



Designing Representations for Digital Documents

Han Han

► To cite this version:

Han Han. Designing Representations for Digital Documents. Human-Computer Interaction [cs.HC]. Université Paris-Saclay, 2022. English. NNT : 2022UPASG025 . tel-03662229

HAL Id: tel-03662229

<https://theses.hal.science/tel-03662229>

Submitted on 9 May 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Designing Representations for Digital Documents

Conception de représentations pour les documents numériques

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat : Informatique

Graduate School : informatique et sciences du numérique

Référent : Faculté de sciences d'Orsay

Thèse préparée dans l'unité de recherche **Laboratoire Interdisciplinaire des Sciences du Numérique** (Université Paris-Saclay, CNRS, Inria),
sous la direction de **Michel BEAUDOUIN-LAFON**, Professeur

Thèse soutenue à Paris-Saclay, le 30 Mars 2022, par

Han HAN

Composition du Jury

Sarah COHEN-BOULAKIA Professeure, Université Paris-Saclay	Présidente
James HOLLAN Professeur, University of California San Diego	Rapporteur & Examineur
Yannick PRIÉ Professeur, Université de Nantes	Rapporteur & Examineur
Victoria BELLOTTI Chercheur UX Senior, Netflix	Examinatrice
Ken HINCKLEY Directeur de la recherche, Microsoft	Examineur
Michel BEAUDOUIN-LAFON Professeur, Université Paris-Saclay	Directeur de thèse

Titre : Conception de représentations pour les documents numériques

Mots clés : Interaction Humain-Machine, Design d'interaction, Utilisateur extrême, Interface graphique, Documents numériques, Modèles d'interaction

Résumé : Des millions d'utilisateurs travaillent à l'aide de documents afin d'effectuer leurs tâches quotidiennes, mais les interfaces utilisateurs n'ont pas fondamentalement changé depuis leur première conception à la fin des années 70. Les ordinateurs d'aujourd'hui sont utilisés par une grande variété d'utilisateurs pour réaliser un large éventail de tâches, ce qui interroge les limites des interfaces actuelles. Je soutiens qu'en se concentrant sur les utilisateurs extrêmes et en adoptant une perspective fondée sur des principes de conception, nous pouvons concevoir des représentations efficaces et flexibles pour soutenir le travail de connaissance lié aux documents.

J'étudie d'abord l'une des tâches les plus courantes, à savoir le traitement de texte dans le contexte des documents techniques. En nous concentrant sur les professionnels du droit, nous mettons en lumière les limites des logiciels de traitement de texte actuels. Les professionnels du droit doivent faire appel à leur mémoire pour gérer les dépendances et maintenir un vocabulaire cohérent dans leurs documentations. Pour résoudre ces problèmes, nous introduisons Textlets, des objets interactifs qui réifient les sélections de texte en éléments persistants. Nous présentons un prototype de preuve de concept démontrant plusieurs cas d'utilisation, notamment la recherche et le remplacement sélectifs, le comptage des mots et les mots alternatifs. L'évaluation observationnelle montre l'utilité et l'efficacité de Textlets, ce qui prouve la validité du concept.

Au cours de mon travail avec des professionnels du droit, j'ai été initié à la rédaction et au dépôt de brevets. Dans le processus de brevetage, les avocats rédigent des demandes de brevet qui décrivent une invention donnée. Les examinateurs de brevets étudient la demande et décident si un brevet peut lui être accordé. En collaboration avec l'Office européen des brevets, j'ai étudié le processus de recherche et de révision des examinateurs de brevets.

L'étude montre la nécessité de gérer le texte de plusieurs documents

à travers diverses activités interconnectées, tout en suivant manuellement leur provenance. Je prolonge Textlets pour créer Passages, des objets de sélection de texte qui peuvent être manipulés, réutilisés et partagés entre plusieurs outils. Deux études d'utilisateurs montrent que Passages facilitent les pratiques des professionnels et permettent une plus grande réutilisation des informations.

Ces deux projets ont conduit à un autre aspect important du travail intellectuel : la gestion des fichiers. Je me concentre sur les scientifiques, un autre exemple d'utilisateurs extrêmes, pour étudier leurs pratiques de gestion des documents. Les scientifiques travaillent avec une variété d'outils et ils ont des difficultés à utiliser le système de fichiers pour suivre et maintenir la cohérence entre des informations connexes mais distribuées. Nous avons créé FileWeaver, un système qui détecte automatiquement les dépendances entre les fichiers sans action explicite de l'utilisateur, suit leur historique et permet aux utilisateurs d'interagir directement avec les graphiques représentant ces dépendances et l'historique des versions. En rendant les dépendances entre fichiers visibles, FileWeaver facilite l'automatisation des flux de travail des scientifiques et des autres utilisateurs qui s'appuient sur le système de fichiers pour gérer leurs données.

Je réfléchis à mon expérience de conception et d'évaluation de ces représentations et propose trois nouveaux principes de conception : granularité, individualité et synchronisation.

Avec les résultats empiriques de ces utilisateurs extrêmes, la démonstration technologique de trois prototypes de preuve de concept et trois principes de conception, cette thèse démontre de nouvelles approches originales pour travailler avec des documents. Je soutiens qu'en adoptant une perspective fondée sur les principes et la théorie, nous pouvons contribuer à des concepts d'interface innovants.

Title : Designing Representations for Digital Documents

Keywords : Human-Computer Interaction, Interaction Design, Extreme User, Graphical User Interface, Digital Documents, Interaction Model

Abstract : Millions of users work with documents for their everyday tasks but their user interfaces have not fundamentally changed since they were first designed in the late seventies. Today's computers come in many forms and are used by a wide variety of users for a wide range of tasks, challenging the limits of current document interfaces. I argue that by focusing on extreme users and taking on a principled perspective, we can design effective and flexible representations to support document-related knowledge work.

I first study one of the most common document tasks, text editing, in the context of technical documents. By focusing on legal professionals, one example of extreme document users, we reveal the limits of current word processors. Legal professionals must rely on their memory to manage dependencies and maintain consistent vocabulary within their technical documents. To address these issues, we introduce Textlets, interactive objects that reify text selections into persistent items. We present a proof-of-concept prototype demonstrating several use cases, including selective search and replace, word count, and alternative wording. The observational evaluation shows the usefulness and effectiveness of textlets, providing evidence of the validity of the textlet concept.

During my work with legal professionals in the first project, I was introduced to the domain of patent writing and filling. In the patent process, patent attorneys write patent submissions that describe the invention created by the inventor. Patent examiners review the submission and decide whether the submission can be granted as a patent. In collaboration with a European Patent Office, I studied the patent examiners' search and review process. The study reveals the need to manage text from multiple documents across various interconnected activities, including searching, collecting, annotating, organizing, writing and reviewing, while manually tracking their provenance. I extend Textlets to create Passages, text selection objects that can be

manipulated, reused, and shared across multiple tools. Two user studies show that Passages facilitate knowledge workers practices and enable greater reuse of information.

These two projects led to another important aspect of knowledge work: file management. I focus on scientists, another example of extreme knowledge workers, to study their document management practices. In an age where heterogeneous data science workflows are the norm, instead of relying on more self-contained environments such as Jupyter Notebooks, scientists work across many diverse tools. They have difficulties using the file system to keep track of, re-find and maintain consistency among related but distributed information. We created FileWeaver, a system that automatically detects dependencies among files without explicit user action, tracks their history, and lets users interact directly with the graphs representing these dependencies and version history. By making dependencies among files explicit and visible, FileWeaver facilitates the automation of workflows by scientists and other users who rely on the file system to manage their data.

These three document representations rely on the same underlying theoretical principles: reification, polymorphism and reuse. I reflect on my experience designing and evaluating these representations and propose three new design principles: granularity, individuality and synchronization.

Together with the empirical findings from three examples of extreme users, technological demonstration of three proof-of-concept prototypes and three design principles, this thesis demonstrates fresh new approaches to working with documents, a fundamental representation in GUIs. I argue that we should not accept current desktop interfaces as given, and that by taking on a principled and theory-driven perspective we can contribute innovative interface concepts.

Synthèse en Français

Des millions d'utilisateurs travaillent à l'aide de documents afin d'effectuer leurs tâches quotidiennes, mais les interfaces utilisateurs n'ont pas fondamentalement changé depuis leur première conception à la fin des années 70 (Smith et al., 1982). Les ordinateurs d'aujourd'hui se présentent sous différentes formes et sont utilisés par une grande variété d'utilisateurs pour réaliser un large éventail de tâches, ce qui interroge les limites des interfaces actuelles. Je soutiens qu'en se concentrant sur les utilisateurs extrêmes et en adoptant une perspective fondée sur des principes de conception, nous pouvons concevoir des représentations efficaces et flexibles pour soutenir le travail de connaissance lié aux documents.

J'étudie d'abord l'une des tâches les plus courantes, à savoir le traitement de texte dans le contexte des documents techniques. En nous concentrant sur les professionnels du droit, un exemple d'utilisateurs extrêmes, nous mettons en lumière les limites des logiciels de traitement de texte actuels. Les professionnels du droit doivent faire appel à leur mémoire pour gérer les dépendances et maintenir un vocabulaire cohérent dans leurs documents techniques. Pour résoudre ces problèmes, nous introduisons les Textlets, des objets interactifs qui réifient les sélections de texte en éléments persistants. Nous présentons un prototype de preuve de concept démontrant plusieurs cas d'utilisation, notamment la recherche et le remplacement sélectifs, le comptage des mots et les mots alternatifs. L'évaluation observationnelle montre l'utilité et l'efficacité des Textlets, ce qui prouve la validité du concept.

Au cours de mon travail avec des professionnels du droit dans le cadre du premier projet, j'ai été initié à la rédaction et au dépôt de brevets. Dans le processus de brevetage, les avocats rédigent des demandes de brevet qui décrivent l'invention créée par l'inventeur. Les examinateurs de brevets étudient la demande et décident si un brevet peut lui être accordé. Je me suis intéressé à la manière dont les examinateurs analysent et étudient les documents techniques, en complément du processus de rédaction des documents. En collaboration avec une institution chargée de l'attribution de brevets, j'ai étudié le processus de recherche et de révision des examinateurs de brevets. L'étude montre la nécessité de gérer le texte de plusieurs documents à travers diverses activités interconnectées, notamment la recherche, la collecte, l'annotation, l'organisation, la rédaction et la révision, tout en suivant manuellement leur provenance. Je développe

les Textlets pour créer des Passages, des objets de sélection de texte qui peuvent être manipulés, réutilisés et partagés entre plusieurs outils. Deux études d'utilisateurs montrent que les Passages facilitent les pratiques des professionnels et permettent une plus grande réutilisation des informations.

Ces deux projets ont conduit à un autre aspect important du travail intellectuel : la gestion des fichiers. Je me concentre sur les scientifiques, un autre exemple d'utilisateurs extrêmes, pour étudier leurs pratiques de gestion des documents. À une époque où les flux de travail hétérogènes sont la norme en science des données, au lieu de s'appuyer sur des environnements plus autonomes tels que Jupyter Notebooks, les scientifiques travaillent avec une variété d'outils. Ils ont des difficultés à utiliser le système de fichiers pour suivre, retrouver et maintenir la cohérence entre des informations connexes mais distribuées. Nous avons créé FileWeaver, un système qui détecte automatiquement les dépendances entre les fichiers sans action explicite de l'utilisateur, suit leur historique et permet aux utilisateurs d'interagir directement avec les graphiques représentant ces dépendances et l'historique des versions. En rendant les dépendances entre fichiers explicites et visibles, FileWeaver facilite l'automatisation des flux de travail des scientifiques et des autres utilisateurs qui s'appuient sur le système de fichiers pour gérer leurs données.

Ces trois représentations de documents reposent sur les mêmes principes théoriques : réification, polymorphisme et réutilisation. Je réfléchis à mon expérience de conception et d'évaluation de ces représentations et propose trois nouveaux principes de conception : granularité, individualité et synchronisation.

Avec les résultats empiriques de trois exemples d'utilisateurs extrêmes, la démonstration technologique de trois prototypes de preuve de concept et trois principes de conception, cette thèse démontre de nouvelles approches originales pour travailler avec des documents, une représentation fondamentale dans les interfaces graphiques. Je soutiens que nous ne devrions pas accepter les interfaces de bureau actuelles pour acquises, et qu'en adoptant une perspective fondée sur les principes et la théorie, nous pouvons contribuer à des concepts d'interface innovants.

Abstract

Millions of users work with documents for their everyday tasks but their user interfaces have not fundamentally changed since they were first designed in the late seventies. Today's computers come in many forms and are used by a wide variety of users for a wide range of tasks, challenging the limits of current document interfaces. I argue that by focusing on extreme users and taking on a principled perspective, we can design effective and flexible representations to support document-related knowledge work.

I first study one of the most common document tasks, text editing, in the context of technical documents. By focusing on legal professionals, one example of extreme document users, we reveal the limits of current word processors. Legal professionals must rely on their memory to manage dependencies and maintain consistent vocabulary within their technical documents. To address these issues, we introduce Textlets, interactive objects that reify text selections into persistent items. We present a proof-of-concept prototype demonstrating several use cases, including selective search and replace, word count, and alternative wording. The observational evaluation shows the usefulness and effectiveness of textlets, providing evidence of the validity of the textlet concept.

During my work with legal professionals in the first project, I was introduced to the domain of patent writing and filling. In the patent process, patent attorneys write patent submissions that describe the invention created by the inventor. Patent examiners review the submission and decide whether the submission can be granted as a patent. In collaboration with a European Patent Office, I studied the patent examiners' search and review process. The study reveals the need to manage text from multiple documents across various interconnected activities, including searching, collecting, annotating, organizing, writing and reviewing, while manually tracking their provenance. I extend Textlets to create Passages, text selection objects that can be manipulated, reused, and shared across multiple tools. Two user studies show that Passages facilitate knowledge workers practices and enable greater reuse of information.

These two projects led to another important aspect of knowledge work: file management. I focus on scientists, another example of extreme knowledge workers, to study their document management prac-

tices. In an age where heterogeneous data science workflows are the norm, instead of relying on more self-contained environments such as Jupyter Notebooks, scientists work across many diverse tools. They have difficulties using the file system to keep track of, re-find and maintain consistency among related but distributed information. We created FileWeaver, a system that automatically detects dependencies among files without explicit user action, tracks their history, and lets users interact directly with the graphs representing these dependencies and version history. By making dependencies among files explicit and visible, FileWeaver facilitates the automation of workflows by scientists and other users who rely on the file system to manage their data.

These three document representations rely on the same underlying theoretical principles: reification, polymorphism and reuse. I reflect on my experience designing and evaluating these representations and propose three new design principles: granularity, individuality and synchronization.

Together with the empirical findings from three examples of extreme users, technological demonstration of three proof-of-concept prototypes and three design principles, this thesis demonstrates fresh new approaches to working with documents, a fundamental representation in GUIs. I argue that we should not accept current desktop interfaces as given, and that by taking on a principled and theory-driven perspective we can contribute innovative interface concepts.

Acknowledgments

When I look back at the past three years, I feel fortunate that I've been able to finish this thesis in a company of an incredibly generous, supportive, and inspiring group of people.

To Michel, I offer my heartfelt gratitude. It is from you that I learned to pick the right research question, think deeply, challenge the obvious and communicate the key messages. I have learned that what it takes to do great work: a vision, high standard and hard work. These things have pushed and will always push me to achieve the best as I can, to do great work. I don't think I could ever learn enough of you. Every conversation with you, I learn something.

To Wendy, I learned so much from you. Thank you for sharing your vast knowledge about conducting research and all your experience. I will never forget the energy you have when we talked about research and that energy is infectious.

Thank you my jury members : Sarah, Jim, Yannick, Victoria and Ken, for your interest, support and advice for my research. I appreciate it. Thank you Victoria for your generous mentorship and guidance to my career. I still vividly remember your can-do attitude when I ask you to be an examiner of my thesis. Thank you for taking on this challenge.

