund der Whiteboard-Breite in Millimeter sowie der Anzahl von horizontalen Pixeln des Whiteboards:

$$p_{Size} = \frac{w_{\text{mm}}}{w_{\text{px}}} \; . \tag{3.3}$$

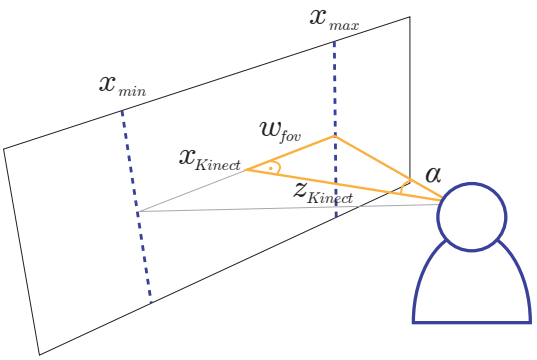


Abbildung 3.16: Aufgrund der Entfernung vom Whiteboard z_{Kinect} und der Position vor dem Whiteboard x_{Kinect} kann unter Annahme des Sichtwinkels die minimale und maximale Grenze des Blickfeldes ermittelt werden.

Für das in dieser Arbeit verwendete Whiteboard-Setup mit einer Breite von $4,5$ Metern sowie einer Auflösung von 3072×768 Pixel ergibt sich als Größe für ein projiziertes Pixel

$$p_{Size} = \frac{4500 \text{ mm}}{3072 \text{ px}} = 1,46 \text{ mm}. \tag{3.4}$$

Unter Einbeziehung von p_{Size} sowie der von der Kinect ermittelten Entfernung z_{Kinect} kann nun das halbe Blickfeld (in Pixel) ermittelt werden. Dieses ergibt sich durch:

$$w_{FOV} = \frac{z_{Kinect} \cdot \tan(\alpha)}{p_{Size}} \, . \tag{3.5}$$

Messungen ergaben als guten Wert für den wahrnehmbaren Blickwinkel α eines Benutzers am Whiteboard *unter Einbeziehung von Augenbewegungen aber keinen Kopfdrehungen* 62,5° nasal. Ein Wert von 50° bis 60° Grad nasal gilt für das Sichtfeld (Field Of View) des menschlichen Auges als wissenschaftlich erwiesen [23].

Aus der ermittelten halben Blickfeldbreite ergibt sich anschließend die linke minimale Grenze x_{min} sowie die rechte maximale Grenze x_{max} des Blickfeldes in Bildschirmkoordinaten durch

$$\begin{aligned} x_{min} &= x_{Kinect} - w_{FOV} \text{ und} \\ x_{max} &= x_{Kinect} + w_{FOV} \, . \end{aligned} \tag{3.6}$$

In Abbildung 3.16 wird die Annäherung des Blickfeldes einer am Whiteboard arbeitenden Person veranschaulicht.

Aus x_{min}, x_{max} und der Höhe der Anwendung in Pixel ergibt sich somit für jeden Benutzer U_{Pen} der aktuell wahrgenommene Bereich, welcher anschließend als Undo bzw. Redo-Region verwendet werden kann.

Feedback Proximity-Technik

Die Darstellung des Blickfeldes eines jeden Benutzers orientiert sich an der Gestaltung der Guideline-Technik (siehe Abschnitt 3.3.3). Mit dem einzigen Unterschied, dass der Bereich dieses Konzeptes vom Benutzer nicht explizit adaptiert werden kann. Eine etwaige Adaptierung geschieht bei der Proximity-Technik implizit über Positionswechsel der Benutzer. Der aktuelle Bereich wird wieder in Form eines Streifens an der Oberseite der Anwendung dargestellt. Eine Farbkodierung in der Farbe des entsprechenden Benutzers und Pfeile nach innen unterstützten die Übersichtlichkeit.

Damit schnelle Positionswechsel oder mögliches „Jittering“ des Trackings nicht zu ruckelnden und damit störenden Änderungen der dargestellten Streifen führt, sind auch die Bewegungen der Streifen in diesem Konzept geglättet. Außerdem muss sich das Blickfeld zuerst um einige Pixel verlagern, bevor die Positionierung des Streifens aktualisiert wird, um der Übertragung von „Jittering“ des Trackings in die Darstellung vorzubeugen.

Stärken und Schwächen Proximity-Technik

Die Proximity-Technik stellt eine sehr intuitive und universelle Methode zur Bestimmung der Undo-Regionen dar. Jedoch kann die Technik auf unerfahrene Whiteboard-Benutzer auch gewöhnungsbedürftig wirken, weil diese es oft nicht gewöhnt sind, Softwareteile wie beispielsweise den Undo-Geltungsbereich durch ihre Position oder ihr Blickfeld steuern zu können. Die Stärken und Schwächen der Proximity-Technik sind in Tabelle 3.4 angeführt.

Tabelle 3.4: Stärken und Schwächen der Proximity-Technik

Stärken	Schwächen
automatische Regionsbestimmung	nur angenähertes Blickfeld
kein Mehraufwand	mglw. gewöhnungsbedürftig
hohe Flexibilität durch Positionswechsel	
auch beim Einsatz von Werkzeugpaletten geeignet	
kreuzende Personen kein Problem	

Stärken Es wird angenommen, dass diese Technik aufgrund der Miteinbeziehung der tatsächlichen Position bzw. des angenäherten Blickfeldes eines Benutzers eine *vielversprechende und intuitive* Möglichkeit zur Umsetzung von regionalem Undo auf großen interaktiven Flächen darstellt. Arbeitet ein Benutzer nahe am Whiteboard, so richtet sich auch der Geltungsbereich von

Undo auf den kleinen, aktuellen Arbeitsbereich. Bewegt sich ein Benutzer weiter weg vom Whiteboard, um sich beispielsweise einen Überblick über das gesamte Dokument zu verschaffen, werden auch die Undo-Operationen auf das ganze Dokument ausgeweitet. Wechselt ein Benutzer seinen Standort, um beispielsweise bei einer anderen Person oder Gruppe mit- oder weiterzuarbeiten, wandert auch sein aktueller Bereich mit. Somit ergibt sich eine *sehr intuitive Positionierung und Handhabung* des Geltungsbereiches von Undo. Das Konzept funktioniert außerdem sowohl beim Einsatz des digitalen Benutzermenüs als auch unter *Einsatz von physischen Werkzeugpaletten.*

Durch das *robuste Tracking* der Kinect sind auch Personen, die sich im Sichtfeld der Kinect *kreuzen und verdecken*, kein wesentliches Problem. In verschiedenen Tests dieses Konzeptes hat sich herausgestellt, dass das Tracking auch kreuzende oder verdeckte Personen nicht verliert. Wird eine Person verdeckt, wird für Undo und Redo das zuletzt bekannte Blickfeld weiterverwendet. In einem solchen Fall kann angenommen werden, dass sich die Person nicht bewegt. Würde sich die Person bewegen und (teilweise) aus dem verdeckten Bereich auftauchen, wird ihr vom Tracking wieder dieselbe Identifikationsnummer zugewiesen und ein neues Blickfeld berechnet.

Schwächen In einem optimalen Szenario würde das Tracking-System die genaue Kopfneigung, Kopfdrehung und Blickrichtung ermitteln und ein *exaktes Blickfeld* errechnen. Dies kann mit der Kinect nur schwer erreicht werden, da dafür die Auflösung des Tiefenbildes zu gering ist. Darüber hinaus kann diese Technik auf die Benutzer sehr *gewöhnungsbedürftig* wirken, da vor allem unerfahrene Whiteboard-Nutzer es nicht gewohnt sind, den Geltungsbereich für Undo/Redo oder andere Softwareteile durch ihre Position oder ihr Blickfeld steuern zu können.

3.4 Allgemeine Konzepte

Nachfolgend werden einige Konzepte vorgestellt, die unabhängig von regionalem Undo für jede Undo-Methode (beispielsweise auch globalem oder benutzerbezogenem Undo) realisiert werden können.

3.4.1 Visualisierung von Undo-Schritten

Neben einem visuellen Feedback für die aktuelle Region ist für den Benutzer auch ein Anzeigen der nächsten betroffenen Undo- bzw. Redo-Aktion hilfreich. Gerade bei der Verwendung von regionalem Undo und der gleichzeitigen Arbeit von mehreren Benutzern sind sich diese möglicherweise nicht sicher, welche Aktion jetzt tatsächlich vom nächsten Undo- oder Redo-Schritt betroffen ist. Deshalb ist es sinnvoll, den Benutzern bei Bedarf ein entsprechendes Feedback zu geben und den nächsten Schritt hervorzuheben.

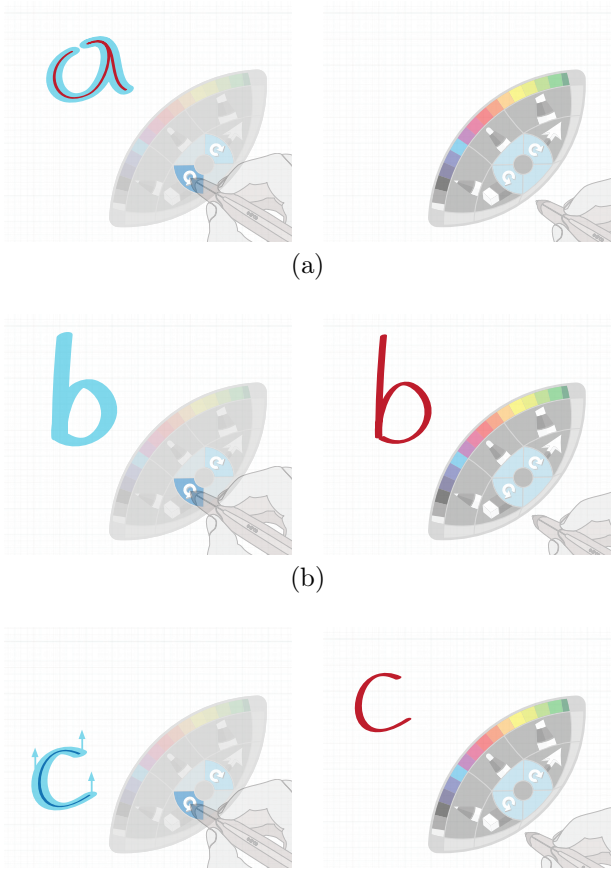


Abbildung 3.17: Unter Verwendung der Outline eines Einzelstriches kann der nächste Undo- bzw. Redo-Schritt einfach und wirkungsvoll visualisiert werden. Das jeweils linke Bild zeigt die Undo-Visualisierung und das jeweils rechte Bild den Zustand nach Durchführung der Undo-Aktion. Folgende Undo-Aktionen werden dargestellt: (a) Undo von Hinzufügen, (b) Undo von Löschen und Undo einer Transformation (c).

Für einen Benutzer sind bei einem etwaigen Aufruf einer Undo- oder Redo-Aktion folgende Informationen interessant:

- Position der Aktion
- Art der Aktion (Löschen, Hinzufügen, usw.)

Unter Einbeziehung dieser Anforderungen und von visuellen „Outlines“ der jeweiligen Dokumentteile kann folgendes Feedback gegeben werden:

Undo von Hinzufügen Ein durch eine Outline umrandeter Inhalt kennzeichnet, dass bei einem Undo-Aufruf das Hinzufügen dieses Inhaltes rückgängig gemacht und somit gelöscht wird. Siehe Abbildung 3.17 (a).

Undo von Löschen Eine freistehende Outline würde kennzeichnen, dass ein gelöschter Inhalt an dieser Stelle wieder eingefügt wird. Siehe Abbildung 3.17 (b).

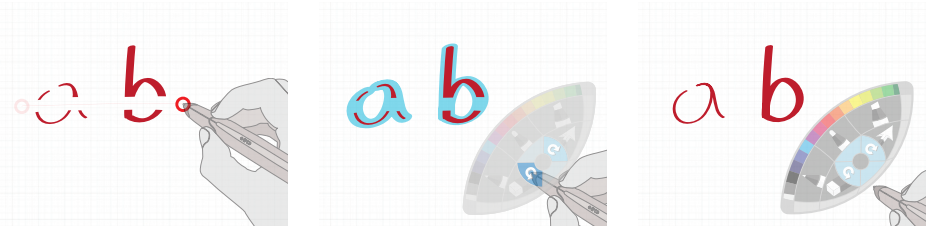
Undo einer Transformation Zeigen bei einer Outline „Motion-Arrows“ in eine bestimmte Richtung, wird der markierte Inhalt beim Aufruf von Undo in die angegebene Richtung bewegt. Je größer die Bewegung, desto länger werden auch die Pfeile. Siehe Abbildung 3.17 (c).

Werden diese drei Methoden verwendet, können alle elementaren Operationen trotz unauffälliger und einheitlicher Visualisierung ohne Probleme unterschieden werden. Eingeblendet werden können die Schrittvisualisierungen beim Drücken des Undo- bzw. Redo-Buttons durch einen Benutzer (Down-Event), solange bis der Stift wieder hochgehoben (Up-Event) wird.

3.4.2 Granularität

Die Wahl der Granularität von Aktionen stellt einen zentralen Punkt in der Gestaltung von Undo und Redo dar [1]. Nicht immer ist es sinnvoll jede elementare Operation als einzelnen Undo-Schritt zur Verfügung zu stellen. Würden zum Beispiel beim Verschieben einer Selektion 50 elementare Transformationsoperationen erzeugt (bei jedem Move-Event), müsste ein Benutzer 50-mal den Undo-Button betätigen, um die Ausgangsposition der Selektion wiederherzustellen. Dieses Beispiel zeigt, dass elementare Operationen nicht immer mit der semantischen Auffassung einer Aktion eines Benutzers korrespondieren.

Einige Rückschlüsse, wie Operationen semantisch zusammengefasst werden sollen, lassen bereits die Funktionsweisen der Werkzeuge zu. So ist es naheliegend, dass z. B. alle Einzelstriche zu einer Operation zusammengefasst werden können, die während eines einzelnen Radiervorgangs gelöscht werden. Ähnlich verhält es sich beim Verschieben mit einem Selektionswerk-



(a) (b) (c)

Abbildung 3.18: Nicht immer entspricht die Auffassung einer Aktion für einen Benutzer den elementaren Operationen (z. B. einzelnen Löschoperationen). Deswegen sollten alle elementaren Operationen, die zwischen einem Down-Event und Up-Event eines Stiftes erzeugt werden (a), zu einem Undo-Schritt (b,c) zusammengefasst werden.

zeug. Trotz einer Vielzahl von Transformationsoperationen reicht bei einem etwaigen Undo oder Redo die Rücksetzung auf den Ausgangspunkt. Generell lässt sich bei Werkzeugen abstrahieren, dass alle elementaren Operationen eines Benutzers, die zwischen Down- und Up-Event seines Stiftes erstellt werden, zu einem Undo- bzw. Redo-Schritt zusammengebündelt werden können. Ein Beispiel für eine derartige Bündelung ist in Abbildung 3.18 ersichtlich.