To all my collaborators, Miguel, Julien, Enjung, Junhang, Raphael, Alex. I am lucky to have the opportunity to work with you. We worked hard and your contribution to this thesis is enormous. To other colleagues on the same boat, Viktor, Liz, Miguel and Téo, we all made it. To the previous generation of Ph.D students, Marianela, Carla, Stacy, Germán, Nolwenn, Jean-Philippe, Michael and Philip, thank you all for the sharing of Ph.D experience and advice with me. The advice guided me through my own journey. To the new members of ExSitu team, Tove, Alex, Arthur, Camille, Capucine, Anna and Wissal, best of luck for your journeys. I am sure they will be uniquely interesting.

Thank you Junhang for showing me design and showing me how to use my eyes to see good design. With

you, we were able to make the vision a reality. Working with you is one of my most creative periods in the past three years. Thank you for supporting me to the end of this journey by being my technical support for this defense. You are the best producer. We need to make a movie together.

To Domenico, the director of European Patent Office. Thank you for accepting my research proposal and your generous support and trust to a young researcher. I will always remember the first time we met in the business meeting. You brought sense of humour to the seriousness of the meeting. It reminds me of what John Cleese said something about "being funny does not make the business we are discussing less serious". I truly believe that.

And my girlfriend Mylène. You let me see the exciting parts of life outside of research. You bring the sunshine to me. I bring HCI to you... You have this unique ability to see my research in a crystal clear way. Thank you for being just you.

And lastly, to my mum and dad, I know that you guys are always there for me, as my parents who give me love but also as friends who give me advice about life. You will always be my mentors no matter how many doctor degrees I have.

Thank you all very much.

This work was supported by European Research Council (ERC) grant No 695464 "ONE: Unified Principles of Interaction.

Contents

1	Introduction	1
1.1	Thesis Statement	3
1.2	Research Approach	3
1.3	Contribution	4
1.4	Organization	5
1.5	Publications	6
1.6	On The Use Of The Pronoun 'We'	7
2	Background	9
2.1	Representation and Manipulation	9
2.2	Personal Information Management	14
2.3	Document Software Systems	17
2.4	Interaction Frameworks and Models	20
3	Representation for Document Editing	23
3.1	Context	23
3.2	Related Work	24
3.3	Interview with Legal Professionals	26
3.4	Results and Discussion	28
3.5	Textlets Concept	33
3.6	Textlets User Interface	40
3.7	Structured Observation	44
3.8	Results and Discussion	45
3.9	Conclusion	48
4	Representation for Document Analysis	51
4.1	Context	52
4.2	Related Work	53
4.3	Study 1: Interviews with Patent Examiners	56
4.4	Results and Discussion	57
4.5	Study 2: Interviews with Scientists	62
4.6	Results and Discussion	63

4.7	Passages Concept	65
4.8	The User Experience	66
4.9	Passages User Interface	70
4.10	Study 3: Design Walkthrough with Patent Examiners . .	75
4.11	Results and Discussion	77
4.12	Study 4: Structured Observation with Scientists	78
4.13	Results and Discussion	80
4.14	Conclusion	85
5	Representation for Document Management	87
5.1	Context	88
5.2	Related Work	89
5.3	Interviews with Scientists	92
5.4	Results and Discussion	92
5.5	FileWeaver User Interface	99
5.6	The User Experience	101
5.7	System Implementation	104
5.8	Discussion and Evaluation	108
5.9	Conclusion	110
6	Design Principles	113
6.1	Granularity	113
6.2	Individuality	116
6.3	Synchronization	118
6.4	Summary	121
7	Conclusion and Future Work	123
7.1	Conclusion	123
7.2	Future Work	125

List of Figures

- 1.1 The Star Interface. Source: Smith et al. (1982) 1
- 1.2 The scrollbar to navigate a document has not evolved from Xerox Star (1981) to Windows 10 (2015). (Arguably, MacOS changed the scrollbars significantly by reversing the direction of movement related to the users motion. Also, iOS popularized inertial scrolling and the disappearance of scrollbar.) Source: <https://scrollbars.matoseb.com/> 3
- 1.3 Illustration of the thesis organization. The main chapters focuses on 1) single document editing, 2) multiple documents analysis and 3) multiple documents management. 6
- 2.1 Selected examples of evolving writing surfaces and tools. Images from a wonderful BBC documentary about the history of writing. Source: <https://www.bbc.co.uk/programmes/m000mtml> 9
- 2.2 A reproduced page of Leonardo da Vinci's notebook. Source: https://commons.wikimedia.org/wiki/File:Reproduction_of_page_from_notebook_of_Leonardo_da_Vinci_LCCN2006681086.jpg 10
- 2.3 If Sumerians designed the icon, it may look like this. 10
- 2.4 Evolution of document icon. Source: https://commons.wikimedia.org/wiki/File:Evolution_of_the_document_icon_shape.jpg 10
- 2.5 Perceptual properties of visual and auditory systems. Left is from *Semiology of Graphics* (Bertin, 1983) and right is from SonicFinder (Gaver, 1989) 11
- 2.6 A simplified definition of representation 12
- 2.7 Example of Denis Diderot's encyclopedie. Source: https://commons.wikimedia.org/wiki/File:Defehrt_epinglier_pl2.jpg 13
- 2.8 Scott McCloud shows how comics represent invisible things such as smell. Image taken from (McCloud, 1993) 13
- 2.9 Bill Verplank's Sketch interaction design. Source: <http://billverplank.com/Ciid/IDSketch.pdf> 13

2.10 Screenshot from the video where Ron Kaplan and Allen Newell tried to use the copier. Source: https://www.youtube.com/watch?v=DUwXN01ARYg	14
2.11 Biologist's paper lab notebook. Source: Tabard et al. (2008)	15
2.12 Field biologist's paper lab notebook. Source: Yeh et al. (2006)	15
2.13 Jupyter Notebook. Source: https://jupyter.org/	16
2.14 Bush's Memex	17
2.15 Engelbart's NLS. (See demo: https://www.youtube.com/watch?v=qI8r8D46J0Y&list=PL76DBC8D6718B8FD3&index=9)	17
2.16 Nelson's Xanadu. See demo: https://www.youtube.com/watch?v=En_2T7KH6RA	17
2.17 Cut-paste in Gypsy. See demo: https://www.youtube.com/watch?v=Dhmz68CII9Y	18
2.18 Dourish et al.'s Presto	18
2.19 Bederson and Hollan's Pad++	18
2.20 Robertson et al.'s Data Mountain	19
2.21 Fertig et al.'s Lifestreams	19
2.22 Oleksik et al.'s TAGtivity	19
2.23 Jacob et al.'s Reality-Based Interaction	20
2.24 Beaudouin-Lafon's Instrumental Interaction	21
3.1 Editing a document	23
3.2 Example of technical documents: patent (top), technical manual (middle), contract (bottom)	24
3.3 Miller and Marshall's cluster-based search and replace	25
3.4 Beaudouin-Lafon's instrumental search and replace	25
3.5 Oney and Brandt's Codelets	26
3.6 Kery et al.'s Variolite	26
3.7 Ko and Myers's Barista	26
3.8 Critical object interviews in participants' workplace.	27
3.9 A collection of story portraits based on the analysis	28
3.10 Which term to use?	28
3.11 Summary of Invention and claims are linked	29
3.12 Patent claims use three different numbering systems (left). Patent illustration (right).	30
3.13 Normal and automatic numbering. The user cannot tell them apart unless she selects them	30
3.14 Participant jumps at different document locations.	31
3.15 A story portrait illustrating how one participant consolidate changes from various collaborators as it from one party	32
3.16 Transient selection. a,b) the user selects a piece of text; c) she applies a set of command to it; d) the appearance of the text has changed based on the command; e) as soon as she selects another text, the previous selection is gone.	34

3.17	Hill et al. (1992)'s Edit and Read Wear	36
3.18	Microsoft Word's word count with a modal dialog	38
3.19	A mock-up of <i>timelets</i> . The user can record her voice and see the time she takes to read the text.	38
3.20	Dragicevic et al. (2019)'s explorable multiverse analyses	38
3.21	Goffin et al. (2017)'s word-scale graphics embedded in text documents	38
3.22	Bret Viktor's computed text. Source: What can a technologist do about climate change	39
3.23	First prototype with the side panel showing a <i>variantlet</i> , a <i>countlet</i> and a <i>numberlet</i> containing a numbered item and a reference, and their visualization in the document.	40
3.24	<i>Countlet</i> : a textlet for counting words.	41
3.25	<i>Variantlet</i> : a textlet for editing local versions.	41
3.26	<i>Numberlet</i> : a textlet for numbering and referencing.	42
3.27	<i>Searchlet</i> : a textlet for searching and replacing text.	43
3.28	A user think-aloud in the observational study	44
3.29	A timeline of study procedure	45
3.30	Adding the design of individual replacement, undo and ignore, based on participants feedback	45
4.1	Analyze multiple documents	51
4.2	O'Hara et al. (1998)'s model of various document related activities of library users.	52
4.3	Hinckley et al. (2012)'s Gather Reader	54
4.4	Tashman and Edwards (2011b)'s Liquid Text	54
4.5	Subramonyam et al. (2020)'s TexSketch	54
4.6	Hearst (1995)'s TileBar	55
4.7	schraefel et al. (2002)'s HunterGather	55
4.8	Liu et al. (2019)'s Unakite	55
4.9	Gotz (2007)'s Scratchpad	55
4.10	Patent examiners engage in a series of document-related activities when searching for prior art, including formulating search terms; reading and annotating documents; and writing and reviewing their own and other examiners' reports.	58
4.11	Tables created by P2 and P5 for their literature reviews.	64
4.12	Passages reifies text selections into interactive objects that can be manipulated, reused and shared across applications	66
4.13	The Viewer lets Emma collect interesting text as passages, by selecting them and clicking the "Passage" button.	67
4.14	The Table lets Emma drag and drop collected passages and organize them into rows and columns	68

- 4.15 The Searcher lets Emma iteratively search for more documents by specifying multiple search terms, and keeping track of her search history. 68
- 4.16 The Editor helps Emma communicate her findings, while preserving the provenance of each passage as she writes her report. 69
- 4.17 (a) Fluid transitions across applications through drag-and-drop of passages between windows; (b) Reuse of content and structure from the Table directly into the Editor 70
- 4.18 The Reader lets Alex read the report while still easily accessing the source documents to verify the claims 71
- 4.19 *Passages* overall user interface with six applications. 72
- 4.20 One participant's critiques and suggestions for seven interaction points demonstrating the key features of *Passages* in the generative walkthrough workshop. 75
- 4.21 A timeline of the design walkthrough 76
- 4.22 Having both source and reference side by side. 77
- 4.23 Document set and application choice are counterbalanced. 79
- 4.24 Interacting with *Passages*: Table (a) and report (c) created by participants. (b) P4 drag-dropped a newly created passage directly into the *Table*. 81
- 4.25 Participants largely reuse the passages collected. 81
- 4.26 In current design, dropping a passage in the editor (a) inserts the whole raw passage (b). 84

- 5.1 Manage multiple documents. 87
- 5.2 Guo (2012)'s typical data science workflow 88
- 5.3 Information locked in its applications. 89
- 5.4 Tabard et al. (2008)'s research shows how biologists juggles complex mix of paper and computer-based information. 90
- 5.5 A Jupyter notebook combines code, text and visualization in a single notebook format. Image taken from (Rule et al., 2018b). 90
- 5.6 Karlson et al. (2011)'s versionset: folder view with indentation (top) and graph view showing file relationship (bottom) 91
- 5.7 Sourcetree's timeline interface for version control. Source: <https://www.sourcetreeapp.com/> 91
- 5.8 Scientists take advantages of different artifacts such as paper, whiteboard, lab notebook and Jupyter notebook 93
- 5.9 Hoard information such as whiteboard photos, paper and script versions. 94
- 5.10 Various strategies for re-finding, e.g. summarizing, linking and combined with other documents. 95
- 5.11 Rapid lifecycle of various documents in scientists' workflow. 97

5.12	The <i>FileWeaver</i> User Interface. The Folder View is a standard file browsing window. Users can add files together with their dependencies to the Graph View by clicking “Add File”. The Graph View was displayed by selecting <code>main.tex</code> in the Folder View, and shows the dependency graph for that file. The History View shows the history of versions of <code>main.tex</code>	99
5.13	Graph View with a copy (green arrow).	100
5.14	A morph file	100
5.15	History View.	100
5.16	Selected user interaction to detect and maintain file dependencies. (See text and video for a complete interaction)	101
5.17	Selected user interaction to manage variants of files. (See text and video for a complete interaction)	102
5.18	Selected user interaction to manage history and version of a file. (See text and video for a complete interaction)	103
5.19	Selected user interaction to easily share all related files. (See text and video for a complete interaction)	103
5.20	Each file managed by <i>FileWeaver</i> has a cookbook page, in a hidden folder called <code>cookbook</code> . A cookbook page also holds the version control repository.	106
6.1	Text is treated as a character in a word processor (left) but as a graphic in a font design software. Image taken from: Microsoft Word and FontArk (Source: https://fontark.net/farkwp/).	114
6.2	Rivière et al. (2019)’s MoveOn interface with short clips representing a digestible movement.	115
6.3	Variantlets	115
6.4	A morph represents a group of similar files.	116
6.5	Interactions to manipulate groups of objects in Draco (Kazi et al., 2014), DataInk (Xia et al., 2018) and Textlets.	117
6.6	Synchronization between textlets in the side panel and the main text	119
6.7	Synchronization between the Graph View and the Folder View	120
7.1	Textlets	124
7.2	Passages	124
7.3	FileWeaver	124

List of Tables

3.1	How different behaviors of textlet address some issues and challenges observed in the interview	35
4.1	How different features of <i>Passages</i> address the issues found in Studies 1 and 2.	75
5.1	Six issues observed in the interview study.	98
5.2	How different features of <i>FileWeaver</i> address issues observed in the interview study	108
6.1	Questions that can be asked to apply the concepts and principles in this chapter analytically, critically and constructively.	121
7.1	Thesis overview with three aspects of document-related knowledge work.	123

1

Introduction

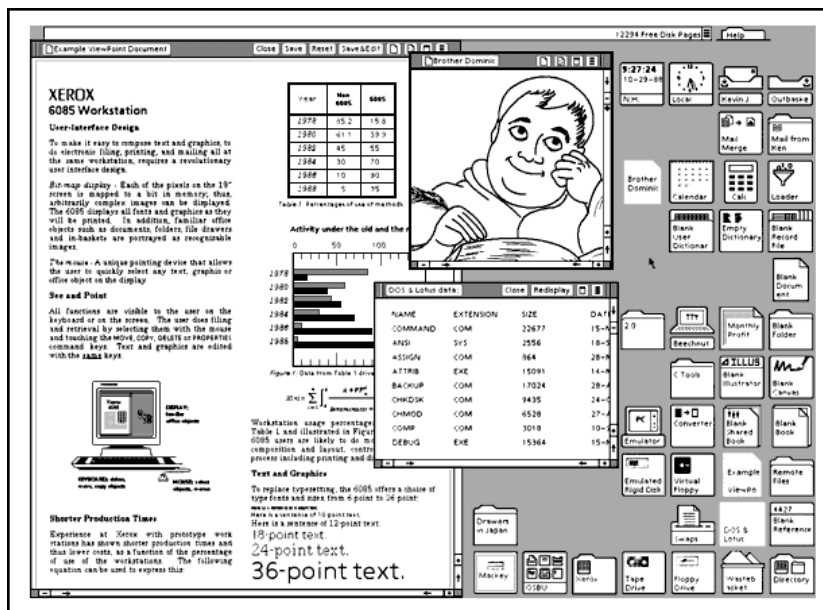


Figure 1.1. The Star Interface.
Source: Smith et al. (1982)

Today's computer interfaces are based on principles and conceptual models created in the late seventies. The Xerox Star (Smith et al., 1982), which pioneered today's graphical user interfaces (GUIs), made use of "the Desktop metaphor" with a real office, featuring the WIMP (windows, icons, menus, pointer) paradigm. Familiar office objects, such as documents, folders and file drawers, are represented as small pictures or icons. The content of the icons are represented in a larger form called "window". One of the fundamental representations is the

document, as the Star interface designers emphasize its importance:

The document is the heart of the world, and unifies it. (Johnson et al., 1989)

This emphasis is based on the assumption that “*the primary use of the system is to create and maintain documents.*” (Johnson et al., 1989). A physical document is represented as an icon that resembles the appearance of a typical document. The desk is represented as a 2D workspace called “*the Desktop*” where documents are displayed.