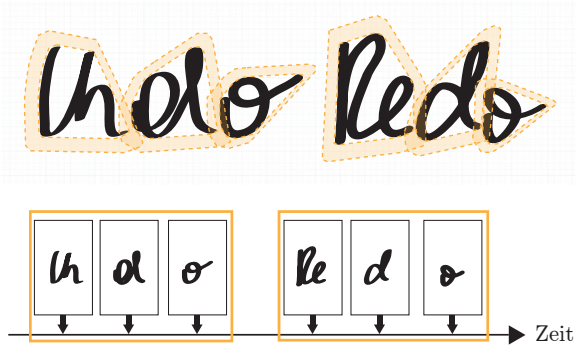
Es gibt aber noch weitere semantische Abhängigkeiten von Operationen, die von dieser Vorgehensweise nicht abgedeckt werden. So ist es zum Beispiel auch nicht sinnvoll, jeden der drei Striche eines Buchstaben „H“ als Undo-Schritt zur Verfügung zu stellen, da ein Benutzer das Schreiben des Buchstaben eher als eine zusammengehörige Aktion wahrnimmt als die Einzelstriche. Generell versuchen einige Programme, wie z. B. Microsoft Word, bei Undo-Schritten semantisch zu kapseln. Auch in Word ist nicht jeder getippte Buchstabe ein einzelner Undo-Schritt, sondern es werden mehrere Buchstaben bzw. Operationen, die innerhalb eines gewissen Zeitraumes erstellt wurden, zu einem Undo-Schritt zusammengefasst.

Zum Zweck der Bestimmung von Undo-Granularitäten wurde in dieser Arbeit ein ähnliches Verfahren entwickelt, das neben der zeitlichen Komponente auch noch die in einem grafischen Editor vorhandenen räumlichen Abhängigkeiten miteinbezieht (siehe Abbildung 3.19). Alle Operationen, die von einem Benutzer innerhalb einer kurzen Zeit mit geringen Abständen zueinander erstellt werden, werden zu einem Undo-Schritt zusammengebündelt. Somit lassen sich effektiv Wörter und zusammengehörige Inhalte erkennen, die aus einer Vielzahl an Einzelstrichen bestehen (z. B. Wörter *Undo* und *Redo* in Abbildung 3.19).

Für jede gebündelte Operation (Undo-Schritt) wird zunächst eine Operation O_{batch} gebildet, die die entsprechenden Einzeloperationen enthält. Erfolgt eine neue Operation O_{new}, so wird mit nachfolgendem Algorithmus überprüft, ob diese Operation zur gebündelten Operation hinzugefügt werden soll. Diese Entscheidung wird aufgrund des Timestamps $t(O)$ und der gleichmäßig vergrößerten konvexen Hülle $\mathcal{R}(O) = \{(x_1, y_1)| \ldots |(x_i, y_i)\}$ getroffen und kann als

$$\begin{aligned} \text{IsGranularDependent}(O_{new}, O_{batch}) := {} & |t(O_{new}) - t(O_{batch})| < \tau \wedge \\ & \mathcal{R}(O_{new}) \cap \mathcal{R}(O_{batch}) \end{aligned} \tag{3.7}$$

ausgedrückt werden. Der Schwellwert τ kann beliebig (ev. benutzerbezogen) gewählt werden. Durch Messen ergab sich als guter allgemeingültiger Wert 300 ms, d. h. zwischen zwei granular abhängigen Operationen dürfen maximal 300 ms liegen. Die konvexen Hüllen können zur Steigerung der Performance nur dann berechnet werden, wenn die zeitliche Bedingung erfüllt ist. Für einen Undo-Schritt eines Benutzers wird dann anstatt der Einzeloperationen die gebündelten Operationen verwendet (dazu mehr in Abschnitt 4.4.3).



(a)

(b)

Abbildung 3.19: Die Bestimmung der Granularität von Undo- bzw. Redo-Schritten beruht auf einer räumlichen und zeitlichen Abhängigkeit. Überschneiden sich die (vergrößerten) konvexen Hüllen der Operationen und liegen diese innerhalb eines gewissen Zeitraums (a) werden diese zu einem Schritt zusammengefasst (b).

Im Allgemeinen haben Benutzer Probleme ihre Arbeit in eindeutige Einheiten zu unterteilen [1]. Zwar ist die in diesem Abschnitt beschriebene grobe Zusammenfassung von Operationen für die Benutzer hilfreich, jedoch wäre unter Umständen das Anbieten von verschiedenen Granularitätsstufen sinnvoll. Der Benutzer kann dann wahlweise aus verschiedenen Granularitäten wählen (siehe Abschnitt 5.1.2). Ebenfalls möglich wäre ein lernendes System zu integrieren, das auf Basis der interagierenden Benutzer und deren Erwartungen die Granularitätserkennung bzw. die Anpassung der Schwellwerte vornimmt.

3.5 Zusammenfassung

Die große Herausforderung bei der Umsetzung von regionalem Undo für eine Multi-User-Anwendung ist, wie die Arbeitsbereiche der gleichzeitig arbeitenden Gruppen und Personen ermittelt werden können. Dabei zeigt dieses Kapitel verschiedene Techniken, wie regionales Undo in großflächigen grafischen Editoren umgesetzt werden kann. Die erläuterten Techniken verwenden da-

bei verschiedene Möglichkeiten der Regionsbestimmung. Folgende Techniken wurden vorgestellt:

Selektionstechnik: Jener Bereich, der von einem Selektionswerkzeug (beispielsweise Lasso) definiert wird, wird als Region für Undo und Redo genutzt. Alle Operationen, die in den selektierten Bereich fallen, sind von Undo und Redo betroffen.

Clustering-Technik: Es wird versucht, dass zusammengehörige Inhalte bestmöglich in Regionen zusammengefasst werden, welche dem Benutzer dann für Undo und Redo zur Verfügung gestellt werden. Die aktuelle Region wird über die Position des Benutzermenüs bestimmt.

Guideline-Technik: Nicht der Inhalt bestimmt die Region, sondern ein festgelegter Bereich links und rechts vom Benutzermenü. Die in der Clustering-Technik gebildeten Cluster werden zur intelligenten Anpassung der Benutzerbereiche verwendet.

Proximity-Technik: Ermittelt das Blickfeld einer Person aufgrund seiner Position und der Entfernung zum Whiteboard. Das Blickfeld wird als Region für Undo und Redo verwendet.

Jedes der erläuterten Konzepte hat seine Stärken und Schwächen. So kann mit der Selektionstechnik eine sehr einfache Einbindung von regionalem Undo in eine Anwendung erfolgen, obwohl der zusätzliche Mehraufwand der Selektion bei jedem Undo-Aufruf für den Benutzer ein großes Problem darstellt. Aus diesem Grund wäre dies nur als Ergänzung zu einem anderen Konzept denkbar. Daraus folgt die Notwendigkeit einer automatischen Bestimmung der Regionen, die mit den anderen drei Verfahren gewährleistet ist. Die Guideline-Technik stellt aufgrund ihrer Flexibilität und der automatischen Anpassung der Bereiche ein sehr praktisches und benutzerfreundliches Konzept dar, wonach dieses Verfahren der starren Clustering-Technik vorzuziehen ist. Das Problem, dass nicht immer eine geeignete Abstraktion der Position der Benutzer (z. B. durch Benutzermenü) zur Verfügung steht, kann nur durch die Proximity-Technik gelöst werden. Grundsätzlich ist die Proximity-Technik die intuitivste und universellste aller präsentierten Lösungen. Jedoch kann die Proximity-Technik gerade auf wenig erfahrene Whiteboard-Benutzer auch gewöhnungsbedürftig wirken, weil sie es nicht gewohnt sind, Programmteile wie den Undo-Geltungsbereich mittels ihrer Position bzw. ihres Blickfeldes zu steuern. Generell wäre, entgegen dem in dieser Arbeit vorgestellten Verfahren, eine Tracking-Methode wünschenswert, die auch Kopfneigung und -drehung miteinbezieht und so ein exaktes Blickfeld ermittelt. Dies würde die Intuitivität einer solchen Lösung weiter erhöhen und vielleicht eine zusätzliche Darstellung oder Abstraktion des aktuell wahrgenommenen Blickfeldes des Benutzers obsolet machen. Weitere Konzepte wie die Darstellung des nächsten Undo- bzw. Redo-Schrittes oder der Zusammenfassung von zusammengehörigen Operationen erleichtern den Nutzern den (ungewohnten) Umgang mit regionalem Undo.

Kapitel 4

Implementierung

Um regionales Undo in einer Multi-User-Anwendung technisch zu ermöglichen, ist ein spezielles und umfangreiches Undo-Modell erforderlich. Welches Undo-Modell für eine entsprechende Funktionalität notwendig ist und wie ein entsprechendes Framework und dessen Integration in einen grafischen Editor gestaltet werden kann, beschreiben die nachfolgenden Abschnitte.

4.1 Anforderungen

Folgende Anforderungen muss ein entsprechendes Undo-Modell bzw. Undo-Framework für regionales Undo in Hinblick auf die technische Umsetzung erfüllen [7]:

Konsistenz: Undo soll jeden Dokumentstatus wiederherstellen, der ohne der rückgängig gemachten Operation vorherrschte, so als ob diese Operation nie ausgeführt worden wäre.

Generik: Der Ansatz eines Multi-User-Undo sollte so generisch gestaltet sein, dass er auch in anderen Anwendungen einsetzbar ist.

Undo von beliebigen Operationen: Jede Operation eines Benutzers muss zu jedem Zeitpunkt rückgängig machbar bzw. wiederherstellbar sein.

4.2 Auswahl des Undo-Modells

Der erste Schritt in der Implementierung eines regionalen Undos betrifft die Umsetzung des zugrunde liegenden Undo-Modells. Ein rein lineares Modell, wie es beispielsweise in den meisten Desktop-Anwendungen realisiert ist und welches beim Aufruf von Undo auch immer alle nachfolgenden Operationen rückgängig macht (siehe Abschnitt 1.1), reicht für ein regionales Undo nicht aus. Das Problem besteht darin, dass mehrere Benutzer simultan arbeiten können und sich somit Operationen der Benutzer im zeitlichen Verlauf

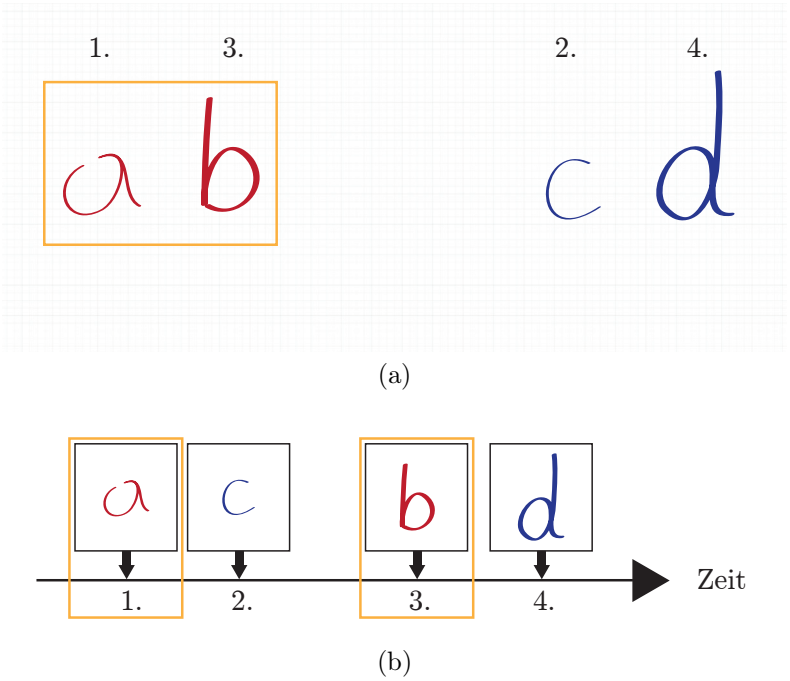


Abbildung 4.1: Fügen zwei Benutzer (rot und blau) abwechselnd Inhalte in ein gemeinsames Dokument ein (a), so werden diese Operationen auch chronologisch abwechselnd im Verlauf abgelegt (b). Ein regionales Undo entspricht dann dem selektiven Rückgängigmachen von Operationen aus dem Verlauf.

abwechseln. Um nur Arbeitsschritte eines Benutzers oder innerhalb einer Region rückgängig zu machen, müssen also Operationen *selektiv* aus dem Verlauf rückgängig gemacht werden, ohne dass nachfolgende Operationen ebenso rückgängig gemacht werden. Dafür müssen Operationen aus der Vergangenheit so rückgängig gemacht werden und der resultierende Dokumentstatus derart verändert werden, als hätte diese Operation nie stattgefunden. Fügen zum Beispiel zwei getrennt arbeitende Benutzer in einem Dokument abwechselnd Inhalte ein, so werden diese auch in der Verlauf-Liste jeweils chronologisch abwechselnd abgelegt. Will jetzt ein Benutzer Arbeitsschritte „seiner“ Region rückgängig machen, so müssen diese Schritte selektiv aus dem linearen Verlauf entfernt werden (siehe Abbildung 4.1). Um also regionales Undo technisch zu ermöglichen, bedarf es einer selektiven Undo-Lösung, die jede Operation zu jedem Zeitpunkt in beliebiger Reihenfolge rückgängig machen und wiederherstellen können muss.

Die Schwierigkeit dabei ist, dass sich nachfolgende Operationen in der Regel auf die vorhergehenden beziehen. Neben einer möglichen semantischen Abhängigkeit besteht bei Auflistungen von beispielsweise Buchstaben (Text-

dokument) oder Einzelstrichen (Zeichnung) möglicherweise eine Abhängigkeit durch Indizes. Dieses Problem kann am folgenden Beispiel veranschaulicht werden: Demzufolge fügen vier Operationen O_1, O_2, O_3 und O_4 nacheinander die Einzelstriche aus Abbildung 4.1 in das Datenmodell (Liste) ein:

$$\begin{aligned} O_1 &= \mathrm{Ins}(a, 0) \ , \\ O_2 &= \mathrm{Ins}(c, 1) \ , \\ O_3 &= \mathrm{Ins}(b, 2) \ , \\ O_4 &= \mathrm{Ins}(d, 3) \ . \end{aligned} \tag{4.1}$$

O_1 fügt sein Element an Indexposition 0 und O_2, O_3 sowie O_4 aufgrund der chronologischen Abfolge an den nachfolgenden Indexpositionen ein. Der Inhalt des Datenmodells würde dann $D = \{a, c, b, d\}$ lauten (siehe Abbildung 4.1 (b)). Wird jetzt die Operation O_1 rückgängig gemacht, so ergibt sich durch die inverse Operation O_1, ausgedrückt durch

$$\mathrm{Undo}(O_1) = \overline{O_1} = \mathrm{Del}(a, 0) \ , \tag{4.2}$$

das Entfernen des Elements a an Indexposition 0 und das daraus resultierende Datenmodell $D = \{c, b, d\}$. Würde anschließend beispielsweise Operation O_3 rückgängig gemacht, ergibt sich durch die inverse Operation von O_3, ausgedrückt durch

$$\mathrm{Undo}(O_3) = \overline{O_3} = \mathrm{Del}(b, 2) \ , \tag{4.3}$$

das Entfernen von Element b an der Indexposition 2. Das Element b befindet sich aber im aktuellen Datenmodell $D = \{c, b, d\}$ nicht mehr an Indexposition 2, sondern aufgrund der rückgängig gemachten Operation O_1 an Indexposition 1. Somit kommt es bereits bei diesem einfachen Beispiel zu einem inkonsistenten Verlauf (Konflikt).