These powerful document representations were revolutionary but seldom evolved since then. If we look at the scrollbar for example (Fig. 1.2), we can see that they are based on the same logic and provide the same interaction. Today, millions of users work with documents for a variety of tasks. Do these representation still satisfy today users’ needs? Maybe not.

First, today’s document editing tools are “*bloated*” with hundreds of features (McGrenere et al., 2002), making them harder to use. Despite the development of direct manipulation (Shneiderman, 1983), users still rely on the manipulation of additional user interface elements such as menus and dialog boxes to achieve simple tasks, resulting in cumbersome and indirect interaction with the objects of interest (Beaudouin-Lafon, 2000). How can we design new representations while preserving simplicity of interaction? Second, today’s users are overloaded with information (Whittaker and Sidner, 1996) from various sources and in different formats. They often need to work with multiple documents (Tashman and Edwards, 2011a) using multiple separated applications (Oleksik et al., 2012). How do they keep track of this complex information and manage multiple applications?

At the same time, researchers have explored new Post-WIMP ¹ interaction theories and principles for designing new interfaces. For example, Beaudouin-Lafon (2000)’s *Instrumental Interaction* model extends and generalizes the principles of direct manipulation, and is operationalized by three design principles (Beaudouin-Lafon and Mackay, 2000): Reification turns commands into first class objects or instruments; polymorphism applies instruments to different types of objects; and reuse makes both user input and system output accessible for later use. How can we apply modern interaction design theories and principles to create alternative representations?

¹ <https://en.wikipedia.org/wiki/Post-WIMP>

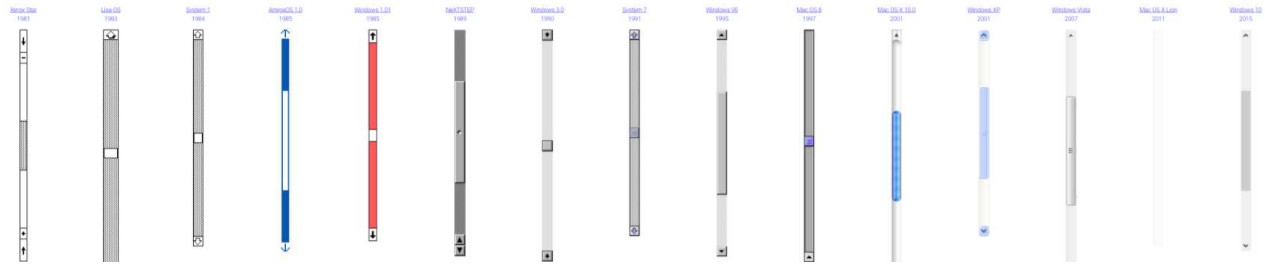


Figure 1.2. The scrollbar to navigate a document has not evolved from Xerox Star (1981) to Windows 10 (2015). (Arguably, MacOS changed the scrollbars significantly by reversing the direction of movement related to the users motion. Also, iOS popularized inertial scrolling and the disappearance of scrollbar.) Source: <https://scrollbars.matoseb.com/>

1.1 Thesis Statement

I argue that by focusing on extreme users and taking on a principled perspective, e.g. to represent the object of interest in the users' mind into manipulable interface elements, we can design effective and flexible representations to support document-related knowledge work.

1.2 Research Approach

Triangulation

Mackay and Fayard (Mackay and Fayard, 1997) pointed out that “*HCI cannot be considered a pure natural science because it studies the interaction between people and artificially-created artifacts, rather than naturally-occurring phenomena, which violates several basic assumptions of natural science.*” This thesis follows this triangulation framework, interleaving observation, design and theory. I use a wide variety of research methods from other disciplines including critical incident interviews (Mackay, 2002), interactive thread (Mackay, 2004), participatory design workshops (Mackay, 2002), technology probes (Hutchinson et al., 2003), generative walkthroughs (Lottridge and Mackay, 2009) and structured observations (Garcia et al., 2014).

Extreme User Innovation

Another approach I applied is the lead user method. Given the fact that the topic of digital documents have been extensively researched, I deliberately chose to focus on the *Extreme User*. The concept of *Extreme User* is similar to von Hippel (1986)'s *Lead User* because both of them face extreme problems and have extreme needs. The difference is that *Lead User* is leading in respect to the market trend and the goal is to forecast needs for marketing research as von Hippel (1986) points out: “*users whose present strong needs will become general in a marketplace*

months or years in the future." My goal is not to identify future markets for a specific trend. During the thesis, I have discovered many user innovations and found them extremely valuable and useful for both understanding the underlying problems and for design. For example, one patent engineer has developed a prototype to help him keep track of the reference numbers in a patent document. This user innovation was also disseminated to several colleagues of his because they share similar problems.

Reflecting on the discovery of these user innovations, although I did not intend to "find" the "right" extreme users, the way I discovered these user innovations is similar to Von Hippel's method of network-ing ². That is, by asking participants "Who do you know that has more extreme problems?", I "crawl" towards the user innovators. I also found out that innovative users also share some common characteristics: 1) extreme needs, 2) unique knowledge or understanding of their work and, 3) technical or design skills (so that they can build prototypes). I believe the combination of these characteristics influences the user innovation.

² MIT OpenCourseWare: <https://youtu.be/31iUEuwi740>

As a designer, this observation makes me realize the danger of a single-minded view on what *user* is. The user is not a single person; it is a group of people who have different working practices, knowledge of their work and levels of technical skills. The user does not just have problems and needs that wait to be understood by designers, they have incentives and can innovate for themselves. I believe it is the openness and willingness to learn from users that can bring together both knowledge and skills of user and designer, to achieve the design purpose. In relation to market research, Von Hippel shows four types of users: lead users, early adopters, routine users and laggards. In this thesis, I have encountered all of them. The identification and distinction of these users helps me to bring their different values throughout the design process. These user-focused approaches and methods are the foundation of this thesis.

1.3 Contribution

This thesis provides empirical findings from multiple studies, technical contributions in the form of functional prototypes, and theoretical contributions that introduce design principles.

Empirical Contributions: Through the study of three groups of knowledge workers, I found that

- legal professionals must rely on their memory to manage dependencies and maintain consistent vocabulary within their technical documents;
- patent examiners and scientists need to manage text from multiple documents across various interconnected activities, including searching, collecting, annotating, organizing, writing and reviewing, while manually tracking their provenance; and
- computational scientists have difficulties using the file system to keep track of, re-find and maintain consistency among related but distributed information.

Technical Contributions: I designed or contributed to the design of

- *Textlets*, interactive objects that reify text selections into persistent items;
- *Passages*, interactive objects that can be manipulated, reused and shared across multiple tools while maintaining their provenances; and
- *FileWeaver*, a system that automatically detects dependencies among files without explicit user action, tracks their history, and lets users interact directly with the graphs representing these dependencies and version history.

Theoretical Contribution: I propose

- three complementary design principles for creating new representations: granularity, individuality, synchronization.

1.4 Organization

Chapter 2 presents the definition of representation and manipulation through examples. I review the research area of personal information management and influential document software systems. I also review two new interaction models.

Chapter 3 focuses on document editing and presents the design and evaluation of *Textlets* with legal professionals.

Chapter 4 focuses on document analysis and presents the design and evaluation of *Passages* with patent examiners and scientists.

Chapter 5 focuses on document management and presents the design of *FileWeaver* with scientists.

Chapter 6 proposes three design principles when creating new presentations, generated from the design process and the lessons learned from the thesis.

Chapter 7 concludes the thesis with a summary of the main contributions and directions for future research.

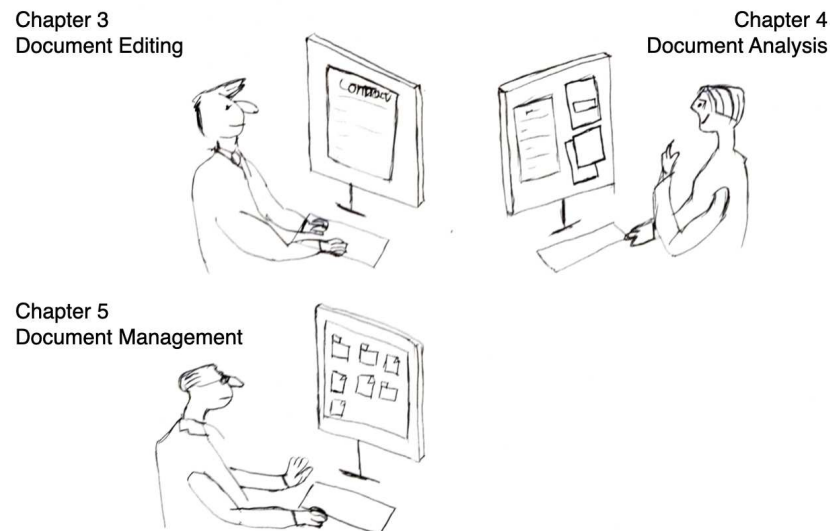


Figure 1.3. Illustration of the thesis organization. The main chapters focuses on 1) single document editing, 2) multiple documents analysis and 3) multiple documents management.

1.5 Publications

Some ideas and figures appeared previously in the following publications:

Chapter 3 Han L. Han, Miguel A Renom, Wendy E. Mackay, Michel Beaudouin-Lafon (2020). Textlets: Supporting Constraints and Consistency in Text Documents. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20).

DOI: <https://doi.org/10.1145/3313831.3376804>

Video: <https://youtu.be/9xD1hFVsKU>

Chapter 4 Han L. Han, Junhang Yu, Alexandre Ciorascu, Raphael Bournet, Wendy E. Mackay, Michel Beaudouin-Lafon (2022). Passages: Reading and Interacting with Text Across Documents. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI'22).

DOI: <https://doi.org/10.1145/3491102.3502052>

Video: <https://youtu.be/aLC2GVVltl0>

Chapter 5 Julien Gori, Han L. Han, Michel Beaudouin-Lafon (2020). FileWeaver: Flexible File Management with Automatic Dependency Tracking. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST'20).

DOI: <https://doi.org/10.1145/3379337.3415830>

Video: <https://youtu.be/PrcuF1MG1to>

Han L. Han (2020). Designing Representations for Digital Documents. In Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST'20 Doctoral Symposium).

DOI: <https://doi.org/10.1145/3379350.3415805>

1.6 On The Use Of The Pronoun 'We'

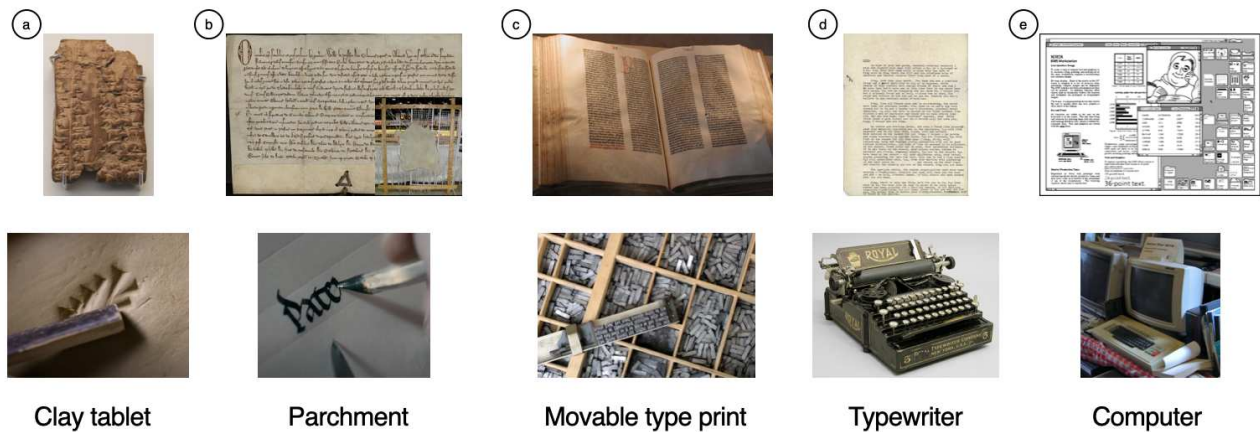
The core research projects in this thesis were highly collaborative. Upon reflection, I feel both lucky and grateful for my collaborator's contributions, efforts and support throughout the thesis. I have learned a lot from them and am greatly indebted to them. In recognition of the collaborative nature of this thesis, and for ease of reading, I thus use the pronoun 'we' when describing collaborative parts of this thesis and use "I" when it is done by myself.

2

Background

This chapter presents the definition of representation and manipulation in interaction design through examples. I represent key research literature in personal information management (PIM) and influential document systems in a chronological order. I also describe interaction frameworks and models.

2.1 Representation and Manipulation



About 5000 years ago in Mesopotamia (present-day Iraq), the Sumerians developed the first city states. The city dwellers felt the need for a

Figure 2.1. Selected examples of evolving writing surfaces and tools. Images from a wonderful BBC documentary about the history of writing. Source: <https://www.bbc.co.uk/programmes/m000mtml>

kind of record keeping and developed the first form of document. It is a clay tablet with symbols representing numbers and small stylised pictures representing commodities, created by a reed stylus (Fig. 2.1, a).

Since then, we started to see a rich history of writing, along with an evolution of writing surfaces, tools and technologies. Egyptians first invented papyrus as a writing surface. Later on, Europeans replaced it by locally produced parchment, usually made of untanned animal skins (Fig. 2.1, b). The documents produced by parchment were extremely durable but were expensive to make and only accessible to the nobles. In 1448, Johannes Gutenberg, a German goldsmith, introduced movable-type printing, largely speeding up the process of putting ink on paper (Fig. 2.1, c). What he did ushered in the modern period of human history, including Renaissance, Scientific Revolution, knowledge-based economy and mass communication. Leonardo da Vinci surely took advantage of the paper at that time and used it as a canvas for his ideas and thoughts (Fig. 2.2).

As time travels to 1874, we started to type the alphabet on paper using a typewriter using a QWERTY keyboard (Fig. 2.1, d). This lasted for one century. In the 1980s, the invention of the graphical user interface at Xerox PARC lowered the learning curve of using a computer, making it accessible to the masses (Fig. 2.1, e). Instead of typing commands that have to be learned, people interact with the information through graphical representations such as windows, icons, menus and pointer (WIMP).

Representation

If we take a look at the first document 5000 years ago and the current document in the graphical user interface, one commonality is that they both use something to represent something. The first document is a clay tablet with symbols representing numbers and small stylised pictures representing commodities (Fig. 2.3). In the Star interface, a physical document is represented as an icon that resembles the appearance of that document. The location of the document is represented in a 2D space, the Desktop. A group of properties of the document are represented in graphical forms called property sheets. The length of the document is represented as a scrollbar. These representations are powerful because they allow users to interact with the information in a way that is familiar and meaningful to them. Herbert Simon understands the importance of representations as he wrote in his book *The Sciences of the Artificial* (Simon, 1996):

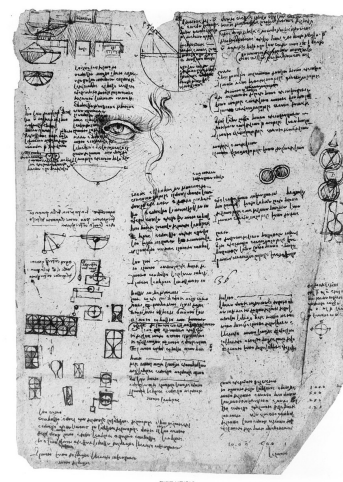


Figure 2.2: A reproduced page of Leonardo da Vinci's notebook. Source: https://commons.wikimedia.org/wiki/File:Reproduction_of_page_from_notebook_of_Leonardo_da_Vinci_LCCN2006681086.jpg



Figure 2.3: If Sumerians designed the icon, it may look like this.

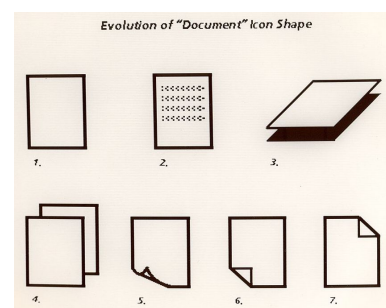
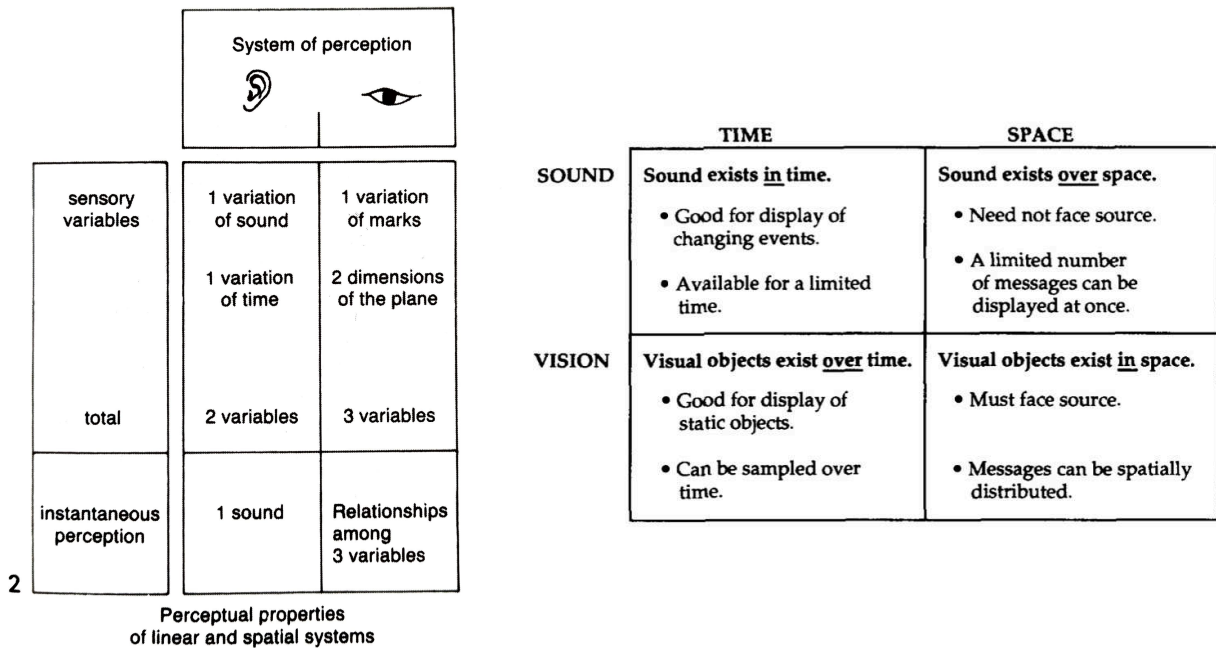


Figure 2.4: Evolution of document icon. Source: https://commons.wikimedia.org/wiki/File:Evolution_of_the_document_icon_shape.jpg

This view can be extended to all of problem solving. *Solving a problem simply means representing it so as to make the solution transparent.*¹ If the problem solving could actually be organized in these terms, the issue of representation would indeed become central. But even if it cannot if this is too exaggerated a view a deeper understanding of how representations are created and how they contribute to the solution of problems will become an essential component in the future theory of design. [Chapter 5, Page 132]

¹ The text is not emphasized in the original book. I added the emphasis.



So what is a representation? In semiology, De Saussure (2011) proposes that signs are made up of two parts: *signified* and *signifier*. Signified refers to the concept or object that is represented; signifier is the form of a sign. Another similar concepts for *signified* and *signifier* could be: content and form, meaning and appearance, information and representation. Charles Sanders Peirce (Atkin, 2013) adds that signs can be defined as belonging to one of three categories: *icon*, *index*, or *symbol*. An *icon* has a resemblance to the signified. An *index*, such as a clock or thermometer, correlates to an object by presenting a quality of an object. A *symbol* has no resemblance between the signifier and the signified. The connection between them must be learned. In interface design, Gaver (1989) introduced another categorization: *symbolic*, *metaphorical* and *iconic*. The one different category *metaphorical* makes use of similarities between *signified* and *signifier*, as he clarified, “an icon does not imply a literal pictorial, or recorded mapping, instead its characteristics are casually related to the things it represents”.

Figure 2.5. Perceptual properties of visual and auditory systems. Left is from *Semiology of Graphics* (Bertin, 1983) and right is from *SonicFinder* (Gaver, 1989)

Signs can communicate through any of the senses such as visual, auditory, tactile, olfactory, or gustatory. For example, the study of visual information is about communicating the sign through visual, such as Jacques Bertin's *Semiology of Graphics* (1983). The study of sonification is about communicating sign through our auditory sense, such as Gaver (1989)'s *SonicFinder*.

What are the *things* that can be represented? Tversky (2015) examines historical artifacts of information and finds that they all depict people, animals, things, space, place, time, events and numbers. One characteristic of these things is that they are important to the people who created them. Norman (1991) considers the cognition as a set, in the mind (internal) and in the world (external). Internal things are the knowledge, concepts, and structure in the forms of mental images, schemas, or connections. External things are external rules, constraints or relationship embedded in physical configurations.

Combining all these concepts of representation, I propose a simplified definition in this thesis.

A representation is a sign that describe the "things" that can be both in humans' mind and in the world. The representation can be perceived through different senses, including visual, auditory, olfactory, gustatory and tactile. The relationship between the sign and the thing that it represents can be symbolic, metaphorical and iconic. (Fig. 2.6)

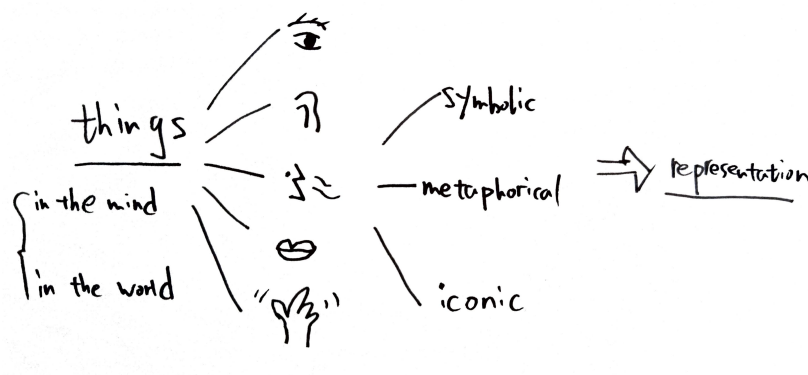


Figure 2.6. A simplified definition of representation

This thesis focuses on describing things in people's mind through visual signs, thus visual representations. With this definition of representation, design becomes a process to transcribe the *things* into a sign-system and arrange them in such a way as best to accomplish a particular purpose.

Now, let us see some good examples of visual representations. In this

painting (Fig. 2.7) from Denis Diderot's encyclopedia, the important elements are extracted and selectively represented from an ordinary scene in a factory. The idea of selecting important things to represent might seem trivial to us. But I believe this idea is important in designing representation because the first question is to choose which things to represent.

Tufte (2001)'s book *The Visual Display of Quantitative Information* presents a collection of hundreds of visual representations. These beautiful representations exploit the characteristics of perceptual properties of the visual system and show the underlying data (or *things*) effectively. Categories of representations introduced in the book includes data maps, time-series, space-time narrative designs and relational graphics. McCloud (1993)'s book *Understanding Comics* have great examples of representation of stories and events. It illustrates how comics, as a form of sequential visual art, represent invisible things such as time, motion and feelings.

Manipulation

While representation is about how to represent the *things* that people are interested in, manipulation concerns what the people can do with the representation. Bill Verplank introduces interaction design as *representation for manipulation* in his sketchbook ².

What computers do is to represent other things both real and imaginary. The form of representation is not arbitrary. The best representations are compact and extensible, efficient and widely available. The goal for representations is usually some form of manipulation or translation.

In his sketch, he gives the example of controls such as handles and knobs for manipulation. In today's graphical user interfaces, people do not use controls but perform their action directly on the objects (precisely, the representation of objects of interest). This is known as *direct manipulation*. The interaction model of direct manipulation proposed by Ben Shneiderman (1983) includes several principles: 1) continuous representation of objects of interest, 2) physical actions on objects rather than complex syntax, 3) fast, incremental, and reversible operations with an immediately-apparent effect, and 4) the layered or spiral approach to learning. Hutchins et al. (1985) give a cognitive account of both the advantages and disadvantages of direct manipulation interfaces, identifying underlying phenomena that give rise to the feeling of directness. They argue that direct engagement is the feeling that "*one is directly engaged with control of the objects - not with the programs, not with the computer, but with the semantic objects of our goals and*

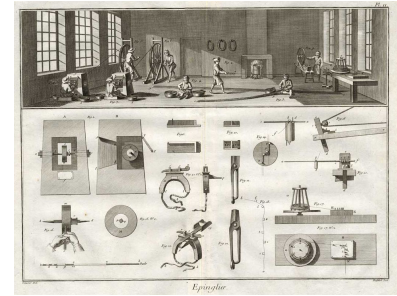


Figure 2.7: Example of Denis Diderot's encyclopédie. Source: https://commons.wikimedia.org/wiki/File:Defehrt_epinglier_pl2.jpg

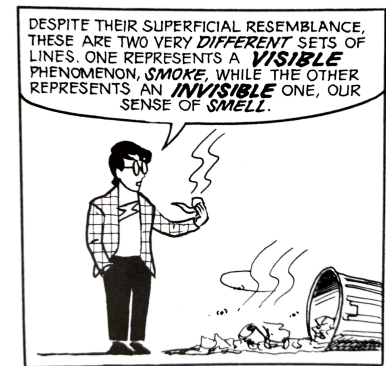


Figure 2.8: Scott McCloud shows how comics represent invisible things such as smell. Image taken from (McCloud, 1993)

² Source: <http://www.billverplank.com/IxDsketchBook.pdf>



Figure 2.9: Bill Verplank's Sketch interaction design. Source: <http://billverplank.com/CiiD/IDSketch.pdf>

intentions". In order to have direct engagement, the representation of objects should behave as if they were the real thing. Beaudouin-Lafon (2000) extends and generalizes the principles of direct manipulation to instrumental interaction: interaction between users and domain objects is mediated by interaction instruments, similar to the tools and instruments we use in the real world to interact with physical objects.

The manipulations I have designed in this thesis focus on the interactive behaviors of the new representations. The purpose is to give users a sense of directness and engagement.

2.2 Personal Information Management

Personal Information Management (PIM) describes the collection, storage, organization and retrieval of information by an individual computer user. It draws from several disciplines such as digital libraries, database management, information retrieval and human-computer interaction (HCI). The following sections describe three focuses of PIM research: context in PIM, paperless office and today's diverse user groups and types of documents. While this section provides a historical account, more recent related work is in each chapter.

Context in PIM

Suchman (1987) criticized the cognitivist view that both human mind and computers are information processors manipulating representations of the world. Her theoretical view, *situated action*, emphasizes the interrelationship between people's actions and their context: "*underscores the view that every course of action depends in essential ways upon its material and social circumstances. Rather than attempting to abstract action away from its circumstances and represent it as a rational plan, the approach is to study how people use their circumstances to achieve intelligent action.*" A live example is the video made in 1983 where PARC computer scientist Austin Henderson and Lucy Suchman, Ph.D. student in Anthropology at UC Berkeley, observed two computer scientists, Ron Kaplan and Allen Newell, using a copier. This situated approach motivated early research on PIM to study practices in context, such as their workplace.

In the 1980s, there were several early personal information systems, including the Xerox Star (Smith et al., 1982). Malone (1983) pointed out that "*None of these systems, however, is based on a systematic under-*



Figure 2.10: Screenshot from the video where Ron Kaplan and Allen Newell tried to use the copier. Source: <https://www.youtube.com/watch?v=DUwXN01ARYg>

standing of how people actually use their desks and how they organize their personal information environments.”. His study identified two key user strategies for organizing their desk: filing and piling. He also noted that categorizing information is cognitively difficult and that informal piles on the desk allow people to avoid the cognitive effort required for long-term filing.

Barreau and Nardi (1995) investigated a similar question for computer files. They identified three types of information: ephemeral, working and archived. They observed that ephemeral and working items were mainly retrieved by browsing whereas archived items were searched for. Teevan et al. (2004) also observed users preference towards “*orienteering*”, i.e. navigating one step at a time rather than jumping to a search results, because orienteering lets users specify less information at once. Many researchers have focused on other types of personal information such as email (Ducheneaut and Bellotti, 2001; Mackay, 1988; Whittaker and Sidner, 1996), web bookmarks (Abrams et al., 1998) and images (Rodden and Wood, 2003).

Paperless Office

Although these early studies have implied a shift from paper-based to digital-based knowledge work, paper is still around. The term “*paperless office*” can be tracked back to Xerox PARC, the birthplace of many radical ideas including laser printer, graphical user interface and Ethernet. Interestingly, “*Paperlessness as a goal ran completely counter to what was then Xerox’s main business: the making of money from paper, in particular the copying of one paper document onto another*” (Sellen and Harper, 2003). In their book “*The Myth of the Paperless Office*” (2003), Abigail Sellen and Richard Harper pointed out the unique *affordance* of paper such as supporting annotation while reading, quick navigation, flexibility of spatial layout, and its value in a wide variety of activities in office life. Both a real-world reading study (Adler et al., 1998) and a controlled lab study (O’Hara and Sellen, 1997) have shown people’s diverse reading practices and how paper can support these activities. Based on these insights, they advocate viewing paper as an analytical resource for the design of technologies rather than as a problem (Sellen and Harper, 1997). Other ethnographic studies with air traffic controllers (Mackay, 1999), field biologists (Yeh et al., 2006) (Fig. 2.12) and bench biologists (Tabard et al., 2008) (Fig. 2.11) have also shown the importance of physical paper in these users work practices.

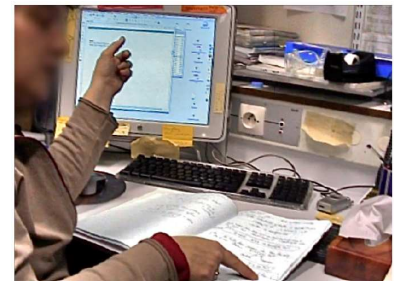


Figure 2.11: Biologist’s paper lab notebook. Source: Tabard et al. (2008)

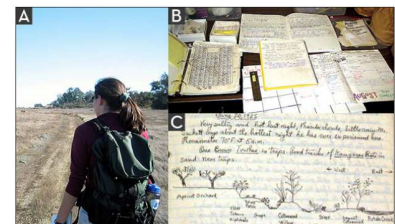


Figure 2.12: Field biologist’s paper lab notebook. Source: Yeh et al. (2006)

Diverse Digital Information

Today's digital information is becoming increasingly diverse, raising new challenges for users to manage. New forms of documents emerge such as computational notebook and interactive documents. Computational notebooks (Fig. 2.13) combine code, visualizations, and text in a single document, supporting the workflow of data analysis, from interactive exploration to publishing a detailed record of computation (Kluyver et al., 2016). Researchers, data analysts and even journalists are rapidly adapting this new form of notebook. In recent years, HCI researchers have started to investigate the information management challenges of this new type of documents (Kery et al., 2018; Rule et al., 2018b).