Um dieser Inkonsistenz vorzubeugen wird ein Algorithmus benötigt, der den Verlauf zu jedem Zeitpunkt konsistent hält, egal in welcher Reihenfolge Operationen rückgängig gemacht oder wiederhergestellt werden. In dem eben genannten Beispiel müssten also alle nachfolgenden Operationen von O_1, darunter Operation O_3, aufgrund der Auswirkungen der rückgängig gemachten Operation O_1 *transformiert* werden. Das Löschen von a an der Indexposition 0 würde damit bei Operation O_3 zur Verminderung des Index führen und somit

$$\begin{aligned} O_3' &= \mathrm{Ins}(b, 1) \text{ bzw.} \\ \mathrm{Undo}(O_3') &= \overline{O_3}' = \mathrm{Del}(b, 1) \end{aligned} \tag{4.4}$$

ergeben. Somit würde beim Aufruf von Undo von O_3' das Element b im Datenmodell $D = \{c, b, d\}$ an der richtigen Indexposition 1 gelöscht werden.

Ein solches Verfahren wird in der Fachsprache als *Operational Transformation*[1] (OT) bezeichnet und wurde dazu entwickelt, um in einer verteilten Anwendung auch bei gleichzeitig erstellten Operationen verschiedener Clients ein konsistentes Datenmodell auf jedem Client zu erzeugen.

In dem genannten Beispiel ist Operational Transformation trivial, jedoch steigt die Komplexität und Anzahl von Sonder- und Konfliktfällen mit der Anzahl von verschachtelten Operationen stark an. Der nachfolgende Abschnitt zeigt die Implementierung des OT-Algorithmus, der in dieser Arbeit realisiert wurde.

4.3 Undo-Algorithmus

In dieser Arbeit wurde der auf Operational Transformation basierende Algorithmus *AnyUndo* von Sun et al. [24] implementiert. *AnyUndo* wurde ursprünglich für verteilte Texteditoren konzipiert und garantiert im Gegensatz zu anderen Lösungen [19–21], dass zu jedem Zeitpunkt jede Operation rückgängig gemacht werden kann.

Eine wesentliche Aufgabe kommt Operational Transformation in der Erhaltung der Konsistenz der Datenmodelle von verschiedenen Clients zu, sodass mehrere Benutzer auf verschiedenen Clients in einer beliebigen Reihenfolge (mitunter auch gleichzeitig) Operationen erstellen können, die aber auf jedem Client zu einem konsistenten und gleichen Datenmodell führen. Dabei ist unter Umständen eine Konfliktbehandlung notwendig, die Konflikte zwischen gleichzeitig erstellten und auf den gleichen Inhalt verweisenden Operationen auflöst. In einer nicht verteilten Multi-User-Anwendung haben Probleme, die die Sicherung eines konsistenten Datenmodells auf *verschiedenen* Clients betreffen, keine Relevanz, da bei nicht verteilten Anwendungen eine eindeutige kausale Abfolge der Operationen der Benutzer gegeben ist. Nichtsdestoweniger bleibt die zugrunde liegende Index- und Transformationsproblematik beim Durchführen von Undo oder Redo beliebiger Operationen dieselbe (wie in Abschnitt 4.2 beschrieben).

Die Hauptaufgabe des Algorithmus in dieser Arbeit ist somit die Sicherstellung eines konsistenten Verlaufs und Datenmodells zu jedem Zeitpunkt, wenn Operationen in einer beliebigen Reihenfolge rückgängig gemacht oder wiederhergestellt werden.

4.3.1 Operational Transformation

Wie bereits erwähnt sorgt Operation Transformation im Falle eines Undo-Aufrufs dafür, dass die Auswirkungen der rückgängig gemachten Operation auf das Dokument und auf andere Operationen (Erhöhung und Verminderung der Indizes) eliminiert werden. Mit anderen Worten verändert Opera-

[1] http://cooffice.ntu.edu.sg/otfaq/

tional Transformation das Datenmodell und die Operationen so, wie wenn die rückgängig gemachte Operation nie existiert hätte.

Transformationsfunktionen

Operational Transformation bezieht sich zumeist auf *Einfüge- und Löschoperationen* in eine Liste (Datenmodell). Mit diesen elementaren Funktionen lassen sich wiederum andere Funktionen wie z. B. Ersetzen abbilden. Einfügeoperationen fügen Inhalte an eine bestimmte Indexposition im Datenmodell ein, während Löschoperationen Inhalte von bestimmten Indizes löschen. Wird beispielsweise von einer Indexposition etwas gelöscht, müssen die Indizes der Operationen, die auf nachfolgende Indexpositionen verweisen, aktualisiert und folglich um eins vermindert werden. Diese Auswirkungen von Einfüge- und Löschoperationen aufeinander werden in OT mittels Transformationsfunktionen abgebildet. Der Algorithmus *AnyUndo* nutzt dabei folgende Transformationsfunktionen [26]:

- **IT** (**Inclusion** Transformation)
- **ET** (**Exclusion** Transformation)

$O_1' = \mathrm{IT}(O_1, O_2)$ transformiert O_1 gegenüber O_2 so, dass die Auswirkungen von O_2 garantiert in die resultierende Operation O_1' **inkludiert** sind. $O_1' = \mathrm{ET}(O_1, O_2)$ transformiert O_1 gegenüber O_2 hingegen so, dass die Auswirkungen von O_2 garantiert aus der resultierenden Operation O_1' **exkludiert** sind. Bei Auswirkungen handelt es sich im Allgemeinen um Indexänderungen in einer Operation, die durch andere Operationen bedingt werden.

Fügt beispielsweise eine Einfügeoperation O_2 einen Inhalt vor einer anderen Einfügeoperation O_1 in das Datenmodell ein, so würde $\mathrm{IT}(O_1, O_2)$ den Index der ersten Operation O_1 um eins erhöhen. Die Auswirkungen von O_2 sind dann in O_1 inkludiert. $\mathrm{ET}(O_1, O_2)$ würde hingegen den Index der ersten Operation O_1 um eins vermindern, die Auswirkungen von O_2 wären dann von O_1 exkludiert. Vereinfacht gesagt werden bei IT und ET die Indizes je nach beteiligten Operationen und deren Indizes verändert. In Anhang A sind die Definitionen aller möglichen Kombinationen von Einfüge- und Löschoperationen inklusive von Beispielen angeführt.

Eine weitere wichtige Anforderung an die Transformationsfunktion ist die Umkehrbarkeit. Für jede Transformationsfunktion muss gelten:

$$\begin{aligned} O_1' &= \mathrm{IT}(O_1, O_2) \text{ und} \\ O_1 &= \mathrm{ET}(O_1', O_2)\ . \end{aligned} \tag{4.5}$$

LIT und LTranspose

Der Algorithmus *AnyUndo* verwendet noch zwei weitere Hilfsfunktionen aus Operational Transformation [26]. $\mathbf{LIT}(O, L)$ transformiert O gegen eine Liste aus Operationen L so, dass die Auswirkungen von allen Operationen aus der

Liste in der resultierenden Operation O' inkludiert sind (siehe Algorithmus 4.1). In Abbildung 4.2 (a) ist das Verhalten des Algorithmus dargestellt.

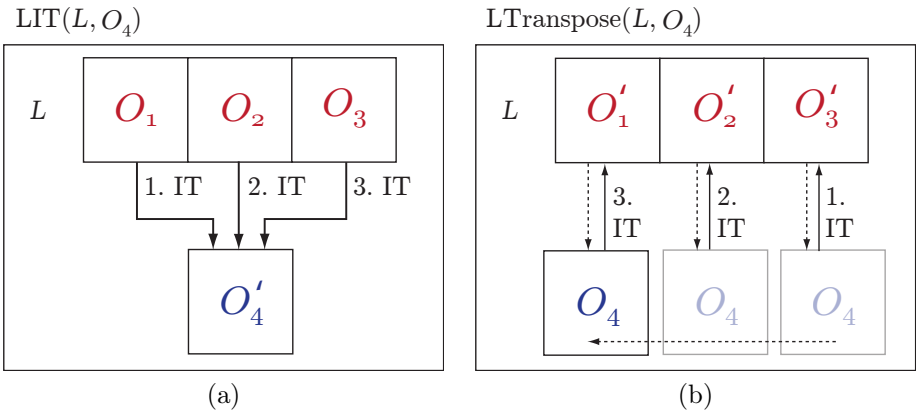
Algorithmus 4.1: LIT

```
1: LIT(O, L)
2:     O' = O;
3:     for i = 0 ... sizeof(L) do
4:         O' = IT(O', L[i]);
5:     end for
6: end
```

LTranspose(L, O) hingegen verändert die Operationen in L so, als ob eine Operation O vom Ende der Liste L an den Anfang der Liste verschoben würde. Die Operationen der Liste L enthalten dann effektiv die Auswirkungen der Operation O, wobei O kein Element von L ist (siehe Algorithmus 4.2). In Abbildung 4.2 (b) ist das Verhalten des Algorithmus dargestellt.

Algorithmus 4.2: LTranspose

```
1: LTranspose(L, O)
2:     for i = sizeof(L) − 1 ... 0 do
3:         O = ET(O, L[i]);
4:         L[i] = IT(L[i], O);
5:     end for
6: end
```



(a) (b)

Abbildung 4.2: Dargestelltes Verhalten der Standardfunktionen LIT (a) und LTranspose (b) bei einer Liste von Operationen L und einer Operation O_4.

Angenommen es gibt eine Liste von Operationen $L = \{O_1, O_2, O_3\}$. Würde jetzt bei einem Undo die inverse Funktion von O_1 gebildet, kann sie mittels LTranspose($\{O_2, O_3\}, \overline{O_1}$) so verschoben werden, dass O_2 und O_3 die Auswirkungen der inversen Operation $\overline{O_1}$ effektiv beinhalten. In einem solchen Fall würden O_2' und O_3' neben den Auswirkungen von O_1 auch die Auswirkungen der inversen Operation $\overline{O_1}$ enthalten, welche gemeinsam eine Einheitsoperation bilden und sich aufheben. Für O_2' und O_3' scheint es somit, als ob es O_1 oder die inverse Operation $\overline{O_1}$ nie gegeben hätte. Damit kann O_1 einfach als Einheitsoperation O_1* markiert werden. Eine als Einheitspaar bzw. Undo-Paar markierte Operation hat keinerlei Auswirkungen auf andere Operationen, während die Auswirkungen von anderen Operationen trotzdem weiterhin (für den Falle eines Redo-Aufrufs) in die markierte Einheitsoperation miteinfließen. Für die Transformationsfunktionen und Undo-Paare gilt:

$$\begin{aligned} &T(O_1, O_2*) = O_1 \text{ und} \\ &T(O_1*, O_2) = T(O_1, O_2) * \ . \end{aligned} \tag{4.6}$$

Der große Vorteil einer derartigen Lösung ist, dass bei einem Undo oder Redo keine Operationen aus dem Verlauf hinzugefügt oder entfernt werden müssen. So werden etwaige Verweise oder zusätzliche komplexe Datenstrukturen vermieden. Bei einem Undo werden die nachfolgenden Operationen lediglich so verändert, als ob es die rückgängig gemachte Operation nie gegeben hätte.

4.3.2 AnyUndo

In den nachfolgenden Abschnitten wird ein Verlauf von Operationen als *HB* bezeichnet. Das Datenmodell, also im Falle dieser Arbeit die Liste der im grafischen Editor gezeichneten Einzelstriche, wird als D bezeichnet.

Erfolgt jetzt beispielsweise für eine ausgewählte Operation O_i aus dem Verlauf $HB = \{O_1, O_2, \ldots, O_i, \ldots, O_n\}$ ein Undo, so wird Algorithmus 4.3 ausgeführt.

Algorithmus 4.3: AnyUndo

```
1: AnyUndo(O_i, HB)
2:     O_i' = makeInverse(O_i)              ▷ Inverse Operation bilden
3:     O_i'' = LIT(O_i', HB[i+1,n])         ▷ O_i' transformieren
4:     Execute(O_i'')                       ▷ O_i'' ausführen (z. B. Einfügen)
5:     LTranspose(HB[i+1,n], O_i'')         ▷ O_i'' verschieben
6:     O_i* = Mark(O_i)                     ▷ Undo-Paar markieren
7: end
```

Dabei wird im ersten Schritt von der identifizierten Operation O_i die inverse Operation gebildet. Bei einer Einfügeoperation an einer bestimmten Indexposition wäre das beispielsweise die Löschoperation mit der gleichen Indexposition. Im zweiten Schritt wird O_i aufgrund aller nachfolgenden

Operationen in *HB* transformiert. Es werden also alle Effekte von allen Operationen, die O_i in *HB* nachfolgen, in die inverse Operation $\overline{O_i}$ inkludiert, da es vorkommen kann, dass nachfolgende Operationen den Index von O_i im Datenmodell bereits wieder verändert haben. Ist dies erfolgt, so kann die transformierte inverse Operation $\overline{O_i}'$ auf das aktuelle Datenmodell ausgeführt werden (z. B. Einfügen in das Datenmodell an der jeweiligen Indexposition). Anschließend werden alle nachfolgenden Operationen von O_i mittels LTranspose so verändert, als ob die inverse transformierte Operation $\overline{O_i}'$ vom Ende von *HB* so weit nach links verschoben würde, dass sie direkt an die ursprüngliche Funktion O_i angrenzt und mit ihr ein Undo-Paar bildet. Die Wirkung der beiden Operationen hebt sich somit auf und hat auf die nachfolgenden Operationen keinen Einfluss mehr. Für die nachfolgenden Operationen von O_i scheint es, als hätte es O_i nie gegeben. Anschließend wird die ursprüngliche Funktion als O_i* markiert, damit sie keine Auswirkungen mehr auf folgende Transformationen hat.

Ein Redo einer Operation kann durch Aufrufen des gleichen Algorithmus mit einer als Undo-Paar markierten Operation O_i* erreicht werden. Die inverse Operation einer als Undo-Paar markierten Operation ist per Definition wieder die ursprüngliche Operation. Anschließend werden wiederum alle Schritte des Algorithmus durchgeführt, wobei im letzten Schritt die Operation wieder als nicht rückgängig gemacht markiert wird:

$$\text{Mark}(O*) = O \ . \tag{4.7}$$

4.3.3 Beispiel

Der eben beschriebene Algorithmus wird im folgenden Abschnitt am Beispiel aus Abschnitt 4.2 erläutert. Um die nachfolgende formale Beschreibung des *AnyUndo*-Algorithmus besser zu verstehen, empfiehlt es sich vorab einen Blick auf Abbildung 4.3 zu werfen, welche die durch den Algorithmus verursachten Indexänderungen der Operationen und des Datenmodells darstellt.

Zunächst fügten vier Einfügeoperationen nacheinander Inhalte in das Datenmodell ein:

$$\begin{aligned} O_1 &= \text{Ins}(a,0) \quad D = \{a\} \ , \\ O_2 &= \text{Ins}(c,1) \quad D = \{a,c\} \ , \\ O_3 &= \text{Ins}(b,2) \quad D = \{a,c,b\} \ , \\ O_4 &= \text{Ins}(d,3) \quad D = \{a,c,b,d\} \ . \end{aligned} \tag{4.8}$$

Das Datenmodell besteht nach dem Ausführen aller vier Operationen aus $D = \{a,c,b,d\}$ und der Verlauf aus $HB = \{O_1, O_2, O_3, O_4\}$. Die nachfolgenden Tabellen zeigen das Verhalten des Algorithmus bei der Ausführung. Tabelle 4.1 zeigt die Ausführung von *AnyUndo* beim Rückgängigmachen von O_1, Tabelle 4.2 die Ausführung beim Rückgängigmachen von O_3 und Tabelle 4.3 die Ausführung beim Wiederherstellen von O_1.

Tabelle 4.1: Ausführung des Algorithmus beim Rückgängigmachen von O_1

Undo(O_1)	
Inverse bilden	$\text{Undo}(O_1) = \overline{O_1} = \text{Del}(a, 0)$
$\overline{O_1}$ transformieren	$\overline{O_1}' = \text{LIT}(\overline{O_1}, \{O_2, O_3, O_4\})$
$\overline{O_1}'$ ausführen	$\overline{O_1}' = \text{Del}(a, 0)$ um $D = \{c, b, d\}$ zu erhalten
$\overline{O_1}'$ verschieben	$\text{LTranspose}(\{O2, O3, O4\}, \overline{O_1}')$ um $HB = \{O_1, O_2', O_3', O_4'\}$ zu erhalten
Undo-Paar markieren	$O_1* = \text{Mark}(O_1)$ um $HB = \{O_1*, O_2', O_3', O_4'\}$ zu erhalten

Tabelle 4.2: Ausführung des Algorithmus beim Rückgängigmachen von O_3

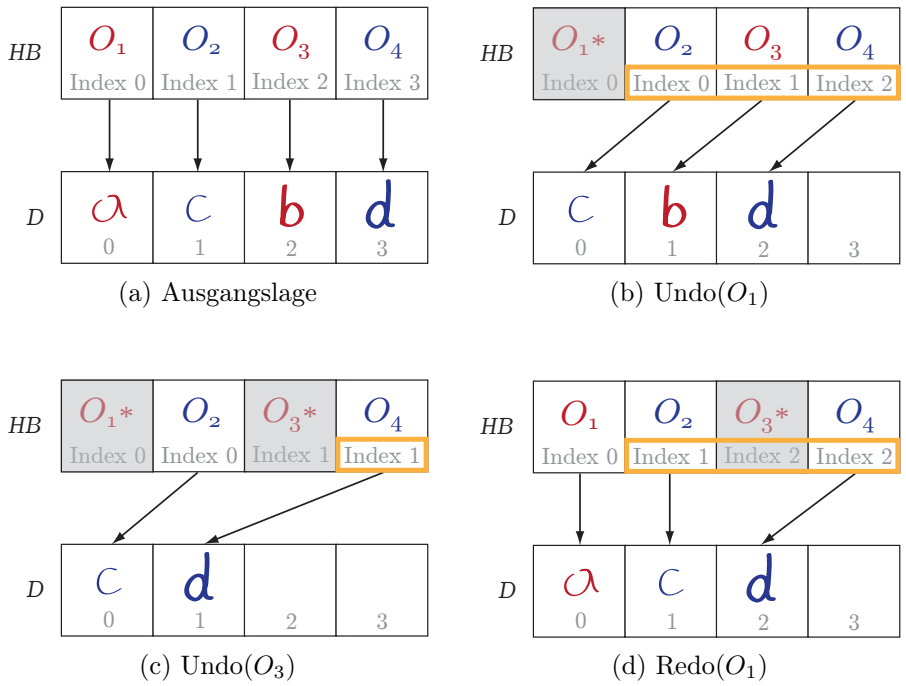
Undo(O_3)	
Inverse bilden	$\text{Undo}(O_3) = \overline{O_3} = \text{Del}(b, 1)$
$\overline{O_3}$ transformieren	$\overline{O_3}' = \text{LIT}(\overline{O_3}, \{O_4'\})$
$\overline{O_3}'$ ausführen	$\overline{O_3}' = \text{Del}(b, 1)$ um $D = \{c, d\}$ zu erhalten
$\overline{O_3}'$ verschieben	$\text{LTranspose}(\{O4'\}, \overline{O_3}')$ um $HB = \{O_1*, O_2', O_3', O_4''\}$ zu erhalten
Undo-Paar markieren	$O_1* = \text{Mark}(O_3)$ um $HB = \{O_1*, O_2', O_3'*, O_4''\}$ zu erhalten

Tabelle 4.3: Ausführung des Algorithmus beim Wiederherstellen von O_1

Redo(O_1)	
Inverse bilden	$\text{Undo}(O_1*) = \overline{\overline{O_1}} = O_1 = \text{Ins}(a, 0)$
O_1 transformieren	$O_1' = \text{LIT}(O_1, \{O_2', O_3'*, O_4''\})$
O_1' ausführen	$O_1' = \text{Ins}(a, 0)$ um $D = \{a, c, d\}$ zu erhalten
O_1' verschieben	$\text{LTranspose}(\{O_2', O_3'*, O_4''\}, O_1')$ um $HB = \{O_1*, O_2, O_3*, O_4'\}$ zu erhalten
Undo-Paar markieren	$O_1 = \text{Mark}(O_1*)$ um $HB = \{O_1, O_2, O_3*, O_4'\}$ zu erhalten

Resultierende Änderungen

Die nachfolgenden Abschnitte erläutern die konkreten Indexänderungen, die die formale Beschreibung der drei verschiedenen Fälle an Datenmodell sowie den Operationen in *HB* vorgenommen hat (siehe auch Abbildung 4.3).



(a) Ausgangslage (b) Undo(O_1)

(c) Undo(O_3) (d) Redo(O_1)

Abbildung 4.3: Diese Abbildung zeigt die Änderungen, die der Algorithmus an Datenmodell und an Indizes der Operationen im beschriebenen Beispiel vornimmt. (a) zeigt die Ausgangslage, (b) die Situation nach dem Undo von O_1, (c) die Situation nach einem Undo von O_3 und (d) die Situation nach einem Redo von O_1.

Undo(O_1) Die inverse Operation $\overline{O_1}$ ergibt das Löschen von Element a an Indexposition 0. Weil keine der nachfolgenden Operationen indexmäßig *vor* der Operation O_1 einfügt, wird die inverse transformierte Operation $\overline{O_1}'$ durch LIT nicht verändert. Anschließend wird $\overline{O_1}'$ ausgeführt und Element a aus dem Datenmodell entfernt:

$$\overline{O_1}' = \mathrm{Del}(a, 0) \quad D = \{c, b, d\} \ . \tag{4.9}$$

Da die nachfolgenden Operationen O_2, O_3 und O_4 indexmäßig *nach* der (rückgängig gemachten) Operation O_1 einfügen, vermindert LTranspose die Indizes dieser Operationen:

$$\begin{aligned} O_1* &= \mathrm{Ins}(a, 0) \ , \\ O_2' &= \mathrm{Ins}(c, \mathbf{0}) \ , \\ O_3' &= \mathrm{Ins}(b, \mathbf{1}) \ , \\ O_4' &= \mathrm{Ins}(d, \mathbf{2}) \ . \end{aligned} \tag{4.10}$$

Undo(O_3) Die Bildung der inversen Operation $\overline{O_3}$ ergebe hier das Löschen von Element b bei Index 1. Da auch hier keine nachfolgenden Operationen die Indexpositionen vor O_3 beeinflussen, findet auch hier keine Transformation durch LIT statt. Die transformierte inverse Operation $\overline{O_3}'$ entfernt somit Element b aus dem Datenmodell:

$$\overline{O_3}' = \mathrm{Del}(b, 1) \quad D = \{c, d\} \ . \tag{4.11}$$

Weil O_3 indexmäßig vor der einzigen nachfolgenden Operation O_4 kommt, wird O_4 durch LTranspose ein weiteres Mal um eins vermindert:

$$\begin{aligned} O_1* &= \mathrm{Ins}(a, 0) \ , \\ O_2' &= \mathrm{Ins}(c, 0) \ , \\ O_3'* &= \mathrm{Ins}(b, 1) \ , \\ O_4'' &= \mathrm{Ins}(d, \mathbf{1}) \ . \end{aligned} \tag{4.12}$$

Redo(O_1) Per Definition ist die inverse Funktion einer als rückgängig gemachten Operation wieder die Operation selbst. Weil keine Operationen auf Indexpositionen vor O_1 verweisen, wirkt sich LIT nicht weiter aus. Somit fügt die ursprüngliche, transformierte Operation das Element a wieder an Indexposition 0 ein:

$$O_1' = \mathrm{Ins}(a, 0) \quad D = \{a, c, d\} \ . \tag{4.13}$$

Da die nachfolgenden Operationen O_2, O_3 und O_4 indexmäßig *nach* Operation O_1 kommen, erhöht LTranspose die Indizes der nachfolgenden Operationen wieder um eins:

$$\begin{aligned} O_1 &= \mathrm{Ins}(a, 0) \ , \\ O_2 &= \mathrm{Ins}(c, \mathbf{1}) \ , \\ O_3* &= \mathrm{Ins}(b, \mathbf{2}) \ , \\ O_4' &= \mathrm{Ins}(d, \mathbf{2}) \ . \end{aligned} \tag{4.14}$$

4.3.4 Sonderfälle

Bei dem genannten Algorithmus kann es zu zahlreichen Sonderfällen kommen, die speziell behandelt werden müssen. Die zwei wichtigsten werden in den nachfolgenden Abschnitten genauer erläutert.

Einheitsoperationen

Es kann vorkommen, dass eine Einfüge- und Löschoperationen auf den gleichen Inhalt im Datenmodell operieren und deshalb zueinander in Konflikt stehen. Dies kann beispielsweise auftreten, wenn ein Benutzer einen Einzelstrich zeichnet (Einfügeoperation O_1) und ein anderer Benutzer diesen mit

einem Löschwerkzeug wieder entfernt (Löschoperation O_2). In einem solchen Fall bilden O_2 und O_1 eine Einheitsoperation, die durch eine Abhängigkeit in Form $O_2 \rightarrow O_1$ ausgedrückt wird. O_2 bezieht sich also immer auf ein Datenmodell, das von Operation O_1 beeinflusst wurde.

Werden jetzt beide Operationen rückgängig gemacht, können die Transformationsfunktionen beim Aufruf der Funktion LTranspose keinen validen Index mehr für O_2 liefern, weil der Status des Datenmodells bzw. die Operation O_1, auf die sich O_2 bezieht, nicht mehr existiert. Somit muss O_2 als „nicht valide Einheitsoperation“ gekennzeichnet werden und kann damit auch nicht mehr wiederhergestellt werden. Aufgrund des nicht validen Zustandes bzw. der nicht mehr vorhandenen Indexposition von O_2 können auch keine weiteren Transformationen auf dieser Operation ausgeführt werden. Erst wenn O_1 und damit der Zustand des Datenmodells, auf den sich O_2 bezieht, wiederhergestellt wird, gibt es auch für O_2 wieder einen validen Zustand bzw. eine richtige Indexposition. Diese entspricht dem aktuellen Index von O_1, an der die Operation den Inhalt im Zuge des Redo-Aufrufes wieder eingefügt hat.

Mehrdeutige Referenzen

Ein weiteres Problem bei *AnyUndo* ist das Auftreten mehrdeutiger Referenzen, die Sun et al. in ihrer Arbeit zwar erwähnt, jedoch nicht gelöst haben. Mehrdeutige Referenzen entstehen bei mehreren Operationen, die an dieselbe Indexposition einfügen oder löschen. Beispielsweise könnte durch ein Radierwerkzeug oder bei der Nutzung von Ebenen (in Grafikprogrammen üblich) der Fall eintreten, dass eine Operation O_2 einen Inhalt vor den Inhalt einer früheren Operation O_1 in das Datenmodell einfügt. Das Problem ist, dass durch Anwenden des *AnyUndo*-Algorithmus die Information, dass O_2 ein Element ursprünglich vor O_1 in das Datenmodell eingefügt hat, verloren gehen kann und folglich ein falsches Datenmodell erzeugt werden kann.

Das Problem kann mit folgendem Beispiel veranschaulicht werden: Zwei Operationen O_1 und O_2 fügen a und b jeweils an Indexposition 0 in das Datenmodell ein:

$$\begin{aligned} O_1 &= \mathrm{Ins}(a, 0) \;\; D = \{a\} \;\; , \\ O_2 &= \mathrm{Ins}(b, 0) \;\; D = \{b, a\} \;\; . \end{aligned} \tag{4.15}$$

Nach der Ausführung der Operationen O_1 und O_2 besteht das Datenmodell aus $D = \{b, a\}$, weil die nachfolgende Operation O_2 an die gleiche Indexposition wie O_1 einfügte und das Element a somit um eine Position nach hinten verschoben wurde.

Undo(O_2) Wird O_2 rückgängig gemacht, ergibt die inverse Operation das Löschen von b bei Index 0 (LIT hat keine Auswirkung):

$$\text{Undo}(O_2) = \text{Del}(b, 0) \;\; D = \{a\} \;\; . \tag{4.16}$$

Da im Verlauf keine Operationen der rückgängig gemachten Operation O_2 nachfolgen, hat LTranspose keine Auswirkung:

$$\begin{aligned} O_1 &= \text{Ins}(a, 0) \;, \\ O_2* &= \text{Ins}(b, 0) \;. \end{aligned} \tag{4.17}$$

Undo(O_1) Beim Rückgängigmachen von O_1 wird a an der Indexposition 0 gelöscht (LIT hat keine Auswirkung):

$$\text{Undo}(O_1) = \text{Del}(a, 0) \;\; D = \{\} \;\; . \tag{4.18}$$

LTranspose hat bei den beiden Operationen im Verlauf ebenfalls keine Auswirkung:

$$\begin{aligned} O_1* &= \text{Ins}(a, 0) \;, \\ O_2'* &= \text{Ins}(b, 0) \;. \end{aligned} \tag{4.19}$$

Redo(O_1) Das Wiederherstellen von O_1 führt dazu, dass a wieder bei Index 0 in das Datenmodell eingefügt wird (LIT hat keine Auswirkung):

$$\text{Undo}(O_1*) = \text{Ins}(a, 0) \;\; D = \{a\} \;\; . \tag{4.20}$$

Aufgrund des LTranspose-Aufrufs kommt es jetzt zu $\text{IT}(O_2*, O_1)$, also der Einbeziehung der Einfügeoperation O_1 an Indexposition 0 in die Operation O_2* bei Index 0. Dieser Aufruf führt zu einer Erhöhung des Index von O_2*, da eine Einfügung an die gleiche Indexposition dazu führt, dass sich alle anderen Inhalte ab dieser Indexposition um eine Indexposition nach hinten verschieben:

$$\begin{aligned} O_1 &= \text{Ins}(a, 0) \;, \\ O_2'* &= \text{Ins}(b, \mathbf{1}) \;. \end{aligned} \tag{4.21}$$

Redo(O_2) Das Problem ist jetzt jedoch, dass beim Wiederherstellen von O_2 und dem damit verbundenen Einfügen an Indexposition 1 ein falsches Datenmodell (gegenüber der Ausgangslage) erzeugt wird:

$$\text{Undo}(O_2'*) = \text{Ins}(b, 1) \;\; D = \{a, b\} \;\; . \tag{4.22}$$

Der Verlauf wird durch LTranspose beim Wiederherstellen von O_2 nicht weiter verändert, jedoch stimmen die wiederhergestellten Indizes nicht mehr mit der Ausgangslage überein:

$$\begin{aligned} O_1 &= \text{Ins}(a, 0) \;, \\ O_2 &= \text{Ins}(b, \mathbf{1}) \;. \end{aligned} \tag{4.23}$$