The Explorable Explanations³ introduced by Bret Victor have opened up a new way of thinking about documents.

The goal is to change people's relationship with text. People currently think of text as information to be consumed. I want text to be used as an environment to think in.

His project has inspired and motivated other HCI researchers to re-think and explore the dynamic nature of digital documents (Conlen and Heer, 2018; Dragicevic et al., 2019). For example, Dragicevic et al. (2019) apply the idea to multiverse analysis documents, allowing readers to explore alternative analysis options by dynamically changing some elements of the document.

Users also interact with increasing numbers of devices and applications for their work. Jung et al. (2008) introduced the concept of *ecology of artifacts* to describe any implicit or explicit relationships among interactive artifacts in one's personal life, encouraging designers to consider the dynamic interplays among multiple related artifacts. Other empirical studies have identified the challenges of managing these devices and applications. Dearman and Pierce (2008) found that managing information across devices was the most challenging aspect of using multiple devices. Oleksik et al. (2012) studied the artifact ecology of a research center and found that scientists used multiple computing applications to create information artifacts that are locked into applications, making it difficult to reuse content and get a unified view of the related research material.

This thesis thus investigates how today's knowledge workers interact with information in the modern age where new forms of documents and an increasing number of applications are the norm.

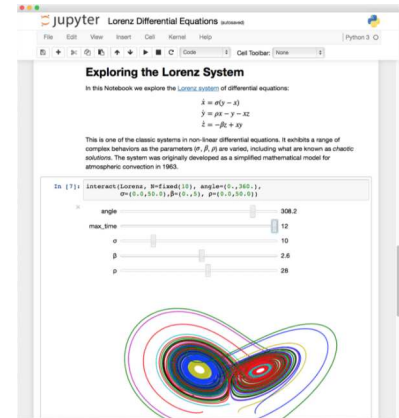


Figure 2.13: Jupyter Notebook.
Source: <https://jupyter.org/>

³ <http://worrydream.com/ExplorableExplanations/>

2.3 Document Software Systems

If we look back in history, several visions and demos have fundamentally changed the way people interact with information. I describe influential document systems in chronological order, followed by more recent systems that provide alternative perspectives.

Influential Visions and Demos

Memex (1945). Bush (1945)'s essay "*As We May Think*" envisioned a new way of accessing information through association. Bush described a system called "*Memex*" (Fig. 2.14) based on linked microfilm records:

A library of a million volumes could be compressed into one end of a desk. If the human race has produced since the invention of movable types a total record, in the form of magazines, newspapers, books, tracts, advertising blurbs...

Even though he did not build it, his idea of cross-linked information inspired HypterText and the World Wide Web, fundamentally changing the way people access information.

NLS (1968). In 1968, Doug Engelbart gave the demo of his NLS system⁴, which is also known the "*mother of all demos*". In this demo, he showed the graphical interfaces, hypertext, and computer supported cooperative work, which are all widely used today. Let us take a closer look at one of this demo (Fig. 2.15). At one point, he calls in the remote collaborator, Bill Paxton, and starts to work on the shared document together. If we look closely, we can actually see two separate cursors. These two cursors even has different capabilities as Doug Engelbart explained: "*but we have carefully reserved for me the right to control and operate on this so that my bug (cursor) is more powerful than yours.*" The intent of this design is to enable people work on the same shared document together.

Xanadu (1967). Deeply inspired by Bush's vision, Ted Nelson introduced the idea of *hypertext*, interconnected information such as text, graphics and sounds. In one demo of his project Xanadu (Fig. 2.16), Ted Nelson shows both the concepts of transclusion and link. The transclusion content is placed side by side and as he navigates the main document, the companion document is taken into the front, ready to be read. The intent of this design is to clearly show the connections among documents.

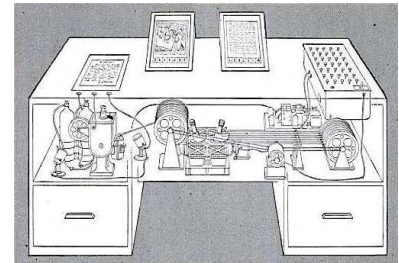


Figure 2.14: Bush's Memex

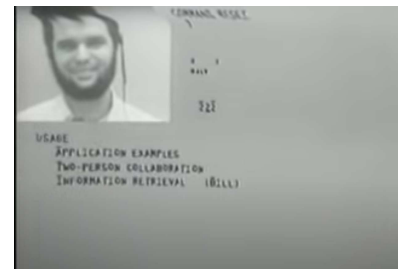


Figure 2.15: Engelbart's NLS. (See demo: <https://www.youtube.com/watch?v=qI8r8D46J0Y&list=PL76DBC8D6718B8FD3&index=9>)

⁴See <https://dougengelbart.org/content/view/155/>

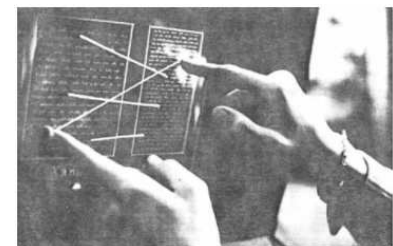


Figure 2.16: Nelson's Xanadu. See demo: https://www.youtube.com/watch?v=En_2T7KH6RA

The idea of linking information was taken further by other systems such as NoteCard (Halasz et al., 1986), HyperCard ⁵, Intermedia and so on. For example, in this demo of Intermedia ⁶, we can see that users can create bi-directional link between specific parts of any document as easily as copy-paste and visualize all the links in a web.

Xerox Star (1981). The Xerox Star is the first commercial system that makes use of many concepts of user interfaces such as direct manipulation, the mouse as input device, overlapping windows and a WYSIWYG text editor. Applying a desktop metaphor, documents are represented by "*concrete objects*" that can be selected, moved, filed, copied, mailed and opened by other applications, etc. (Smith et al., 1982).

Bravo and Gypsy (1974). One killer app of the Xerox Star is Bravo - the first WYSIWYG text editor. However, Bravo is still a modal editor where characters typed on the keyboard can be interpreted as either content or commands, depending on the mode. Larry Tesler realized the issue with modes and introduced the notion of a *modeless* interface. Copy-paste (Fig. 2.17) is one of the most known modeless pattern. One important realization towards the vision of modeless interface is the shift of the syntax of the command from prefix to suffix (Tesler, 2012), which makes selection the first step of most user actions. Larry Tesler and his colleagues developed Gypsy, a document preparation system that is based on Bravo but eliminated modes. Gypsy was taken further to BravoX, LisaWrite and MS Word.

Alternative Representations

More recent HCI research continues to innovate alternative perspectives and ideas.

Metadata-Based Systems. To address the issues of using a hierarchical structure for organizing information, Dourish et al. (1999)'s Presto system (Fig. 2.18) provides user-level document attributes, allowing users to rapidly reorganize their documents for the task at hand. These document attributes such as year, author and topic, are treated as first-class objects and used to search, group and organize documents. Presto is an example of a metadata-based system that makes use of the meta-data of the documents to provide appropriate representations and interactions. Harper et al. (2013) reflected on the representation of digital files and also proposed to encompass meta-data within a file abstraction.

Zoomable Systems. Bederson and Hollan (1994) challenge the metaphor-

⁵ HyperCard demo: <https://www.youtube.com/watch?v=FquNpWdf9vg>

⁶ Intermedia demo: <https://www.youtube.com/watch?v=bGGxdF0Pn4g>

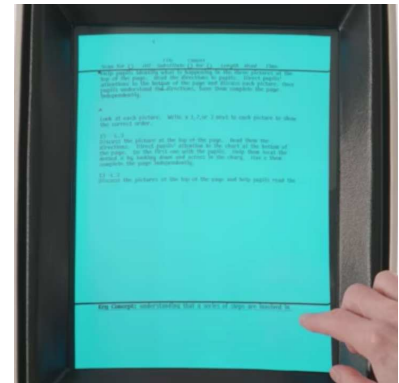


Figure 2.17: Cut-paste in Gypsy. See demo: <https://www.youtube.com/watch?v=Dhmz68CII9Y>

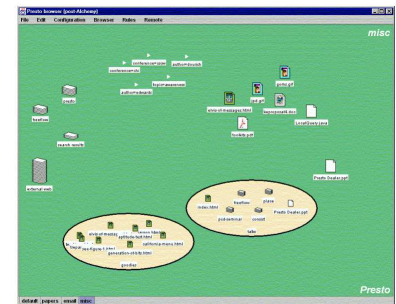


Figure 2.18: Dourish et al.'s Presto

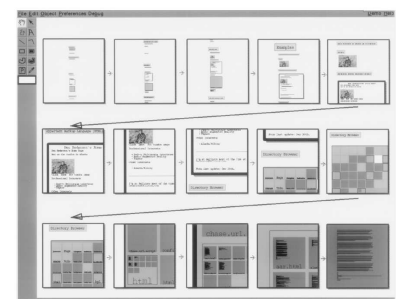


Figure 2.19: Bederson and Hollan's Pad++

based approach, which primarily mimics mechanisms of older media. Their system, Pad++, is a zooming graphical interface where data objects of any size can be created, and zooming is a fundamental interaction technique (Fig. 2.19). They used *semantic zooming* based on Pad (Perlin and Fox, 1993), that changes representations when zoomed:

It is natural to see the details of an object when zoomed in and viewing it up close. When zoomed out, however, instead of simply seeing a scaled down version of the object, it is potentially more effective to see a different representation of it.

The intent of this design is to see interface design as the development of a physics of appearance and behavior for collections of informational objects.

3D and Time-Based Systems. While most desktop interface are 2D based, an alternative approach is to add another dimension and make it 3D. For example, Robertson et al. (1998)'s Data Mountain (Fig. 2.20) allows users to place documents at arbitrary positions on an inclined plane in a 3D desktop virtual environment using a simple 2D interaction technique. Their intent is to take advantage of human spatial memory, i.e., the ability to remember where you put something. Another dimension that can be useful is time. LifeStreams (Fertig et al., 1996) (Fig. 2.21) leverage the temporal dimension of data to organize documents.

Activity-Based System. Bardram et al. (2006) propose the activity-based computing (ABC) framework to let users manage activities on their desktop. The ABC framework challenges the application-centric and document-centric approach to designing desktop interfaces, treating *activity* as a first-class object. The user can create and manage activities under their control, e.g. group windows and resources into activities that can be resumed or suspended to switch among tasks. The concept of focusing on activities has inspired other systems such as Giornata (Volda and Mynatt, 2009) and TAGtivity (Oleksik et al., 2009) (Fig. 2.22).

In summary, these systems explored a variety of design spaces to interact with documents and served as great inspiration for the design of new representations in this thesis.



Figure 2.20: Robertson et al.'s Data Mountain

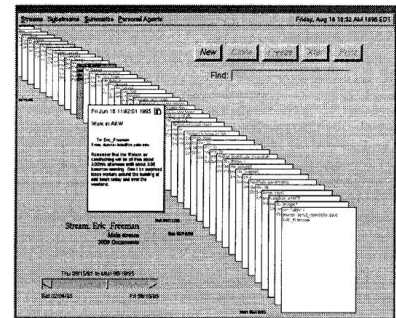


Figure 2.21: Fertig et al.'s LifeStreams

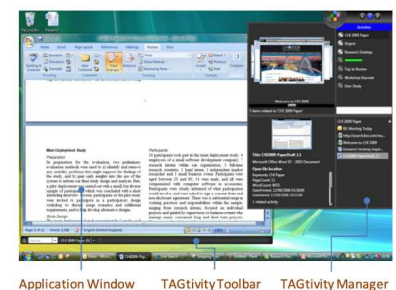


Figure 2.22: Oleksik et al.'s TAGtivity

2.4 Interaction Frameworks and Models

Besides developing novel document interfaces, research has also proposed interaction models to guide the design of alternative user interfaces.

Reality-Based Interaction

Despite of the dominance of direct manipulation interfaces, HCI researchers have not stopped inventing the next generation of UIs. They have developed a broad range of interaction styles beyond the WIMP interaction styles including tangible interaction, ubiquitous computing, virtual and augmented reality, etc. These diverse interaction styles have a common theme that:

"All of these new interaction styles draw strength by building on users' pre-existing knowledge of the everyday, non-digital world to a much greater extent than before." (Jacob et al., 2008)

The Reality-Based Interaction framework (Jacob et al., 2008) proposes four themes from the real world that new interaction styles use: naïve physics, awareness of one's physical body and skills, awareness and skills within the surrounding environment, and awareness and skills in a social context. On the other hand, the framework also recog-

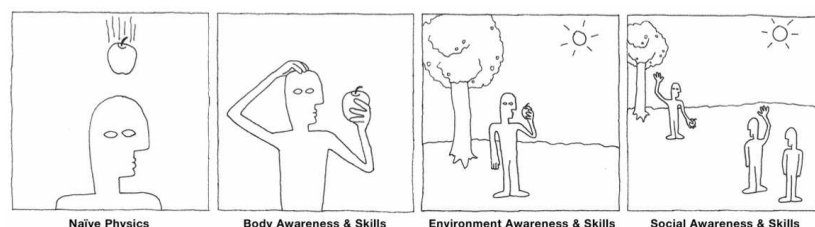


Figure 2.23. Jacob et al.'s Reality-Based Interaction

nizes that purely mimicking reality does not take full advantage of the computation offered by the digital world and *"a useful interface will rarely entirely mimic the real world, but will necessarily include some unrealistic or artificial features and commands."* Interface designers need to make tradeoffs among qualities such as expressive power and efficiency, versatility, ergonomics, accessibility and practicality. The idea of designing interfaces and interactions beyond reality, or the old media, is also suggested by other HCI researchers such as Hollan and Stornetta (1992) and Klemmer et al. (2006).

Instrumental Interaction

GUIs are based on the principle of direct manipulation (Shneiderman, 1983) where users manipulate visual representations of the objects of interest through physical actions. However, current GUIs rely on the manipulation of additional widgets such as menus, scrollbars and dialog boxes, causing indirect interaction with objects of interest (Beaudouin-Lafon, 2000; van Dam, 1997). Instrumental Interaction (Beaudouin-Lafon, 2000) is based on the observation that our interactions in the physical world are also often mediated by instruments or tools, e.g. pencils, screw-drivers and toothbrushes.

In instrumental interaction, the objects of interests, such as documents, are called domain objects, and are manipulated through interaction instruments, or UI elements. The interaction instruments transform the user's actions into commands that act upon the objects. The instru-

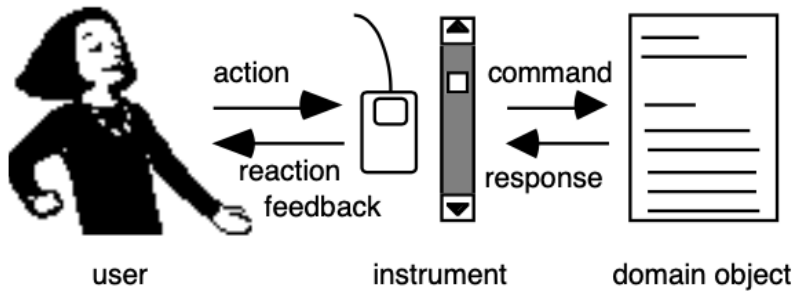


Figure 2.24. Beaudouin-Lafon's Instrumental Interaction

mental interaction also provides a model to compare different interaction styles with three aspects including 1) degree of indirection (spatial and temporal), 2) degree of integration and 3) degree of compatibility.

These interaction models provide the theoretical foundation of this thesis. My goal is to apply these interaction models to design new representations that address the challenges faced by today's knowledge workers.