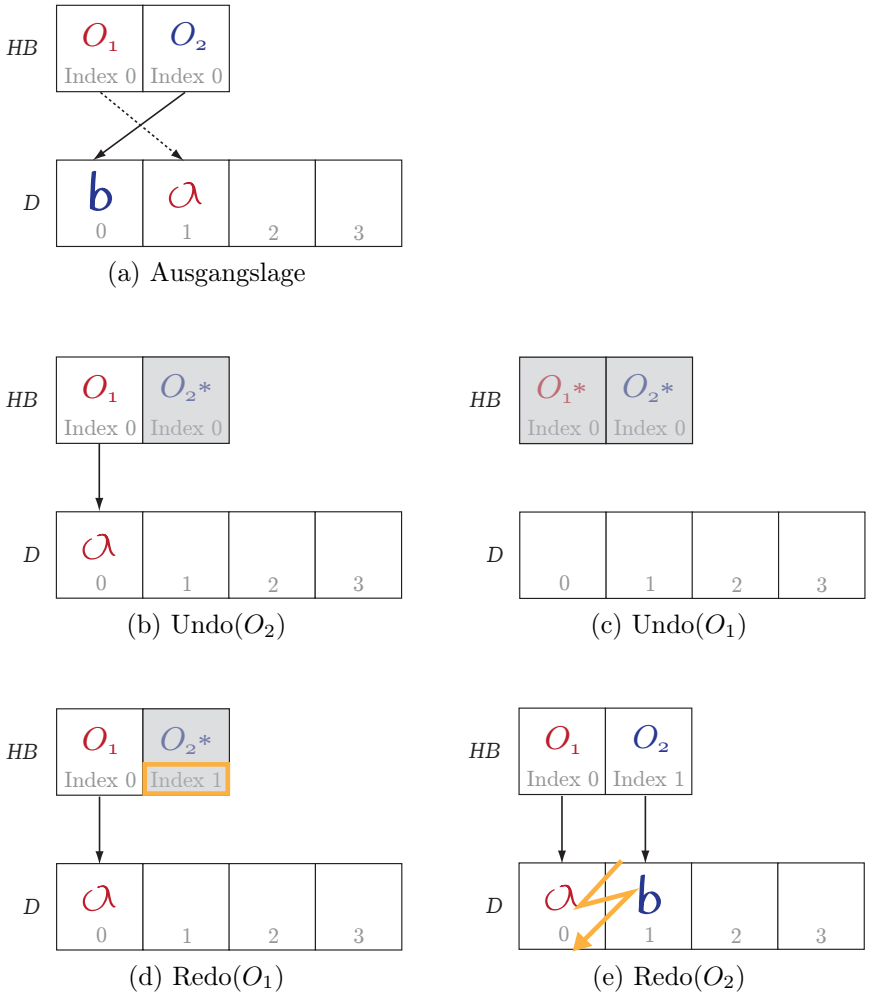


Abbildung 4.4: Mehrdeutige Referenzen entstehen bei mehreren Einfügeoperationen an die gleiche Indexoperation (a). Aufgrund der Transformationsfunktionen in *AnyUndo* kann beim Rückgängigmachen (b, c) bzw. Wiederherstellen (d, e) mehrdeutiger Referenzen unter Umständen ein falsches Datenmodell erzeugt werden.

Gegenüber dem ursprünglichen Datenmodell $D = \{b, a\}$ wurde nach den Redo-Aufrufen ein falsches Datenmodell $D = \{a, b\}$ wiederhergestellt. Das Problem in dem genannten Beispiel ist, dass mehrere Operationen auf eine gleiche Indexposition verweisen und die chronologische Reihenfolge der Operationen (O_2 *nach* O_1) nicht mit der Einfügereihenfolge (O_2 *vor* O_1) übereinstimmt. Durch die Anwendung der Transformationsfunktionen kann es aber vorkommen, dass die Information dieser mehrdeutigen Referenz verloren geht und unter Umständen ein falsches Datenmodell erzeugt wird. In Abbildung 4.4 ist das Problem mehrdeutiger Referenzen dargestellt.

Lösung Eine mögliche Lösung für dieses Problem ist das Aufbauen einer Parallelstruktur, welche die logische Reihenfolge der Einfügeoperationen bzw. mehrdeutigen Operationen enthält. Diese Parallelstruktur kann dann in den Transformationsfunktionen genutzt werden, um zu ermitteln, ob die Indexposition in einem speziellen Fall erhöht bzw. vermindert werden darf. In dem genannten Beispiel würde die Struktur speichern, dass O_2 im Datenmodell ursprünglich vor O_1 eingefügt hat. In Formel 4.21 würde diese Information dann dazu führen, dass die Indexposition nicht erhöht wird, weil O_2 laut Parallelstruktur vor O_1 eingefügt hat.

Somit kann das Problem von mehrdeutigen Referenzen abgefangen werden. Nachteil ist, dass die Parallelstruktur bei jedem Einfügen einer Operation in den Verlauf aufgebaut bzw. aktualisiert werden muss. Dies kann bei vielen Operationen mitunter zum Performanceproblem werden.

4.3.5 Zusammenfassung

Der im letzten Abschnitt beschriebene Algorithmus besteht bei jeder Durchführung aus fünf Teilschritten. Zunächst wird von der ausgewählten Operation die inverse Operation gebildet (Schritt 1) und alle Indexänderungen, bedingt durch nachfolgende Operationen, miteinbezogen (Schritt 2). Anschließend wird die inverse transformierte Operation auf das Datenmodell ausgeführt (Schritt 3). Danach werden die Auswirkungen der inversen transformierten Operation in alle nachfolgenden Operationen miteinbezogen, sodass diese wieder auf einen validen Index verweisen (Schritt 4). Im letzten Schritt wird die rückgängig gemachte Operation als Undo-Paar markiert (Schritt 5).

Der erläuterte Algorithmus ermöglicht, dass jede beliebige Operation aus *HB* zu einem beliebigen Zeitpunkt rückgängig gemacht werden kann. Der auf Operational Transformation basierende Algorithmus verändert den Verlauf dabei immer so, dass die Auswirkungen von nicht mehr existierenden Operationen (Undo) oder wieder existierenden Operationen (Redo) in alle Operationen korrekt miteinbezogen werden. Darüber hinaus wird sichergestellt, dass jede Operation in *HB* beim Aufruf von Undo oder Redo auf eine valide und richtige Indexposition verweist. Damit können verschiedenste Kombinationen von Einfüge- und Löschoperationen mit beliebig verschachtelten Indizes behandelt werden, wobei einige Sonderfälle berücksichtigt werden müssen.

4.4 Framework

Für die Verwaltung von verschiedenen Operationen einer Anwendung mit dem oben genannten Algorithmus wurde ein generisches Framework in C# entwickelt. Damit kann das entwickelte Undo-Framework in verschiedensten auf Microsoft .Net basierenden Anwendungen genutzt und wiederverwendet

werden. Das Framework stellt eine Implementierung des selektiven Undos zur Verfügung, mit der verschiedenste Operationen verwaltet werden können. Für einige Problemstellungen wie der beschriebenen Indexproblematik oder der Behandlung von semantischen Operationen stellt das Framework vorgefertigte Klassen und Verhaltensweisen zur Verfügung. Anwendungen selbst legen nur mögliche Ausprägungen der Operationen fest und regeln das Hinzufügen dieser in den Verlauf aus dem Framework (Klasse `History`).

4.4.1 Generisches Undo-Framework

Neben dem eigentlichen Algorithmus kann auch die Verwaltung von elementaren Indexoperationen, also einer Einfüge- und Löschoperation auf eine einfache Liste, abstrahiert werden. Diese können dem Anwendungsprogrammierer in vorgefertigter Art und Weise, *inklusive* aller Transformationsfunktionen (IT und ET), zur Verfügung gestellt werden. Dabei ist es völlig unerheblich auf welche Liste die Operationen in der späteren Anwendung tatsächlich operieren, da die auf die Indizes bezogenen Transformationsfunktionen davon unabhängig sind. Damit auch Einfüge- und Löschoperationen mehrerer Listen im einzigen Verlauf des Frameworks verwaltet werden können, verfügen die vorgefertigten Operationen `InsertOperation` und `DeleteOperation` über einen gemeinsamen Kontext, damit sich Transformationen von Operationen aus verschiedenen Listen nicht gegenseitig beeinflussen. Die Abstraktion der Einfüge- und Löschoperation hat den großen Vorteil, dass sich der Anwendungsprogrammierer keine Gedanken über die recht komplexen Transformationsfunktionen machen muss, sondern nur festlegt, welche Daten die Einfüge- und Löschoperation enthält und auf welche Liste in der Anwendung die Operationen tatsächlich operieren.

Eine weitere große Stärke des erläuterten selektiven Undo-Algorithmus ist, dass mit einer technischen Lösung *verschiedenste Undo-Methoden* verwendet werden können. So ist mit dem beschriebenen Algorithmus mit einem einzigen Verlauf grundsätzlich globales, benutzerbezogenes, regionales Undo sowie selektives Undo möglich. Welche Undo-Methode verwendet werden soll, entscheidet ausschließlich die Anwendung durch die Aufrufe der entsprechenden Undo- und Redo-Methoden der Verlauf-Klasse `History`.

Das Framework enthält also zusammenfassend die allgemeine Implementierung des selektiven Undo-Algorithmus (`History`) sowie eine Basisklasse für Operationen (`Operation`), von denen die konkreten Ausprägungen der Operationen in einer Anwendung ableiten müssen, sofern sie in der Verlauf-Klasse History verwaltet werden sollen. Darüber hinaus enthält das Framework noch weitere abstrahierte Operationsklassen. `IndexOperation` erweitert eine `Operation` um eine zusätzlich gespeicherte Indexposition. Die Klassen `InsertOperation` und `DeleteOperation` leiten davon ab und bilden vorgefertigte Implementierungen für Einfüge- und Löschoperationen auf eine Liste, inklusive der dafür notwendigen Transformationsfunktionen.

4.4.2 Einbindung in eine Anwendung

Eine Anwendung, die das Framework benutzt, legt Dank des generischen Frameworks in der Regel nur *konkrete Ausprägungen* der in der Anwendung *auftretenden Operationen* fest. Eine konkrete Ausprägung einer Operation (ohne Transformationsfunktionen) in einer Anwendung definiert in der Regel:

- `object Data`
  Daten, auf welche die Operation operiert
- `object Owner`
  Besitzer der Operation
- `DateTime Timestamp`
  Erstellungszeitpunkt der Operation
- `void Execute()`
  Führt die aktuelle Operation aus, z. B. Einfügen/Löschen in das Datenmodell
- `Operation Inverse()`
  Liefert die inverse Operation
- `object Clone()`
  Liefert eine vollwertige Kopie der Operation (keine Referenzen!)
- `Rect GetBounds()`
  Liefert die rechteckigen Begrenzungen der Operation
- `PolygonRegion GetConvexHull()`
  Liefert die konvexe Hülle der Operation

Die auftretenden Operationen in dem in dieser Arbeit beschriebenen grafischen Editor wären u. a.:

- `StrokeAddOperation`
  Hinzufügen eines Einzelstrichs in das Datenmodell
- `StrokeDeleteOperation`
  Löschen eines Einzelstrichs aus dem Datenmodell
- `StrokeTransformationOperation`
  Verschieben eines Einzelstrichs in der Anwendung

`StrokeAddOperation` sowie `StrokeDeleteOperation` stellen dabei Ableitungen der generischen `InsertOperation` und `DeleteOperation` dar und enthalten lediglich den zur Operation gehörigen Einzelstrich sowie definieren, dass die Operationen auf die Liste von Einzelstrichen im Datenmodell operieren. Wie erwähnt legen die vorgefertigte Einfüge- und Löschoperation bereits alle dafür notwendigen Transformationsfunktionen fest. Die Klasse `StrokeTransformationOperation` ändert lediglich Eigenschaften von im Datenmodell befindlichen Inhalten (Transformation) und ist deshalb von der in Abschnitt 4.2 beschriebenen Index- und Transformationsproblematik unbeeinflusst. Für Änderungen von Elementen wie z. B. Bildern sind analog zu den genannten Operationen auch Operationen für Elemente definiert.

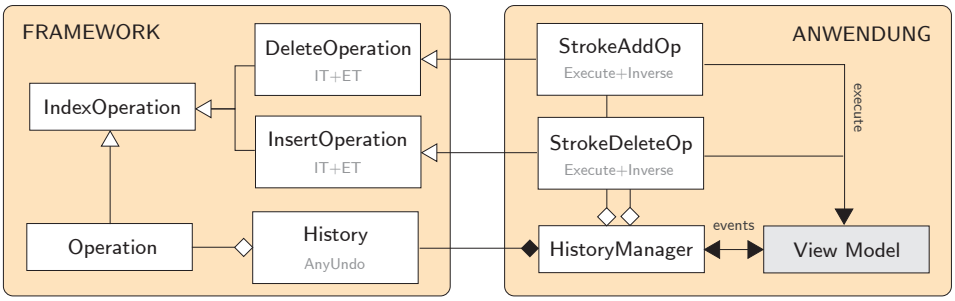


Abbildung 4.5: Das anwendungsunabhängige Framework enthält die Verwaltung des Verlaufs sowie vorgefertigte Operationen. Der anwendungsabhängige Teil legt die konkreten Ausprägungen der Operationen fest. Alle Änderungen die im Datenmodell auftreten, werden vom `HistoryManager` in Operationen abgebildet und in den Verlauf `History` im Framework hinzugefügt.

Neben den konkreten Operationen ist in jeder Anwendung eine Instanz notwendig, die das Hinzufügen und Löschen der Operationen in den Verlauf des Frameworks übernimmt (`HistoryManager`). Alle Aktionen, die ein Benutzer in der Anwendung tätigen kann und die mit Undo/Redo rückgängig machbar oder wiederherstellbar sein sollen, müssen durch diese Instanz in Operationen abgebildet und in den Verlauf eingefügt oder bei Bedarf wieder entfernt werden. Dabei muss von der Anwendung bei jeder Operation die eindeutige Zuweisung eines Benutzers gewährleistet sein, damit dieser als „Besitzer“ der zugehörigen Operation eingetragen werden kann. Werden in einer Anwendung verschiedene Undo-Methoden angeboten, ist der `HistoryManager` darüber hinaus gut geeignet, um der Anwendung die Aufrufe von Undo und Redo zur Verfügung zu stellen. Die Anwendung signalisiert dem `HistoryManager` allgemein, dass jetzt ein Undo ausgeführt werden soll, während der `HistoryManager` aufgrund der aktuell ausgewählten Undo-Methode den korrespondieren Undo-Aufruf an das Framework absetzt. Das Undo-Framework kann dann von der Anwendung vollständig verborgen werden, während der `HistoryManager` als Fassade den einzigen Zugriffspunkt auf Undo und Redo aus der Anwendung darstellt. Die Architektur und das Zusammenspiel von Framework und Anwendung ist in Abbildung 4.5 schematisch veranschaulicht.

ModelView-ViewModel

Der in dieser Arbeit verwendete grafische Editor basiert auf dem ModelView-ViewModel (MVVM), welches sich durch die strikte Trennung zwischen Daten und Oberfläche auszeichnet. Während das *Model* (Datenmodell) die Daten der Anwendung beinhaltet, stellt das *ViewModel* die aufbereiteten Daten

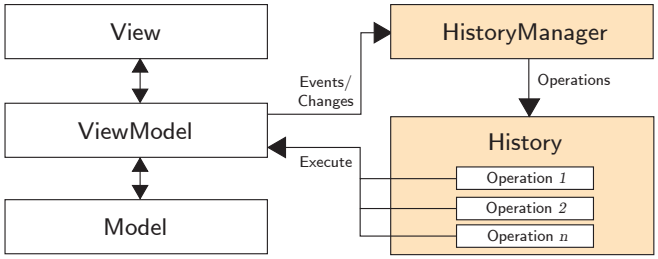


Abbildung 4.6: MVVM-basierende Anwendungen bieten üblicherweise umfangreiche Benachrichtigungs- und Event-Mechanismen, die dazu genutzt werden können, dass Undo auch in bestehende Anwendungen integriert werden kann. Dabei können beispielsweise alle *ViewModel*-Änderungen von einer zentralen Instanz als Operationen abgebildet und in einem als Parallelstruktur gehaltenen Verlauf gespeichert werden.

für die Oberfläche bereit [22]. Gerade in Windows Presentation Foundation (WPF) wird dieses Konzept oft verwendet, weil die umfangreichen Möglichkeiten von *Data Bindings* in WPF für das MVVM-Modell prädestiniert sind.

Grundsätzlich sollte Undo bereits in der Frühphase der Anwendungsentwicklung vorgesehen und integriert werden [1]. Manchmal ist es aber erforderlich, dass Undo als Funktion nachträglich in bereits bestehende Anwendungen integriert werden soll. Dabei gestaltet sich eine derartige nachträgliche Integration von Undo und Redo oft als schwierig. Für MVVM trifft dies jedoch nicht ganz zu, da auf MVVM basierende Anwendungen aufgrund ihrer *DataBinding*-fähigen Struktur üblicherweise *umfangreiche Benachrichtungs- und Event-Mechanismen* bereitstellen, die jede Änderung im MVVM-Modell mitteilen. Diese Benachrichtigungen können bei einer nachträglichen Implementierung von einer zentralen Instanz genutzt werden, um die *Model*- bzw. *ViewModel*-Änderungen in Operationen abzubilden und in einer Parallelstruktur (`History`) zu speichern (siehe Abbildung 4.6). Die im Verlauf gespeicherten Operationen müssen dabei die Daten nicht redundant speichern, sondern können auf die Daten aus dem Datenmodell verweisen.

Für den in dieser Arbeit beschriebenen nicht verteilten grafischen Editor wurde der Verlauf nachträglich an das *ViewModel* angebunden. Das *ViewModel* verfügt in der Regel über noch umfangreichere Benachrichtigungsmechanismen als das *Model* und darüber hinaus können *View*-bezogene Daten, wie z. B. benutzerbezogene oder werkzeugbezogene Informationen, leichter in den Verlauf miteinfließen. Alle *ViewModel*-Änderungen werden also von der zentralen Instanz `HistoryManager` in Operationen abgebildet und in dem als Parallelstruktur gehaltenem Verlauf (`History`) abgelegt. Undo- oder Redo-Aufrufe des Verlaufes ändern dann, ähnlich wie ein imaginärer Benutzer, das *ViewModel* selbst, welches die Änderungen auf das Datenmodell überträgt.

Grundsätzlich kann die Einbindung des Undo-Frameworks in eine auf dem MVVM basierenden Anwendung auch noch auf anderen Arten bzw. an anderen Stellen erfolgen, die aber in der Regel voraussetzen, dass Undo bereits in der Frühphase des Anwendungsdesigns berücksichtigt wird.

4.4.3 Bündelung von Operationen

Wie bereits in Abschnitt 3.4.2 erläutert, ist es nicht immer sinnvoll jede elementare Operation als Undo-Schritt zur Verfügung zu stellen. Deshalb bietet das Framework auch eine Möglichkeit, wie verschiedenste elementare Operationen zu einer *gemeinsamen Operation* (`BatchOperation`) zusammengebündelt werden können (siehe Abbildung 4.7). Jede elementare Operation, die in den Verlauf hinzugefügt wird, wird immer auch einer Batch-Operation zugeordnet. Wird beispielsweise nur eine einzelne elementare Operation zum Verlauf hinzugefügt, wird eine Batch-Operation erzeugt, die nur diese elementare Operation enthält. Darüber hinaus besteht aber die Möglichkeit, dass eine Batch-Operation je nach Anwendungsfall selbst erzeugt und geschlossen werden kann. Somit können beispielsweise Werkzeuge signalisieren, wann eine Aktion eines Benutzers beginnt und wann sie endet. Diese Signale können an das Erzeugen und Schließen von Batch-Operationen gekoppelt werden. Benutzt beispielsweise ein Benutzer ein Radierwerkzeug, wird beim

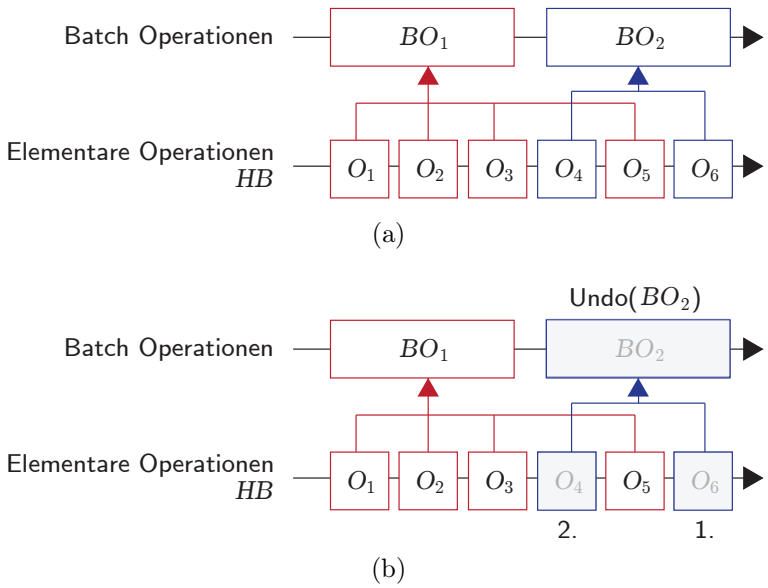


Abbildung 4.7: Elementare Operationen werden zu gebündelten, benutzerbezogenen Batch-Operationen zusammengefasst (a). Wird eine Batch-Operation rückgängig gemacht, werden alle beinhalteten Operationen von hinten nach vorne rückgängig gemacht (b).

Down-Event aufgrund des Events des Werkzeuges eine Batch-Operation für diesen Benutzer erzeugt. Alle Löschoperationen, die in der Folge durch das Radierwerkzeug des Benutzers auftreten, werden dann in die aktive Batch-Operation des Benutzers gekapselt. Signalisiert das Werkzeug beim *Up-Event* das Ende der Interaktion wird die aktuelle Batch-Operation des Benutzers geschlossen und im Verlauf abgelegt.

Da zu jeder elementaren Operation auch eine Batch-Operation besteht, wird vom Framework nur mehr das Rückgängigmachen von gebündelten Batch-Operationen und nicht von elementaren Operationen erlaubt. So würde beispielsweise bei einem Undo einer Batch-Operation alle in ihr gespeicherten elementaren Operationen von hinten nach vorne rückgängig gemacht, während bei einem Redo die beinhalteten elementaren Operationen von vorne nach hinten wiederhergestellt würden. Der interne Undo-Algorithmus selbst arbeitet aber unabhängig der Batch-Operationen weiterhin mit den elementaren Operationen.

4.4.4 Auswahl der nächsten Undo-Schritte

Wie bereits erwähnt, können von der Anwendung aus verschiedenste Undo-Techniken aufgerufen werden. Dazu stellt das Framework für Undo verschiedene Methoden zur Verfügung (die Redo-Methoden sind analog dazu vorhanden):

- `Undo()`
  Globales Undo
- `Undo(object owner)`
  Benutzerbezogenes Undo
- `Undo(PolygonRegion region)`
  Regionales Undo
- `UndoBatch(BatchOperation o)`
  Selektives Undo

Die Operationen für ein globales Undo bzw. Redo werden aufgrund der chronologischen Reihenfolge im Verlauf ausgewählt. Die nächste globale Operation für Undo ist die letzte im Verlauf, die nicht als rückgängig gemacht gekennzeichnet ist. Für Redo hingegen ist die nächste globale Operation die am weitesten vorne im Verlauf platzierte Operation, die als rückgängig gemacht gekennzeichnet ist. Ähnlich verhalten sich auch benutzerbezogenes und regionales Undo, wobei bei benutzerbezogenem Undo zusätzlich noch der gegebene Benutzer und bei regionalem Undo die gegebene Region übereinstimmen muss (siehe Abbildung 4.8). Für regionales Undo im Speziellen kann im Framework festgelegt werden, ob die als Parameter übergegebene Region die Begrenzungen der jeweiligen Batch-Operation komplett oder nur teilweise beinhalten muss. Die Begrenzungen der Batch-Operationen werden von den elementaren Operationen festgelegt. Jede konkrete Implementierung

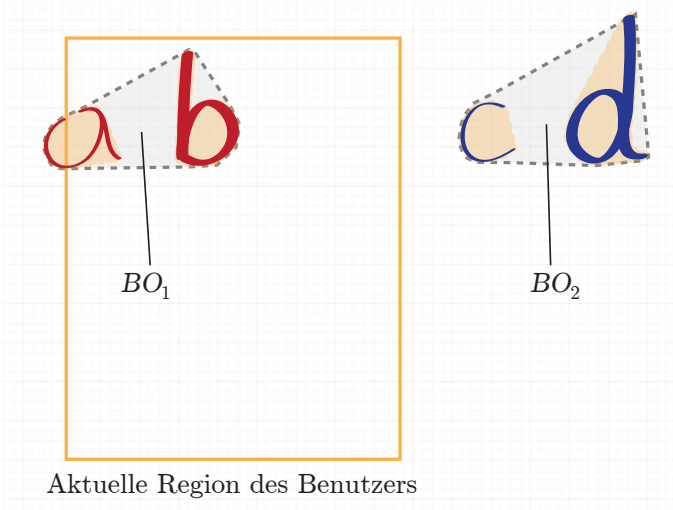


Abbildung 4.8: Die Begrenzungen von Batch-Operationen ergeben sich jeweils durch die konvexe Hülle aller beinhalteten Operationen. Eine Batch-Operation wird immer dann für Undo und Redo eines bestimmten Benutzers berücksichtigt, sofern sie je nach Einstellung im Framework komplett oder teilweise in der aktuellen Region des Benutzers liegt.

einer elementaren Operation in der Anwendung muss auch seine Begrenzungen (konvexe Hülle sowie Axis-Aligned Bounding-Box) bekannt geben. Die Begrenzungen der Batch-Operation ergeben sich aus den vereinigten Begrenzungen der beinhalteten Operationen.

4.4.5 Semantische Abhängigkeiten

Es kann natürlich vorkommen, dass sich zwischen den verschiedenen elementaren Operationen *semantische Abhängigkeiten* ergeben. Eine Abhängigkeit tritt immer dann auf, wenn Operationen auf den gleichen Index bzw. den gleichen Inhalt im Datenmodell verweisen.

Anschaulich wird das Problem in folgendem Beispiel: Angenommen ein Benutzer A zeichnet einen Einzelstrich an die Position P_1 und ein zweiter Benutzer B verschiebt den Strich anschließend auf Position P_2. Damit Benutzer A die Aktion „Zeichne Strich bei Position P_1“ rückgängig machen kann, muss zunächst der ursprüngliche Zustand wiederhergestellt und der Strich von P_2 zu P_1 zurückverschoben werden.

Daraus ergibt sich, dass ein derartiger Undo-Aufruf ohne dem Rückgängigmachen der Abhängigkeiten nicht möglich ist oder dass zuerst alle Abhängigkeiten rückgängig gemacht bzw. wiederhergestellt werden müssen. Die Studie von Cass et al. ergab, dass die Benutzer am ehesten erwarten, dass semantische Abhängigkeiten mit rückgängig gemacht werden [6]. Be-

vor also eine Operation rückgängig gemacht werden kann, müssen zuerst alle abhängigen Operationen rückgängig gemacht werden. In einer konkreten Implementierung muss dies rekursiv geschehen, da auch abhängige Operationen mit mehrfacher Schachtelung möglich sind. Umgekehrt gilt dies natürlich auch für Redo. Soll eine beliebige Operation wiederhergestellt werden, müssen zuerst auch all jene Operationen rekursiv wiederhergestellt werden, von denen die Operation abhängig ist. Ein derartiges Verhalten ist im Framework in `AdvancedBatchOperation` vorimplementiert. Wird eine Batch-Operation rückgängig gemacht, werden zuerst rekursiv alle abhängigen Batch-Operationen (ergeben sich durch die Abhängigkeiten der beinhalteten elementaren Operationen) rückgängig gemacht bzw. wiederhergestellt.

Geht eine semantische Abhängigkeit über den gleichen Index bzw. Inhalt hinaus (z. B. durch anwendungsbezogene Abhängigkeiten) können etwaige semantische Abhängigkeiten der elementaren Operationen auch manuell durch explizites Setzen der `DependentOn`-Eigenschaft einer Operation gebildet werden.

4.5 Umsetzung der regionalen Undo-Konzepte

Um die in Kapitel 3 erläuterten Konzepte für regionales Undo in einer einzigen allgemeinen technischen Lösung zu integrieren, wurde in der Anwendung eine gemeinsame Basis für regionales Undo eingeführt. Einzig das Verfahren basierend auf dem Selektionswerkzeug lässt sich nicht gut abstrahieren, weil dieses stark vom Selektionswerkzeug abhängt und mitunter in dieses integriert werden muss. Für alle anderen Verfahren wurde hingegen verallgemeinert, dass jedes Konzept für regionales Undo für jeden Benutzer aus einer

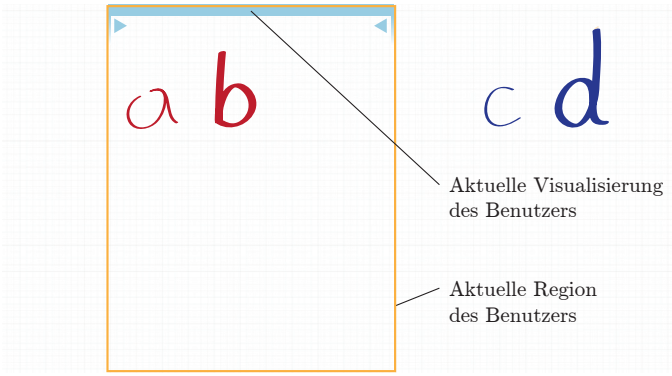


Abbildung 4.9: Jede regionale Undo-Technik verfügt für jeden Benutzer über eine (nicht sichtbare) Region und einer Visualisierung. Die Guideline-Technik besteht beispielsweise aus einer Region, die über die gesamte Whiteboard-Höhe reicht, sowie der entsprechenden Visualisierung an der Oberkante der Anwendung.