3

Representation for Document Editing

The work reported in this chapter was joint with Miguel Renom, a Ph.D student in the same research team. I took the lead on the project. Miguel Renom and myself conducted the interview study and created the initial design concept together with Michel Beaudouin-Lafon. Michel Beaudouin-Lafon implemented the first functional prototype and Miguel implemented the more advanced one used in the evaluation study. I designed and conducted the evaluation study. The whole team contributed to the paper writing. The work was published at ACM CHI2020 (Han et al., 2020).

This chapter explores one of the most common document tasks: editing. We focus on one example of extreme user, legal professionals, to study their current practices with existing text editing tools. Based on this empirical understanding, we explore the idea of reifying the transient text selection into persistent and interactive objects called *Textlets*. Users can manipulate this new representation, user interface object, to perform various editing tasks, including selective search and replace, word count, and alternative wording. The evaluation study shows the validity and generative power of the concept of textlets.

3.1 Context

Text editing was once considered a ‘killer app’ of personal computing (Bergin, 2006). Editing text is usually the first skill a novice computer user masters, and all personal computers are sold with a word processor. Many professions require advanced text editing skills to



Figure 3.1: Editing a document

ensure consistent use of terms and expressions within structured documents, such as contracts, patents, technical manuals and research articles (Fig. 3.2).

For example, lawyers begin each contract with a list of defined terms, and must use them consistently thereafter. This is critical, since ‘minor’ wording changes can have serious legal implications. For example, American patents define “comprises” as “consists at least of”; whereas “consists of” means “consists only of”, indicating a significantly different scope of protection. Sometimes terms are disallowed, e.g. the US Patent and Trademark Office (USPTO) does not accept “new”, “improved” or “improvement of” at the beginning of a patent title. Word limits are also common, such as the European Patent Office’s 150-word limit for patent abstracts.

Despite their many features, standard word processors do not address all of these professional needs. For example, although spell checking is common, flagging forbidden words or ensuring consistent use of particular terms must be done manually. Real-time counts of words and characters can be displayed for the whole document, but not for a single section. Thus, we decided to study how legal professionals manage the contrarians and consistency when editing technical documents.

3.2 Related Work

Literature on both word processing and code editing are relevant to us. Code is a particularly interesting form of technical document that requires professional software developers to manage multiple internal constraints, and the specific tools developed to ensure internal consistency in code may inform our design.

Text Editing Practices

Text editing was an active research topic in the 1980s when word processors became mainstream. For example, Card et al. (1987) modeled expert users’ behavior in manuscript-editing tasks; Tyler et al. (1982) investigated the acquisition of text editing skills; and Rosson (1983) explored the effects of experience on real-world editing behavior. Others examined paper-based editing practices to improve computer-based text editing (Marshall, 1997; O’Hara and Sellen, 1997; Sellen and Harper, 1997) and collaborative writing (Baecker et al., 1993; Churchill et al., 2000; Noël and Robert, 2004).

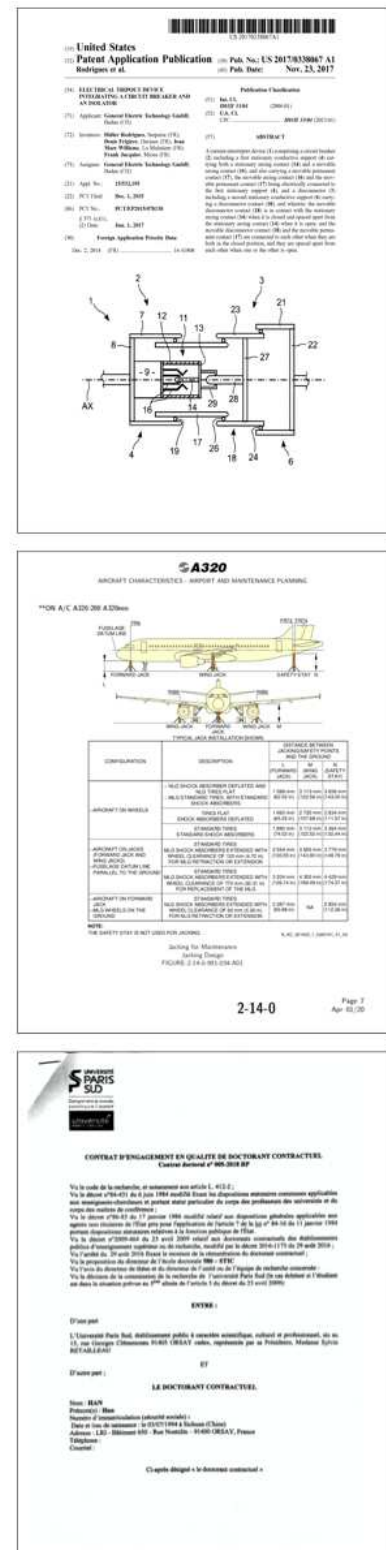


Figure 3.2: Example of technical documents: patent (top), technical manual (middle), contract (bottom)

More recent studies identify issues with modern word processors. For example, Srgaard and Sandahl (1997) found that users rarely take advantage of text styles, and argue that this is because styles do not impose restrictions on the document structure. Alexander et al. (2009) found that although users often revisit document locations, they rarely use the specific revisitation tools found in Microsoft Word and Adobe Reader. Chapuis and Roussel (2007) examined users' frustration with unexpected copy-paste results due to format conversion. This work identifies a clear mismatch between the advanced features offered by modern word processors and actual user practice, and highlights the need for new tools and concepts.

Tools to Support Text Editing

Researchers have created a variety of text editing tools to support annotation (Schilit et al., 1998; Yoon et al., 2013; Zheng et al., 2006), navigation (Alexander et al., 2009; Laakso et al., 2000; Wexelblat and Maes, 1999) and formatting (Myers, 1991); as well as distributing editing tasks (Bernstein et al., 2015; Teevan et al., 2016) and taking advantage of a text's structure (Miller and Myers, 2002a). We focus here on copy-paste (Bier et al., 2006; Stylos et al., 2004), and search-and-replace (Beaudouin-Lafon, 2000; Miller and Marshall, 2004), both especially relevant to supporting internal document consistency.

Chapuis and Roussel (2007) propose new window management techniques to facilitate copy-paste tasks. Citrine (Stylos et al., 2004) extracts structure from text, e.g. an address with different components, that can be pasted with a single operation. Multiple selection (Miller and Myers, 2002b) offers smart copy-paste that is sensitive to source and destination selections, while Entity Quick Click (Bier et al., 2006) extracts information to reduce cursor travel and number of clicks. Cluster-based search-and-replace (Miller and Marshall, 2004) groups occurrences by similarity, allowing entire clusters to be replaced at once (Fig. 3.3). Beaudouin-Lafon (2000) instrumental search-and-replace tool highlights all items at once, so users can make changes in any order, not only as they occur in the document (Fig. 3.4).

Commercial applications such as *Grammarly*¹ check grammar and spelling by suggesting alternative wording, style and tone, among other features. However they do not ensure consistent use of specific terms, e.g. always referring to a party in a contract with a single name. Other software tools automatically generate consistent references, including *Mendeley*² and *EndNote*³ for researchers, and *Exhibit Manager*⁴ for legal professionals. Although automated reference management solves

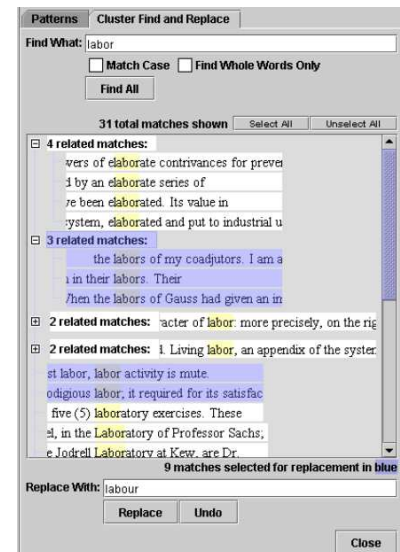


Figure 3.3: Miller and Marshall's cluster-based search and replace

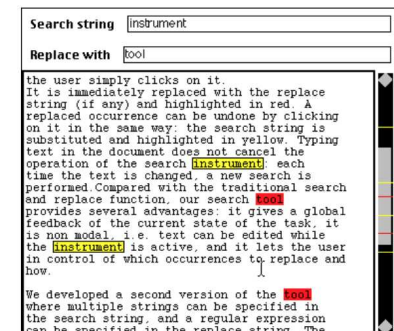


Figure 3.4: Beaudouin-Lafon's instrumental search and replace

¹ <https://grammarly.com>

² <https://mendeley.com>

³ <https://endnote.com>

⁴ <https://exhibitmanager.com>

some problems, users still lack flexibility for others, e.g. creating a custom citation format. These tools are also separate from the word processor, potentially distracting users from their documents and fragmenting workflow.

Code Editing Practices

Code editing has been widely studied, especially copy-paste (Kapsner and Godfrey, 2008; Kim et al., 2004), use of online resources (Brandt et al., 2009) and drawings (Cherubini et al., 2007), and performing maintenance tasks (Ko et al., 2005). A key challenge that emerges from these studies is how to manage dependencies. For example, Kim et al. (2004) found that programmers rely on their memory of copy-pasted dependencies when they apply changes to duplicated code. Ko et al. (2005) identified both ‘direct’ dependencies, e.g. going from a variable’s use to its declaration, and ‘indirect’ ones, e.g. going from a variable’s use to the method that computed its most recent value, and proposed ways of visualizing these dependencies in the editor. While technical document constraints are less stringent than in computer code, we hope to exploit certain commonalities.

We see program code as an extreme case of a technical document, with many internal constraints. For example, Toomim et al. (2004)’s technique supports editing duplicated code and visualizing links among duplicates. To help programmers use web examples more efficiently, *Codelets* (Oney and Brandt, 2012) treat snippets of code examples as ‘first-class’ objects in the editor, even after they are pasted into the code. Kery et al. (2017)’s tool for lightweight local versioning supports programmers performing exploratory tasks, while *AZURITE* (Yoon and Myers, 2015) lets programmers selectively undo fine-grained code changes made in the editor. *Barista* (Ko and Myers, 2006) supports enriched representations of program code, while *Whyline* (Ko and Myers, 2004) and *HelpMeOut* (Hartmann et al., 2010) support debugging tasks.

3.3 Interview with Legal Professionals

Editing technical documents requires a complex editing process (Cohen et al., 1999), especially to maintain the document’s constraints and internal consistency (Farkas, 1985). We first conducted critical object interviews (Mackay, 2002) to better understand how professionals manage such constraints and consistency in their technical documents.

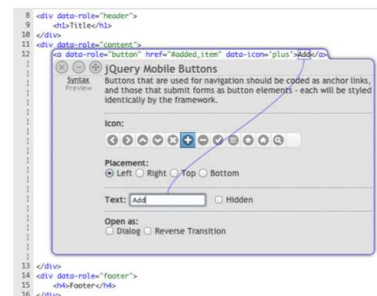


Figure 3.5: Oney and Brandt’s Codelets

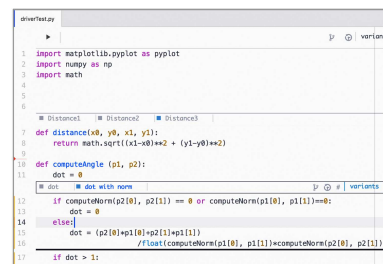


Figure 3.6: Kery et al.’s Variolite

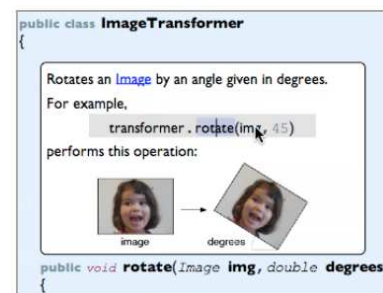


Figure 3.7: Ko and Myers’s Barista

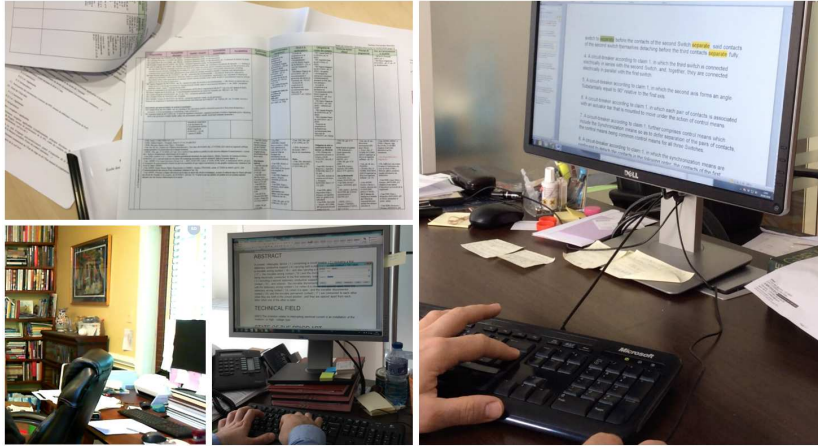


Figure 3.8. Critical object interviews in participants' workplace.

Participants: We interviewed 12 participants (three women, nine men; aged 24-50). Their occupations include: contract manager, legal affairs director, candidate to a Ph.D. in law, lawyer, patent attorney, and patent engineer. All use Microsoft Word on either the Windows (11/12) or MacOS (1/12) platforms; only one uses the latest 2019 version.

Procedure: All interviews were conducted in English, each lasting from 45-60 minutes. We ran four pilot interviews with colleagues to establish the protocol, then visited participants in their offices and asked them to show us specific examples of their current digital and physical documents. We asked them to describe a recent, memorable event related to editing that document, either positive or negative. The first two authors conducted all interviews, alternating asking questions.

Data Collection: All interviews were audio recorded and transcribed. We also took hand-written notes. We were not allowed to videotape or take photographs for confidentiality reasons.

Data Analysis: We analyzed the interviews using reflexive thematic analysis (Braun and Clarke, 2019). We generated codes and themes both inductively (bottom-up) and deductively (top-down), looking for breakdowns, workarounds and user innovations. After interviewing eight participants, we conducted the first analysis together, grouping codes into larger categories and focusing on participants' editing behavior. We discussed any disagreements and rechecked the interview transcripts to reach a shared understanding. We also created story portraits (Jalal et al., 2015) to graphically code the data, which helped us engage with the collected data and resolve disagreements, e.g. Fig. 3.9.

We arrived at the final themes after three iterations.

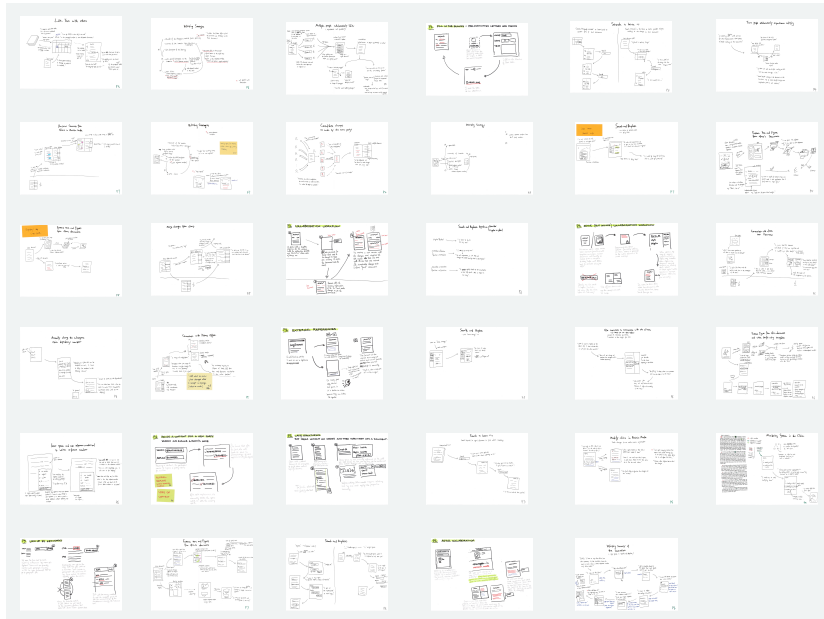


Figure 3.9. A collection of story portraits based on the analysis

3.4 Results and Discussion

We identified six themes: maintaining term consistency, managing dependencies by hand, reusing content, visiting and revisiting document locations, managing annotations, and collaboration.