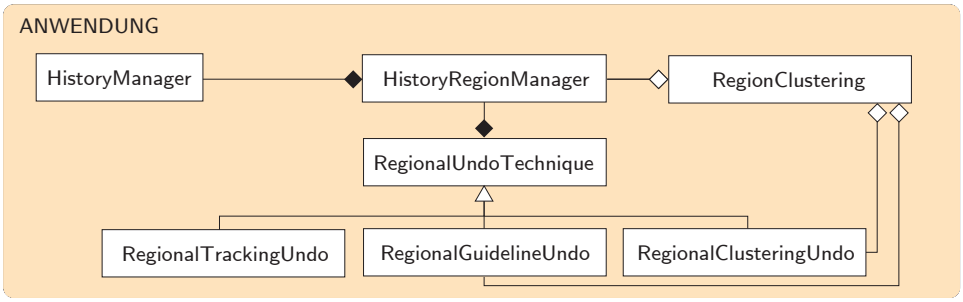


Abbildung 4.10: Die verschiedenen regionalen Undo-Techniken werden von einer gemeinsamen Instanz verwaltet. Jede regionale Undo-Technik liefert eine aktuelle Region und eine aktuelle Visualisierung für jeden Benutzer, die für die Auswahl des nächsten Undo- bzw. Redo-Schrittes jedes Benutzers herangezogen wird.

nicht sichtbaren Region und einer *Darstellung dieser* besteht (siehe Abbildung 4.9). Die Klasse `RegionalUndoTechnique` beinhaltet diese Abstraktion, welche als Schnittstelle die zwei Methoden

- `PolygonRegion CreateRegion(object user)` und
- `UIElement CreateVisualization(object user)`

zur Verfügung stellt. Für die regionalen Undo-Techniken stehen konkrete Implementierungen

- `RegionalClusteringUndo` ,
- `RegionalGuidelineUndo` und
- `RegionalTrackingUndo`

bereit, die aufgrund des jeweiligen Verfahrens eine andere Region und Visualisierung bereitstellen. Die Klasse `HistoryRegionManager` verwaltet die aktuelle Region und aktuelle Visualisierung jedes Benutzers. Über diese Instanz kann dann der `HistoryManager` entsprechend dem ausgewählten regionalen Undo-Verfahren nach Bedarf Region und Visualisierung für jeden Benutzer abfragen und anzeigen, ohne dass dem Rest der Anwendung die konkret verwendete regionale Undo-Technik bekannt sein muss. Der nächste Undo- oder Redo-Schritt bei einem etwaigen Undo- oder Redo-Aufruf in `HistoryManager` (siehe Abschnitt 4.4.4) wird dann aufgrund der aktuellen Region eines Benutzers aus dem `HistoryRegionManager` ermittelt. Der beschriebene Aufbau ist in Abbildung 4.10 visualisiert.

4.6 Zusammenfassung

In diesem Kapitel wurde als Grundlage von regionalem Undo zunächst ein selektiver Undo-Algorithmus vorgestellt, der es erlaubt Operationen aus dem

Verlauf in beliebiger Reihenfolge rückgängig zu machen oder wiederherzustellen. Dabei werden die Operationen und das Datenmodell jeweils so verändert, als hätte eine rückgängig gemachte Operation nie existiert. Darüber hinaus wurde ein generisches Framework vorgestellt, dass die Umsetzung des beschriebenen Algorithmus sowie einige vorgefertigte Funktionalitäten enthält. Dieses Framework kann in verschiedenen Anwendungen eingesetzt werden und stellt dem Anwendungsprogrammierer die umfangreiche technische Basis zur Verfügung, die er benötigt, um regionales Undo in einer Anwendung umzusetzen. Im weiteren Verlauf werden einige technische Konzepte des Frameworks und die Einbindung in den in dieser Arbeit verwendeten grafischen Editor beschrieben. Abschließend wird erläutert, wie die regionalen Undo-Techniken dieser Arbeit allgemein umgesetzt wurden.

Kapitel 5

Diskussion

Die Studie zur Erwartungshaltung der Benutzer zeigte, dass auf großen interaktiven Flächen ein einfaches benutzerbezogenes oder globales Undo nicht den Erwartungen der Benutzer entspricht. Allgemein formuliert erwarten laut Studie Einzelpersonen ein benutzerbezogenes und Gruppen ein gruppenbezogenes Undo, wobei verschiedene Konstellationen von Einzelpersonen und Gruppen diese Erwartungen nicht ändern. Diese Erwartungshaltung kann am häufigsten *regionales Undo* erfüllen, also ein globales Undo in jeder Region in der von Gruppen oder Einzelpersonen gearbeitet wird. Des Weiteren wurde herausgefunden, dass gerade getrennt voneinander arbeitende Benutzer einander nicht aktiv wahrnehmen. Getrennt arbeitende Benutzer wissen also nicht genau, was andere Benutzer auf der gleichen Interaktionsfläche im Konkreten arbeiten. Diese Erkenntnisse müssen bei regionalem Undo darin einfließen, dass getrennt arbeitende Benutzer von wechselseitigen Undo-Aufrufen weder beeinflusst noch gestört werden dürfen.

Die größte Schwierigkeit bei der Umsetzung von regionalem Undo stellt die *Bestimmung der Regionen* dar. Die Bestimmung von Regionen auf einer großen interaktiven Fläche kann entweder manuell durch den Benutzer oder automatisiert erfolgen. Bei der automatisierten Regionsfindung kann diese entweder inhaltsbasierend, positionsbezogen aufgrund der Position der Benutzer oder blickfeldbezogen erfolgen. Oft ist es *schwierig* durch automatisierte Verfahren *klare Trennlinien* zwischen den Arbeitsbereichen und Arbeitsinhalten der verschiedenen Gruppen und Einzelpersonen zu finden, weil oft semantische Zusammenhänge eine exakte Trennung nicht zulassen.

Das wesentlichste Problem bei der inhaltsbasierenden Regionsfindung ist die für Undo notwendige Miteinbeziehung von *nicht sichtbaren Inhalten.* Obwohl der Mensch Regionen sehr stark aufgrund von sichtbaren Informationen bildet, müssen aber für Undo auch nicht sichtbare Operationen, wie zum Beispiel gelöschte Inhalte, berücksichtigt werden. Die daraus resultierende Diskrepanz zwischen den visuellen, für den Benutzer sichtbaren, Regionen und der für Undo gebildeten Regionen stellt für die Benutzer ein Problem dar.

Eine mögliche Lösung für dieses Problem ist, dass nur eine begrenzte Anzahl der letzten Operationen bzw. der nicht sichtbaren Operationen in die Regionsbildung für Undo miteinfließen. Ein weiteres wesentliches Problem ist das kontinuierliche Wachstum der Regionen über längere Arbeitsphasen. Werden in die inhaltliche Regionsbildung immer alle Operationen miteinbezogen, wachsen die Regionen über einen größeren Arbeitszeitraum so stark an, dass sich regionales Undo nicht mehr von globalem dokumentenbasierten Undo unterscheidet. Das Problem kann ebenfalls nur dadurch gelöst werden, dass nur eine begrenzte Anzahl von Operationen in die Regionsbildung miteinbezogen werden. Denkbar wäre auch, dass nach jedem Undo wieder eine zusätzliche Operation in die Regionsbildung miteinfließt, sodass beispielsweise konstant die letzten 30 rückgängig machbaren Operationen berücksichtigt werden. Sichergestellt müsste bei einem solchen Verfahren nur sein, dass trotz der laufenden Änderungen der Regionen bei einem Redo auch wieder die gleichen Operationen, die rückgängig gemacht wurden, wiederhergestellt werden (siehe Anforderungen Abschnitt 4.1).

Neben der inhaltsbasierenden Regionsbestimmung stellen jene Konzepte eine gute Lösung für regionales Undo dar, die die *Regionen aufgrund der Position der Benutzer* bestimmen. In Tests der Konzepte zeigte sich, dass es Benutzer im Gegensatz zu inhaltsbasierenden Regionen leichter fällt zu verstehen, dass von Undo und Redo all jene Operationen betroffen sind, die in *ihrem* Arbeitsbereich liegen. Zentrale Fragestellung in einem derartigen Konzept ist, wie der Arbeitsbereich eines Benutzers möglichst exakt bestimmt werden kann. Zumeist ist bei einer großen interaktiven Fläche nur eine Annäherung der Position über Interaktionsbereiche der Benutzer oder mittels etwaigen Elementen oder Werkzeugen, die sich in der Nähe des Benutzers befinden (z. B. stiftbezogenes Benutzermenü), möglich. Das Problem ist nur, dass nicht sichergestellt ist, dass eine derartige Annäherung tatsächlich der Position des Benutzers entspricht. Dieser Umstand kann eigentlich nur durch eine Technik erreicht werden, welche die *tatsächliche Position des Benutzers* in die Regionsfindung miteinbezieht. Aus der Erfahrung dieser Arbeit heraus sind positions- und blickfeldbezogene Konzepte für die Benutzer leichter verständlich als inhaltsbasierende, jedoch können sich gerade aus der Kombination von beiden gute Synergien ergeben.

Es wird angenommen, dass die Proximity-Technik sich in der Regel aufgrund des zugrundeliegenden Konzeptes und der Miteinbeziehung der Position bzw. des Blickfeldes des Benutzers äußerst intuitiv verhält. Auf der anderen Seite kann die Technik auf Benutzer jedoch auch sehr gewöhnungsbedürftig wirken, weil gerade unerfahrene Benutzer es nicht gewohnt sind, mit ihrem Blickfeld oder ihrer Position Teile einer Software zu steuern. Unabhängig von diesen Erkenntnissen wäre ein Tracking-System wünschenswert, dass auch Kopfneigung und -drehung miteinbezieht und folglich ein *exaktes Blickfeld* ermittelt. Dies würde die Intuitivität dieser Technik mit großer Wahrscheinlichkeit stark steigern.

5.1 Ausblick

Wie effizient bzw. gut ein Konzept für regionales Undo tatsächlich funktioniert, hängt stark davon ab, wie gut und wie genau die Arbeitsbereiche der Benutzer erkannt werden. Nur wenn regionales Undo wenig Fehler in der Regionsfindung und vor allem in der richtigen Auswahl des nächsten (erwarteten) Undo- bzw. Redo-Schrittes macht, bringt eine derartige Lösung für Benutzer auch einen Mehrwert gegenüber herkömmlichen Undo-Methoden. Konzepte in diesem Bereich müssen also vor allem diese Probleme bestmöglich lösen. Wie sich die in dieser Arbeit vorgestellten Konzepte in Bezug auf die genannten Punkte verhalten, kann nur in einer *längerfristigen Evaluierung* ermittelt werden.

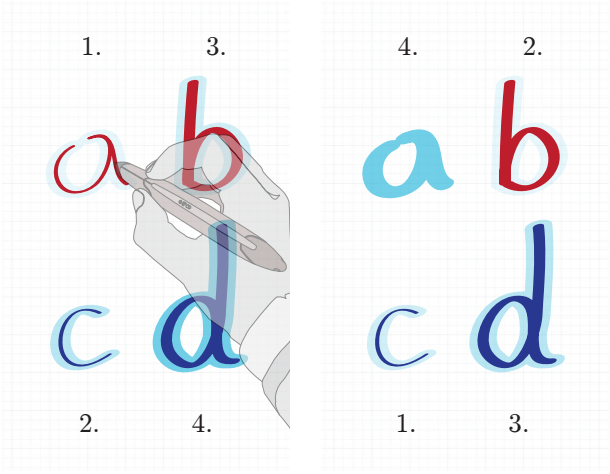
Zum jetzigen Zeitpunkt ist innerhalb einer gebildeten Region nur ein schrittweise chronologisches Undo möglich. Jeder Benutzer kann mit dem Undo- oder Redo-Button aufgrund der chronologischen Abfolge der Operationen beliebig viele Schritte rückgängig machen oder wiederherstellen. In den folgenden Abschnitten werden noch zwei weitere Konzepte präsentiert, die die Funktionalität des chronologischen Undos erweitern und dem Benutzer mehr Flexibilität in Bezug auf Undo bieten. Diese zusätzlichen Konzepte werden in näherer Zukunft in die bestehende Implementierung integriert.

5.1.1 Selektives Undo

Das alleinige Vorhandensein von schrittweisem Undo stellt unter Umständen eine zu große Einschränkung für den Benutzer dar [31]. Möglicherweise möchte der Benutzer eine bestimmte Operation innerhalb der Region auswählen oder ein bestimmtes Objekt wählen, um dessen Zustand durch Undo und Redo zu verändern. Beide Möglichkeiten sind durch schrittweise chronologisches Undo und Redo alleine nicht möglich. Deshalb sollte eine Möglichkeit bestehen, innerhalb einer Region *selektiv* Operationen rückgängig zu machen bzw. wiederherzustellen.

Viele Anwendungen lösen diese Problematik mittels der Auswahl der gewünschten Operation in einer textuellen Auflistung aller Operationen. Dies ist für den Benutzer nicht sonderlich intuitiv, da es voraussetzt, dass der Benutzer die gewünschte Operation in der textuellen Auflistung wiederfindet [14,15]. Einige Arbeiten [15,17] verwenden deshalb grafische Visualisierungen der Operationen, um dem Benutzer die Auswahl zu erleichtern. Nichtsdestoweniger ist dies nicht im Dokument selbst möglich und benötigt spezielle Dialoge zur Auswahl.

Eine Möglichkeit, selektives Undo direkt in Dokumentregionen zu erreichen ist die direkte Auswahl einer visualisierten Operation innerhalb einer Region (siehe Abbildung 5.1). Ein Benutzer hat die Möglichkeit anstelle der einfachen Undo- und Redo-Buttons direkt in einen Modus zu wechseln, der selektives Undo innerhalb seiner aktiven Region ermöglicht. Innerhalb



(a) (b)

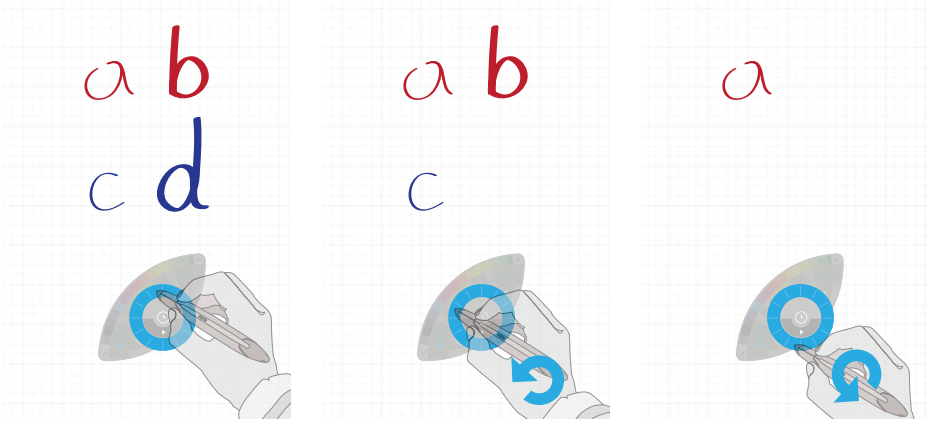
Abbildung 5.1: In einem eigenen Modus kann selektives Undo für eine bestimmte Region durchgeführt werden. Alle verfügbaren Operationen werden dargestellt, wobei die Transparenz die Aktualität der Operation kennzeichnet. Durch einfaches Tippen mit dem Stift kann ein Benutzer so eine bestimmte Operation rückgängig machen (a). Die gewählte Operation wird durchgeführt (*a* wird gelöscht), wobei auch diese wieder als neue (rückgängig machbare) Operation angezeigt wird (b).

der Region werden dann die letzten Schritte (Operationen) mit abfallender Transparenz dargestellt. Durch Tippen auf den jeweilig visualisierten Schritt kann dieser selektiv rückgängig gemacht werden.

Das Verfahren hat den großen Vorteil, dass es aufgrund der Einbeziehung der gebildeten Regionen andere Benutzer in deren Arbeitsfluss nicht stört. Darüber hinaus ist für einen Benutzer das Finden der gewünschten Operation durch die visuelle Repräsentation einfach, weil keinerlei „Mapping“ des Benutzers bezogen auf textuelle Beschreibungen oder Miniaturansichten notwendig ist. Der Nachteil bei dem beschriebenen Verfahren ist, dass nicht alle im Verlauf befindlichen Operationen ausgewählt werden können, sondern nur jene, die sich auf den aktuellen Status des Dokumentes beziehen.