Maintain Term Consistency

All participants rely on their memories to maintain consistency across document terms, which are often defined in the beginning of the document. This causes problems. For example, P7 (legal affair director) struggled to use the full name of a party across the document and P5 (patent attorney) often made the wrong choice between two words with highly similar meanings.

Sometimes terms must be changed, e.g. shifting from British to American English or if the client prefers another word. To avoid introducing inconsistencies, lawyers must update each term and its variations, e.g. singular vs. plural, and adjust verbs (P1), articles (P1,6,7,9) and pronouns (P9) accordingly.

Although all participants use "search and replace" to make consistent



Figure 3.10: Which term to use?

edits, most (9/12) avoid “replace all”: “It is too risky.” (P4); “I will not let the computer do it for me.” (P6); and “I prefer to do it manually.” (P5). Instead, they use a one-by-one search-navigate-check-replace strategy to manually replace each term. They ensure correctness by viewing and assessing each term’s context: “We have to conjugate the verb with the subject. It’s like a lot [of work].” (P4) Checking context is also essential for avoiding partial matches, i.e. when the search term matches a subset of a longer word (P3, 11), which requires performing additional search-and-replace operations.

Summary: Participants maintain consistency across terms primarily by hand, which they find cumbersome and prone to error. Most avoid “replace all” because they do not trust the results and cannot easily check them.

Managing dependencies by hand

We define a dependency as two or more sections of text that must be kept identical or consistent. Most participants (8/12) rely on their memories to manage document dependencies, and synchronize them by hand. We identified three types of dependency problems: managing consistency across pasted copies, numbering items, and managing cross-references.

All patent attorneys (4/4) copy text from the *Claims* section to the *Summary of the Invention* when drafting the latter. However, when they change the claims, they often forget to update the summary accordingly: “Because it is not automatically updated with the claims, I can easily forget to update.” (P6).

Patents contain three types of numbering systems (Fig. 3.12): claim number, claim dependency number (when the current claim depends upon another claim), and reference number (to specific parts of the illustration). Most patent attorneys (3/4) manage these numbers by hand instead of using Word’s cross-reference feature, typically leaving a gap between consecutive references. This lets them add additional numbers later, while ensuring that the reference numbers remain in ascending order.

Most lawyers and patent writers insist on maintaining full control of the text, especially the critically important claims section, even if the process is tedious. This is especially true of the claims section, which is critical to the patent. Even participants who are comfortable using automatic features do not rely on automatic numbering:

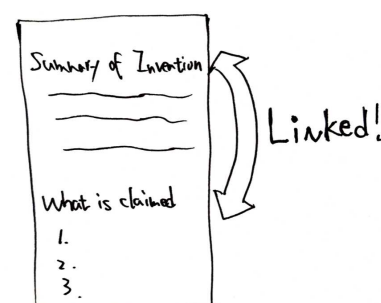


Figure 3.11: Summary of Invention and claims are linked

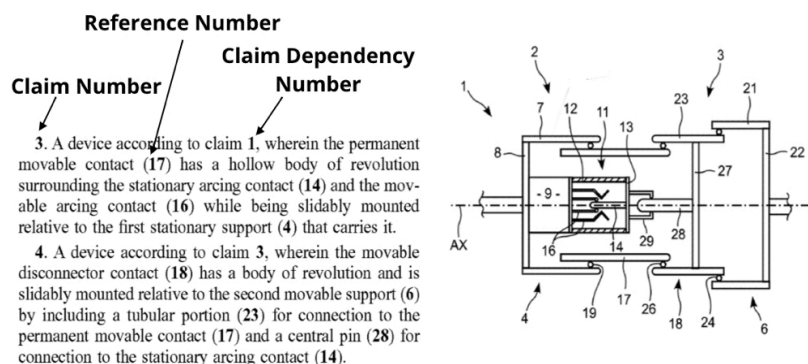


Figure 3.12. Patent claims use three different numbering systems (left). Patent illustration (right).

P7 said: “most of the time, I prefer if something can be automatically achieved” yet avoids automatic numbering: “I cannot really tell you why. One reason might be that if I have automatic numbering set up, this would have become paragraph 2 and all the numbering of the claims would have been changed...I would not be very happy.”.

Summary: Their key reasons for avoiding automatic numbering include 1) their inability to differentiate automatic from normal numbering, unless they select the text; 2) incorrect display of references, e.g. when items are added to a list, until a manual update is triggered; and 3) invisibility of dependencies after an update, since they lack feedback and cannot be sure if the changes are correct.

Reusing content

All participants reuse previous document elements to create new documents, incorporating text, styles and templates. When copy-pasting a piece of text for reuse, they must often edit the content between copy and paste operations or adapt the format after pasting, e.g. using the brush tool (P6) or a macro (P7). If visual formatting results from applying a style, pasting new text can bring “bad” styles into the document and pollute existing styles: “When you copy-paste into a document, you can import the style of the [original] document. Too many unnecessary styles makes the document heavier and you have to remember which style to use. This is a mess.” (P4). Although “paste without formatting” option is available right after pasting the text in the destination document, the option disappears after users start other operation, e.g. clicking the text to start editing. Since the default pasting behavior is to keep the source formatting, users also need two additional clicks to apply “paste without formatting”.

Most participants (10/12) use templates to create new documents, in-

	Unselected	Selected
Normal numbering	1. apple 2. orange 3. pear	1. apple 2. orange 3. pear
Automatic numbering	1. apple 2. orange 3. pear	1. apple 2. orange 3. pear

Figure 3.13: Normal and automatic numbering. The user cannot tell them apart unless she selects them

cluding pre-written text, preset styles or both. Although useful for writing letters, filling cover pages, generating tables and managing formatting consistency, participants still struggle with formatting issues caused by style conflicts.

Summary: They often reuse content, but are not satisfied with the corresponding introduction of format inconsistencies.

Visiting and revisiting document locations

Participants rarely write or edit documents sequentially and often revisit different parts of the document. For example, P7 created a set of keyboard shortcuts to “jump to different parts of the document” because he needs to switch often. This is consistent with Alexander et al.’s findings concerning users’ revisitation behavior (Alexander et al., 2009).

Participants also need better revisitation support when systematically going through the whole document, e.g. incorporating edits one by one or performing search-and-replace tasks. The latter often involves checking an earlier replacement, after the fact. Unfortunately, Word imposes sequential interaction, so users cannot return to the previous replacement: “The problem is that I cannot check. It made the replacement and it goes to the next occurrence, so I don’t see what just happened.” (P7). P8’s workaround to address this problem involves turning on “track changes” to leave an inspectable trace of each replacement.

Summary: Participants experience problems navigating their documents, especially with respect to tracking recent or oft-visited parts of the document.

Managing annotations

Some participants (4/12) appropriate and customize their tools to support comments and annotation, rather than using the dedicated features of their word processor. For example, P5 uses footnotes to add comments for his clients because he dislikes how the text gets smaller when using Word’s *Track Changes*. P7 avoids *Track Changes* altogether and uses different colors to encourage active reading and convey the importance of certain comments to his clients.

For documents with two or more co-authors, some participants (4/12) complained that the *Track Changes* feature introduces more problems than it solves (P2, P4) and makes it difficult to understand the modifications (P5, P7). Instead, some (3/12) use the comparison function

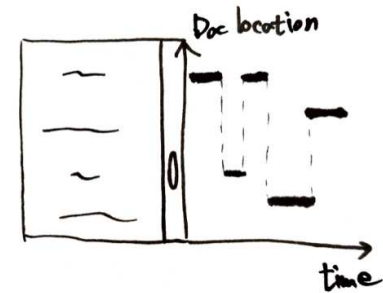


Figure 3.14: Participant jumps at different document locations.

after making changes, to make modifications visible to their clients.

Interestingly, P9 also used the comparison function to ‘cheat’: He modified the document with *Track Changes* on a Saturday night but did not want his client to know he worked over the week-end. So he accepted all the changes and then compared it to the original document on Monday morning, making it appear that the changes had been made on Monday.

Summary: Participants find annotation tools frustrating and constraining, and some creatively use other features to meet their needs.

Collaborate

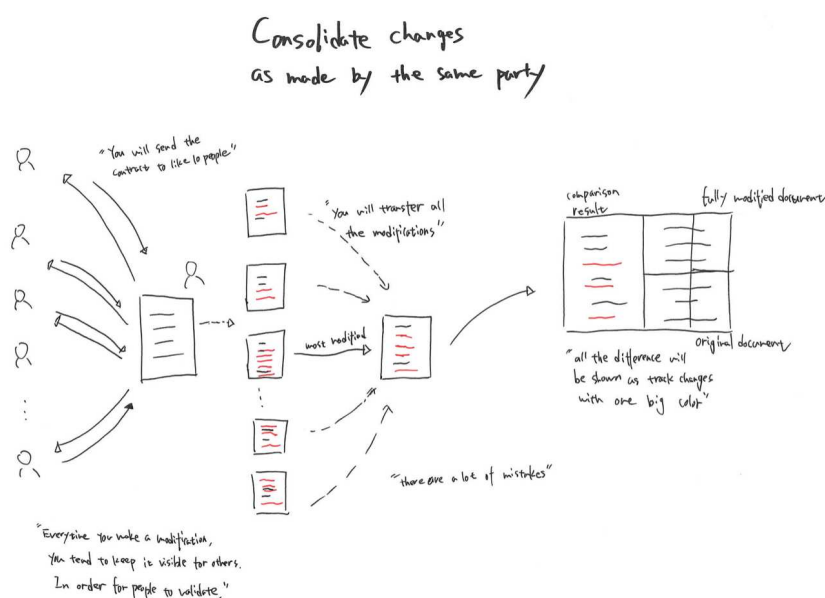


Figure 3.15. A story portrait illustrating how one participant consolidate changes from various collaborators as it from one party

Most participants (11/12) collaboratively edit documents. We categorize their collaboration strategy as branching (versioning and partitioning) and merging.

When versioning, participants exchange documents via email and save successive versions to keep track of changes made to the document. They use simple suffixes to identify versions over email, e.g. V1, V2, V3, so documents with similar content hang around and are hard to find again. P12 complained that she created eight versions of the same document even though she made only minor changes. The notion of File Biography (Lindley et al., 2018) could help them manage these issues. Local versioning, explored for code editing in Variolite (Kery et al., 2017), would also be useful but standard word processors do not

support it. Compare to edit history which is generated by the computer, versions are defined by the users to mark a substantial change to the document that are meaningful to them. In collaborative editing, people are afraid of “stepping on the foot of others” (Larsen-Ledet and Korsgaard, 2019). Version creates a safe and private space for individual work before committing to the collaborative space.

Some participants partition the master document for co-authors to edit and merge it later. The problem in the merge stage is style pollution, as discussed above, due to foreign styles being imported through copy-paste (P4) or forgetting to format text (P2). Because the style panel in Microsoft Word is not displayed by default when users open a document, it is often hidden from users. As a result, formatting and style inconsistencies are often undetected.

When a version of a document is sent out and then returns with proposed changes, participants have to merge these changes into the master document. Even though they use the *Track Changes* feature of Microsoft Word, they usually make the changes by hand, going through each document and deciding which edits to incorporate. They do not accept all the changes for various reasons: “*It might destroy the way [the text] was presented*” (P5), “*We do not consider all comments*” (P6), “*[clients’] comments are difficult to understand*” (P7), or the changes require other modifications to be made in other parts of the text (P7). In summary, we found that participants manually version their documents, even for minor edits, and merge documents by hand, incorporating changes one by one, as they struggle with style pollution.

Summary: The interview study shows not only that professional technical writers must maintain consistent use of terms, but also that they manage the resulting dependencies mostly by hand. They struggle to maintain formatting consistency when reusing text and lack tools for keeping track of their navigation within their document, flexibly generating annotations, and collaborating asynchronously. Based on these results and the theoretical framework provided by Instrumental Interaction (Beaudouin-Lafon, 2000), we propose a general solution to address some of their needs.

3.5 Textlets Concept

General-purpose word processors such as Microsoft Word have hundreds of features. As we saw in the interview study, even when users

know that a feature exists, such as ‘replace all’ or ‘automatic numbering’, they often prefer making changes by hand to stay in control. Rather than proposing specific new features to address the various use cases we observed, we seek a general approach that fits how they actually deal with text.

Word processors rely heavily on the concept of selection: the user selects a piece of text and then invokes a command using a menu, toolbar or keyboard shortcut that affects the content of the selection. However, the selection is transient: selecting a new piece of text causes the previous selection to be lost. (Fig. 3.16)

We introduce the concept of *textlet* as the *reification* (Beaudouin-Lafon and Mackay, 2000) of a text selection into a persistent, interactive, first-class object. A textlet represents a piece of the text document identified as interesting to the user. They can be highlighted in the document itself, listed in a side panel, or visualized through other interface elements, e.g. a scrollbar, for easy access.

To create a textlet, a user simply selects a piece of text and invokes a command, e.g. Ctrl+T. The selected text is highlighted and the textlet is listed in the side panel where a behavior (see below) can be assigned to it.

Textlets can also be created automatically by a higher-level object called a *grouplet*. For example, to create textlets that represent all the occurrences of a word in the document, the user creates a *search grouplet* (or *searchlet*), e.g. with the traditional Find command Ctrl+F. The *searchlet* appears in the side panel and the user can type the search string. A textlet is automatically created for each match of the search string and appears as an item underneath the *searchlet*. This list is automatically updated when editing the document or when changing the search string.

The power of textlets comes from the *behaviors* associated with them. The most basic behavior is to (re)select the piece of text from the textlet, e.g. by double-clicking the textlet representation in the side panel. Other behaviors include the ability to change or automatically generate the content of the text, to change its style, and to attach annotations or additional information, such as character or word count. Creating textlets with different behaviors leverages the power of polymorphism (Beaudouin-Lafon and Mackay, 2000) because a single concept (reified text selection) addresses a variety of commands (searching, counting, referencing), providing users with a unifying

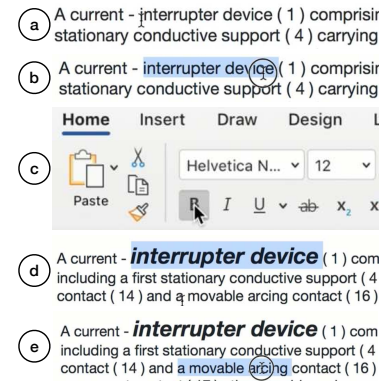


Figure 3.16: Transient selection. a,b) the user selects a piece of text; c) she applies a set of command to it; d) the appearance of the text has changed based on the command; e) as soon as she selects another text, the previous selection is gone.

concept to manage text documents. This slightly extends the definition in (Beaudouin-Lafon and Mackay, 2000), which focused on polymorphic instruments.

The rest of this section illustrates the power of textlets by describing how different behaviors can address some of the issues observed in the interview study. Table 3.1 summarizes the use cases and the solutions we have implemented.

Use Case	Issue	Behavior of textlet
Consistent Reuse	Recurrent copy-paste to start new documents from scratch requires re-selecting the text in one or more documents.	All textlets save their text, which can be reused using simple actions such as drag-and-drop.
Term Consistency	Repeatedly navigating across a document using search terms leaves no traces of scroll positions, making it hard to go back and forth.	<i>Searchlets</i> create <i>occurrence textlets</i> that let users navigate by interacting directly with them on the side panel.
Reference Consistency	Automated numbered lists and cross-references take control away from users. Numbered items and references do not update automatically.	<i>Numberlets</i> are counters that can be manipulated and applied to numbered lists, sections, figures, etc. References to <i>numberlets</i> can be created by copy-pasting them in the document. Item numbers and references are always up to date.
Length Constraints	Standard word processors require selecting text each time to count words in a specific area and get other metrics.	<i>Countlets</i> add a persistent decoration to the text of interest that displays a word count and updates it as users edit the content.
Exploratory Writing	Keeping track of alternatives is difficult. Undo/redo is not adapted to go back and forth between versions.	<i>Variantlets</i> store alternative versions of textlets that can be easily retrieved, compared and edited.