5.1.2 History-Wheel

Undo wird grundsätzlich zur Behebung von Fehlern, dem Austesten von neuen Programmfeatures sowie zum Herstellen früherer Dokumentversionen verwendet. Die ersten beiden Punkte werden durch schrittweise chronologisches und selektives Undo weitestgehend abgedeckt. Für das Herstellen von früheren Dokumentzuständen sind chronologisches oder selektives Undo aufgrund der möglicherweise beträchtlichen Anzahl von Operationen nicht sonderlich geeignet. Hier wäre eine Möglichkeit wünschenswert, die die stufenlose Explo-



(a) (b) (c)

Abbildung 5.2: Das History-Wheel erlaubt eine stufenlose Exploration von Teilen oder dem ganzen Dokument. Dabei steuert das Drehen am Rad die Zeit, während der Radius die Granularität bestimmt.

ration der Entwicklung von Dokumenten ermöglicht. Solche Verfahren gibt es zwar bereits [8, 12, 13, 16], jedoch ermöglicht das Konzept von regionalem Undo eine natürliche Exploration der Entwicklung des eigenen Arbeitsbereiches bzw. der aktuellen Region.

Zu diesem Zweck wurde ein Konzept entwickelt, das als *History-Wheel* bezeichnet wird. Das „Zeitrad“ können die Benutzer aufrufen, wenn sie zu einem früheren Status des Dokuments zurückkehren oder die Entwicklung einer Dokumentregion oder ihres Arbeitsbereiches betrachten möchten. Durch *Rückwärtsdrehen bzw. Vorwärtsdrehen* wird wahlweise die *Zeit* und somit der Status der aktiven Region *zurück- oder vorgespult*. Angedacht ist auch, dass der Radius zum Zentrum des *History-Wheels* über die Granularität der Operationen entscheidet. Bei großen Radien können nahezu elementare Operationen rückgängig gemacht werden, während bei kleineren Radien gebündelte Operationen betroffen sind. Durch die Wahl der Drehgeschwindigkeit und die Wahl des Radius kann der Benutzer somit sehr diffizil die Exploration des Dokuments steuern. Möglich ist auch einen Abspielmodus zu integrieren, der eine Region von Anfang bis zum Ende ähnlich wie in einem Video aufbaut. Der Benutzer kann den Geltungsbereich des *History-Wheels* über die Adaptierung seiner aktiven Region beeinflussen (z. B. zur Exploration des gesamten Dokuments).

Kapitel 6

Zusammenfassung

Diese Arbeit beschäftigte sich mit Undo und Redo bezogen auf Multi-User-Anwendungen auf großen interaktiven Flächen. Um die Erwartungshaltung der Benutzer in verschiedenen Konstellationen von Einzel- und Gruppenarbeit in einem derartigen Szenario zu ermitteln, wurde eine empirische Studie durchgeführt. Dabei wurde herausgefunden, dass sich Gruppen grundsätzlich ein gruppenbezogenes und Einzelpersonen ein benutzerbezogenes Undo erwarten. Diese Erwartungen werden am häufigsten durch ein regionales Undo erreicht, welches einem globalen Undo für jede Region entspricht, in der von einer Gruppe oder Einzelperson gearbeitet wird.

Darauf aufbauend wurden die Anforderungen vorgestellt, die in der Umsetzung von regionalem Undo in einer großflächigen Multi-User-Anwendung bestehen. Gemäß diesen wurden vier verschiedene Techniken präsentiert, wie die Arbeitsbereiche bzw. Regionen von verschiedenen, gleichzeitig arbeitenden Gruppen und Einzelpersonen am Whiteboard ermittelt werden können. Neben einem manuellen Konzept, indem der Benutzer selbst Regionen definiert, wurden weitere automatisierte Verfahren vorgestellt, die aufgrund der Inhalte im Dokument, der Position der Benutzer oder des Blickfeldes der Benutzer Regionen bilden und so versuchen die verschiedenen Arbeitsbereiche bestmöglich zu erkennen.

Darüber hinaus stellt die Arbeit ein auf selektives Undo basierendes Undo-Framework vor, welches notwendig ist, um regionales Undo in einer Multi-User-Anwendung technisch überhaupt zu ermöglichen. Das generalisierte Framework erlaubt den Einsatz in verschiedenen Anwendungen und stellt dem Anwendungsprogrammierer eine umfangreiche technische Basis zur Verfügung, welche für die Umsetzung von regionalem Undo notwendig ist. Neben der Beschreibung des Frameworks wird auch die konkrete Umsetzung von selektivem Undo als technische Grundlage von regionalem Undo umfangreich erläutert.

Um herauszufinden, wie gut regionales Undo anhand der in dieser Arbeit präsentierten Konzepte in der Praxis tatsächlich funktioniert, ist eine

Evaluierung der verschiedenen Techniken vorgesehen. Dabei ist neben der Effizienz und Fehleranfälligkeit der verschiedenen Techniken vor allem interessant, wie sich die Handhabung der Konzepte über längere Arbeitsphasen und längere Dokumente verhält.

Anhang A

Transformationsfunktionen

Für den in Abschnitt 4.3 beschriebenen Undo-Algorithmus bzw. Einfüge- und Löschoperationen auf eine einfache Liste können folgende Transformationsfunktionen verwendet werden. Für einfache Lösch- und Einfügeoperationen entstehen für IT vier verschiedene Transformationsfunktionen. Die Transformationsfunktion zweier Einfügeoperationen ist in Algorithmus A.1, einer Einfüge- und Löschoperation in Algorithmus A.2, einer Lösch- und Einfügeposition in Algorithmus A.3 und von zwei Löschoperationen in Algorithmus A.4 angeführt. Die Methode IsBefore berücksichtigt bei mehrdeutigen Referenzen die logische Einfügereihenfolge von Einfügeoperationen im Datenmodell (siehe Abschnitt 4.3.4).

Algorithmus A.1: IT von zwei Einfügeoperationen

```
1: IT(Ins(o1, p1), Ins(o2, p2))
2:     if p2 < p1 ∨ (p2 = p1 ∧ IsBefore(Ins(o1, p1), Ins(o2, p2))) then
3:         return Ins(o1, p1 + 1)   ▷ z. B. IT(Ins(a, 4), Ins(b, 3)) = Ins(a, 5)
4:     else
5:         return Ins(o1, p1)       ▷ z. B. IT(Ins(a, 3), Ins(b, 4)) = Ins(a, 3)
6:     end if
7: end
```

Algorithmus A.2: IT einer Einfüge- und Löschoperation

```
1: IT(Ins(o1, p1), Del(o2, p2))
2:     if p2 < p1 then
3:         return Ins(o1, p1 − 1)   ▷ z. B. IT(Ins(a, 3), Del(b, 1)) = Ins(a, 2)
4:     else
5:         return Ins(o1, p1)       ▷ z. B. IT(Ins(a, 3), Del(b, 4)) = Ins(a, 3)
6:     end if
7: end
```

Algorithmus A.3: IT einer Lösch- und Einfügeoperation

```
1: IT(Del(o1, p1), Ins(o2, p2))
2:     if p2 ≤ p1 then
3:         return Del(o1, p1 + 1) ▷ z. B. IT(Del(a, 4), Ins(b, 3)) = Del(a, 5)
4:     else
5:         return Del(o1, p1)     ▷ z. B. IT(Del(a, 3), Ins(b, 4)) = Del(a, 3)
6:     end if
7: end
```

Algorithmus A.4: IT zweier Löschoperationen

```
1: IT(Del(o1, p1), Del(o2, p2))
2:     if p2 < p1 then
3:         return Del(o1, p1 − 1) ▷ z. B. IT(Del(a, 4), Del(b, 2)) = Del(a, 3)
4:     else
5:         if p2 > p1 then
6:             return Del(o1, p1) ▷ z. B. IT(Del(a, 3), Del(b, 4)) = Del(a, 3)
7:         else
8:             return I                                  ▷ Identity-Operation
9:         end if
10:    end if
11: end
```

Aus der Umkehrbarkeit von Transformationsfunktionen

$$O_1' = IT(O_1, O_2) \text{ und} \tag{A.1}$$

$$O_1 = ET(O_1', O_2) \tag{A.2}$$

können analog dazu auch die vier möglichen Transformationsfunktionen für ET abgeleitet werden.

Anhang B

Inhalt der DVD

B.1 PDF-Dateien

Pfad: /

Rendl_Christian_2011.pdf Diplomarbeit

B.2 Ressourcen

Pfad: /

Images/ . . . . . Bilder
Videos/ . . . . . Videos
Study Results/ . . . . . Auswertung der Studie
Architecture/ . . . . . Klassendiagramm

Literaturverzeichnis

[1] Abowd, G. D. und A. J. Dix: *Giving undo attention.* Interact. Comput., 4:317–342, December 1992.

[2] Archer, Jr., J. E., R. Conway und F. B. Schneider: *User Recovery and Reversal in Interactive Systems.* ACM Trans. Program. Lang. Syst., 6:1–19, January 1984.

[3] Berlage, T.: *A selective undo mechanism for graphical user interfaces based on command objects.* ACM Trans. Comput.-Hum. Interact., 1:269–294, September 1994.

[4] Berlage, T. und A. Genau: *From Undo to Multi-User Applications.* In: *Proceedings of the Vienna Conference on Human-Computer Interaction*, S. 213–224, London, UK, 1993. Springer-Verlag.

[5] Cass, A. und C. Fernandes: *Using Task Models for Cascading Selective Undo.* In: Coninx, K., K. Luyten und K. Schneider (Hrsg.): *Task Models and Diagrams for Users Interface Design*, Bd. 4385 d. Reihe *Lecture Notes in Computer Science*, S. 186–201. Springer Berlin/Heidelberg, 2007.

[6] Cass, A. G., C. S. T. Fernandes und A. Polidore: *An empirical evaluation of undo mechanisms.* In: *Proceedings of the 4th Nordic Conference on Human-Computer Interaction*, S. 19–27, New York, NY, USA, 2006. ACM.

[7] Choudhary, R. und P. Dewan: *A general multi-user undo/redo model.* In: *Proceedings of the 4th Conference on European Conference on Computer-Supported Cooperative Work*, S. 231–246, Norwell, MA, USA, 1995. Kluwer Academic Publishers.

[8] Grossman, T., J. Matejka und G. Fitzmaurice: *Chronicle: capture, exploration, and playback of document workflow histories.* In: *Proceedings of the 23nd annual ACM Symposium on User Interface Software and Technology*, S. 143–152, New York, NY, USA, 2010. ACM.

[9] Gutwin, C. und S. Greenberg: *A Descriptive Framework of Workspace Awareness for Real-Time Groupware*. Computer Supported Cooperative Work (CSCW), 11:411–446, 2002.

[10] Haller, M., J. Leitner, T. Seifried, J. R. Wallace, S. D. Scott, C. Richter, P. Brandl, A. Gokcezade und S. Hunter: *The NiCE Discussion Room: Integrating Paper and Digital Media to Support Co-Located Group Meetings*. In: *Proceedings of the 28th international Conference on Human factors in Computing Systems*, S. 609–618, New York, NY, USA, 2010. ACM.

[11] Hazemi, R. und L. Macaulay: *User requirements for undo support in CSCW*. In: *Proceedings of the HCI'95 Conference on People and Computers X*, S. 181–193, New York, NY, USA, 1995. Cambridge University Press.

[12] Kurlander, D. und S. Feiner: *A Visual Language for Browsing, Undoing, and Redoing Graphical Interface Commands*. In: *Visual Languages and Visual Programming*, S. 257–275, New York, NY, USA, 1990. Plenum Press.

[13] Kurlander, D. und S. Feiner: *A history of editable graphical histories*, S. 405–413. MIT Press, Cambridge, MA, USA, 1993.

[14] Li, R. und D. Li: *A Regional Undo Mechanism for Text Editing*. In: *Proceedings of the Workshop on Collaborative Editing Systems*, 2008. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.4754.

[15] Meng, C., M. Yasue, A. Imamiya und X. Mao: *Visualizing Histories for Selective Undo and Redo*. In: *Proceedings of the 3rd Asian Pacific Computer and Human Interaction*, S. 459–464, Washington, DC, USA, 1998. IEEE Computer Society.

[16] Mynatt, E. D., T. Igarashi, W. K. Edwards und A. LaMarca: *Flatland: new dimensions in office whiteboards*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, S. 346–353, New York, NY, USA, 1999. ACM.

[17] Nakamura, T. und T. Igarashi: *An application-independent system for visualizing user operation history*. In: *Proceedings of the 21st annual ACM Symposium on User Interface Software and Technology*, S. 23–32, New York, NY, USA, 2008. ACM.

[18] Prakash, A. und M. J. Knister: *Undoing actions in collaborative work*. In: *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*, S. 273–280, New York, NY, USA, 1992. ACM.

[19] Prakash, A. und M. J. Knister: *A framework for undoing actions in collaborative systems.* ACM Trans. Comput.-Hum. Interact., 1:295–330, December 1994.

[20] Ressel, M. und R. Gunzenhäuser: *Reducing the problems of group undo.* In: *Proceedings of the international ACM SIGGROUP Conference on Supporting Group Work*, S. 131–139, New York, NY, USA, 1999. ACM.

[21] Ressel, M., D. Nitsche-Ruhland und R. Gunzenhäuser: *An integrating, transformation-oriented approach to concurrency control and undo in group editors.* In: *Proceedings of the 1996 ACM Conference on Computer-Supported Cooperative Work*, S. 288–297, New York, NY, USA, 1996. ACM.

[22] Smith, J.: *WPF Apps with the Model-View-ViewModel Design Pattern.* MSDN Magazine, February:72–81, 2009.

[23] Spector, R. H.: *Visual Fields*, Kap. 116, S. 565–572. Butterworths, Boston, MA, USA, 1990.

[24] Sun, C.: *Undo any operation at any time in group editors.* In: *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work*, S. 191–200, New York, NY, USA, 2000. ACM.

[25] Sun, C. und D. Chen: *Consistency maintenance in real-time collaborative graphics editing systems.* ACM Trans. Comput.-Hum. Interact., 9:1–41, March 2002.

[26] Sun, C., X. Jia, Y. Zhang, Y. Yang und D. Chen: *Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems.* ACM Trans. Comput.-Hum. Interact., 5:63–108, March 1998.

[27] Sun, D. und C. Sun: *Operation context and context-based operational transformation.* In: *Proceedings of the 2006 20th anniversary Conference on Computer-Supported Cooperative Work*, S. 279–288, New York, NY, USA, 2006. ACM.

[28] Teitelman, W., J. Goodwin und D. Bobrow: *Interlisp reference manual.* Techn. Ber., Xerox Palo Alto Research Centers, 1978.

[29] Vitter, J.: *US & R: A new framework for redoing.* IEEE Software, 1(4):39–52, October 1984.

[30] Yang, Y.: *Undo support models.* International Journal of Man-Machine Studies, 28(5):457–481, 1988.

[31] Zhou, C. und A. Imamiya: *Object-based nonlinear undo model*. In: *Proceedings of the 21st International Computer Software and Applications Conference*, S. 50–55, Washington, DC, USA, 1997. IEEE Computer Society.