Table 3.1. How different behaviors of textlet address some issues and challenges observed in the interview

Textlets for Consistent Reuse

The interview study showed that technical writers often reuse portions of text or entire templates when creating new documents. They rely on copy-paste to incorporate parts of other documents, but this requires precisely (re)selecting the text to be copied.

With textlets, users can create text snippets specifically for reuse, such as common vocabulary and phrases, list templates, or pre-written paragraphs with placeholders. Reusing a snippet simply involves a drag-and-drop or click-based interaction with the textlet. Placeholders can themselves be textlets to highlight the parts that need to be filled in, so that they can be easily identified, selected, and replaced with the proper text.

These snippets can be collected in dedicated documents or embedded into other documents. The interview study identified collaborative practices where users share a set of constraints and consistency criteria. By collecting reusable textlets in separate documents, they can easily share these documents and facilitate consistency across users and documents.

Textlets for Term Consistency

We observed that technical writers need to go back and forth in their documents to check for consistency or make consistent changes across the document. To that end, they often use the search command, but they do not trust the search-and-replace tool enough to perform replace-all actions blindly, and prefer to check the term and its context before each replacement.

Searchlets, briefly introduced earlier, can address these use cases by automatically searching for all the occurrences of a text in the document. A *searchlet* is a *grouplet* that creates *occurrence textlets* for each match they find in the document. These occurrences are listed under the *searchlet* in a side panel and automatically updated when the document changes. This supports fast navigation to each occurrence in the document, e.g. with a click on the occurrence in the side panel.

Searchlets support flexible search-and-replace. After specifying a replacement text for the *searchlet*, the user can replace all occurrences at once, or replace them one by one, in any order. At any time, including after a replace-all, it is possible to revert individual occurrences, giving users full control and visibility over their actions. Multiple *searchlets* can be active simultaneously, so that users can keep earlier searches around and get back to them later.

When users navigate the document to check for consistency and to make changes, they often lose track of where they were when they started the check. *Searchlets* facilitate navigation among occurrences, but do not address the need for location tracking in the document.

Building on previous work such as Read Wear (Alexander et al., 2009; Hill et al., 1992) (Fig. 3.17) and Footprints (Wexelblat and Maes, 1999), a *history grouplet* can record recent selections and let the user navigate among them. Previous selections can appear as individual textlets in a side panel or, to save space, the *grouplet* can display arrows to navigate the history of selections.

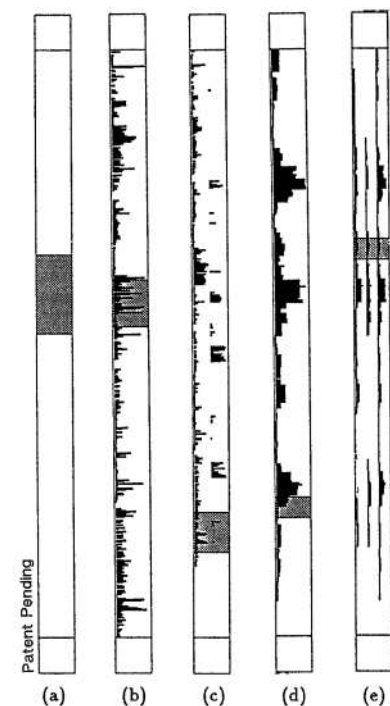


Figure 3.17: Hill et al. (1992)'s Edit and Read Wear

Textlets for Reference Consistency

Standard word processors include tools for managing certain types of dependencies automatically, most notably numbered lists and cross-references. The interview study showed that participants distrust and struggle with automatically numbered lists, and thus avoid automated cross-reference management tools.

Documents often include numbered items such as sections, figures, patent claims or references. Both the numbered items and the references are good candidates for textlets: Both are *computed textlets*, i.e. their content is computed and updated as the document changes, but the user can still interact with them. A *numberlet* is a *grouplet* that creates numbered items and ensures that the number sequence matches the document's item order. Each numbered item is itself a *grouplet* for creating and managing textlets representing references to that item.

Numberlets, numbered items and references can be listed in the side panel for easy navigation. Creating new numbered items and new references involves a simple drag-and-drop or clicking on the corresponding textlet.

This design may seem complex compared to the automatic numbering and cross-referencing features of standard word processors, but it leaves users in control by turning numbered items and references into objects that they can see and manipulate while the system maintains consistency during document editing. It is also more powerful and flexible than the predefined types of references offered by standard word processors. For example, Microsoft Word 16 for Mac can cross-reference *Headings*, *Bookmarks*, *Footnotes*, *Endnotes*, *Equations*, *Figures* and *Tables*, but not *Articles* or *Claims*, which are used extensively by contract and patent writers. *Numberlets* let users control what types of numbered items they need, providing flexibility within a unified interface.

Textlets for Length Constraints

Word count and character count limits are common in technical documents. For example, patent offices limit the number of claims in a patent, the number of words in the abstract, and the number of characters in the patent title. Standard word processors include tools to count words and characters in a selection, but they require users to reselect the text and recount after every modification. Microsoft Word shows the total word count of the entire document and current selection in real time, but counting the characters in, e.g. a section of the

document requires selecting the text and bringing up a modal dialog. (Fig. 3.18)

Counting textlets, or *countlets*, solve this problem by counting the number of words or characters in a segment of the document and displaying it in the document itself and/or a side panel. As the user edits the text, the counter updates, avoiding the need for special commands or re-selection. The user can set a threshold above which the textlet will signal that the text is too long. Additional metrics could easily be included, such as the text's estimated reading time. Such *timelets* would be useful, e.g. for journalists and authors of video subtitles.

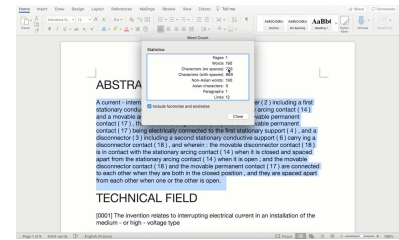


Figure 3.18: Microsoft Word's word count with a modal dialog

Textlets for Exploratory Writing

Study 1 showed how professional technical writers often need to manage multiple alternatives for parts of a document, before deciding or agreeing on which one to keep. Although standard word processors support change tracking, this is insufficient, since it tracks all edits, not the intermediate versions the user may want to keep. Participants must either make copies of the entire document, or use colored text or comments to list alternatives within the document.

Variant textlets, or *variantlets*, let users keep track of the changes made to a selection rather than the entire document.

We were inspired by *Explorable Multiverse Analyses* (Dragicevic et al., 2019), where alternative analyses can be embedded in a research paper and selected by the reader to view them in context. A *variantlet* saves the original content of the selected text. After editing the text, the user can swap it with the original version for immediate comparison, and swap again with the edited version. More sophisticated behaviors can be added to manage multiple alternatives, such as displaying the alternatives side by side or displaying the changes in a manner similar to the track changes mode of word processors. *Variantlets* provide greater control on version management by supporting local versioning rather than traditional document-level versioning. A similar concept is featured in *Variolite* (Kery et al., 2017) for code editing.

Generative Power

The previous examples show the power of textlets to support a variety of tasks. We have also identified other behaviors for textlets that could be useful for a wider range of use cases:

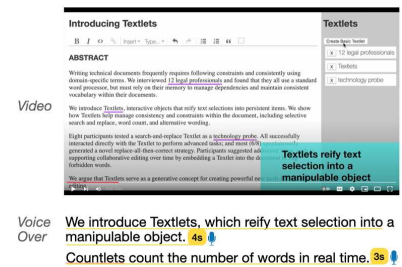


Figure 3.19: A mock-up of *timelets*. The user can record her voice and see the time she takes to read the text.

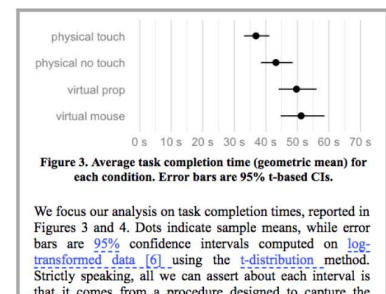


Figure 3.20: Dragicevic et al. (2019)'s explorable multiverse analyses

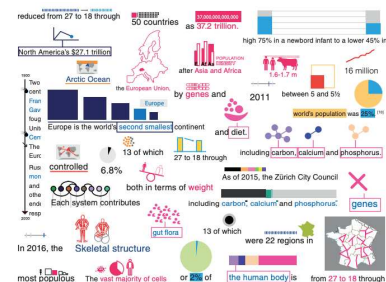


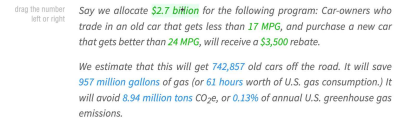
Figure 3.21: Goffin et al. (2017)'s word-scale graphics embedded in text documents

- Attaching comments, summaries, translations, word-scale graphics (Fig. 3.21) or emojis and adding decorations to a textlet, e.g. highlighting or badges, to annotate the document;
- Supporting arbitrary computed content, such as Bret Victor's Reactive Documents⁵, where a textlet is defined by a formula that refers to other textlets, as in a spreadsheet;
- Controlling the style and formatting of the text by associating style attributes with the textlet;
- Crowdsourcing the text of a textlet or a collection of textlets for reviewing or grammar checking, as in Soylent (Bernstein et al., 2015); and
- Organizing textlets freely in a canvas to help analyze or annotate the content of a document

The generative power (Beaudouin-Lafon, 2004) of textlets comes from the combination of a set of behaviors:

- Navigating to the text of the textlet in the document;
- Selecting the text of the textlet, leveraging all the existing commands that act on the selection;
- Replacing/modifying text either based on user edits or automatically;
- Modifying the style of the text;
- Adding decorations that are not part of the text itself; and
- Representing and manipulating textlets in a separate view, such as a list in a side panel.

Generative power also comes from the ability to create textlets not only directly, by selecting text in the document, but also automatically, by using *grouplets* that identify and live-update a set of matching textlets. *Grouplets* let users deal with dynamic collections of text in a concrete way, whereas standard word processors typically offer advanced, specialized commands that users hesitate to learn and use. Although textlets may involve more actions than these specialized commands, we argue that users are more likely to try them, and will save time



drag the number left or right

Say we allocate \$2.7 billion for the following program: Car-owners who trade in an old car that gets less than 17 MPG, and purchase a new car that gets better than 24 MPG, will receive a \$3,500 rebate.

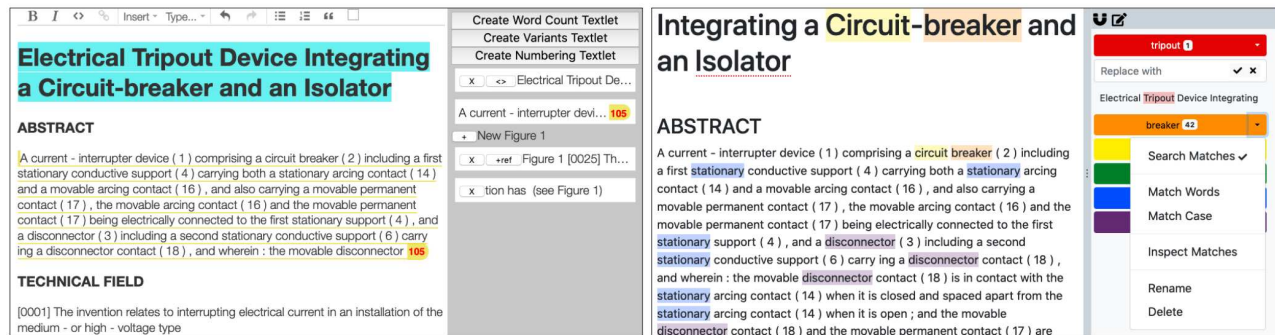
We estimate that this will get 742,857 old cars off the road. It will save 957 million gallons of gas (or 61 hours worth of U.S. gas consumption.) It will avoid 8.94 million tons CO₂e, or 0.12% of annual U.S. greenhouse gas emissions.

Figure 3.22: Bret Viktor's computed text. Source: What can a technologist do about climate change

⁵ <http://worrydream.com/Tangle/>

compared to the manual solutions users resort to.

3.6 Textlets User Interface



In order to demonstrate the concept of textlets, we created a proof-of-concept implementation with four types of textlets: word count (*countlets*), text variants (*variantlets*), numbered references (*numberlets*), and search-and-replace (*searchlets*). These textlets address multiple use cases described in the interview study.

We created two prototypes⁶ as plugins to the ProseMirror⁷ web-based word processing toolkit. The first prototype (Fig. 3.23, left) was developed internally as our first proof of concept and implements *countlets*, *variantlets* and *numberlets*. The second prototype (Fig. 3.23, right) implements *searchlets* and was developed in an iterative process with the participants in the interview study, where it was used as a technology probe (Hutchinson et al., 2003).

The main window contains the text document, with a traditional toolbar for basic formatting at the top, and a side panel dedicated to textlets on the right. The panel features a toolbar for creating new textlets and the list of textlets themselves. It also features *grouplets*, with their list of textlets below them. A textlet is created using any of three techniques:

- Selecting the text content in the document and clicking a creation tool in the toolbar;
- Clicking a creation tool in the toolbar and selecting the text content in the document; or

Figure 3.23. First prototype with the side panel showing a *variantlet*, a *countlet* and a *numberlet* containing a numbered item and a reference, and their visualization in the document.

⁶ See a video illustration: <https://youtu.be/9xD1lhFVsKU>

⁷ <http://prosemirror.net>

c) Entering a keyboard shortcut.

These techniques are also used to create *grouplets*, depending upon their type: some *grouplets* require a text selection, others not, and some may require additional information. Each textlet has a context menu that lets users navigate to the original text in the document, select that text, and delete the textlet. The menu also contains textlet-specific behaviors, such as *search* and *inspector* for the *searchlet* (see below).

Countlets

Our implementation of *countlets* (Fig. 3.24) decorates the selected text with a handle at each end. These handles let users change the scope of the textlet. The right handle also displays the word count of the text in the textlet, which is updated in real time as the user edits the content. A right-click on the *countlet* lets users set a threshold. The counter is displayed in red when its value is higher than the threshold. Deleting the textlet simply removes the word count.

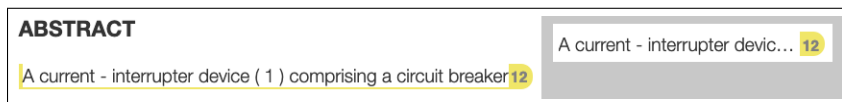


Figure 3.24. *Countlet*: a textlet for counting words.

Variantlets

Our implementation of *variantlets* (Fig. 3.25) supports a single alternative text. When the user creates the *variantlet*, its content is stored. The user can edit the content, and swap it with the stored one by clicking a button in the side panel representation of the *variantlet*. The user can thus easily view and edit the two variants. Combining a *variantlet* with a *countlet* lets the user instantly compare the two lengths by switching between the two alternatives. A more complete implementation of the *variantlet* should include an additional button to save additional versions and a way to navigate through the versions and swap any one of them with the selection.

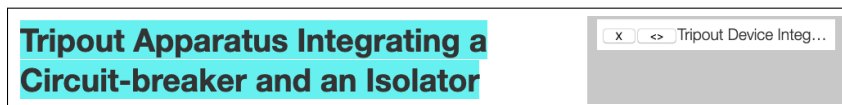


Figure 3.25. *Variantlet*: a textlet for editing local versions.

Numberlets

Our implementation of *numberlets* (Fig. 3.26) uses *grouplets* to create counters, new numbered items for a given counter, and new references

to a given numbered item. The user creates a new counter by selecting a piece of text that contains a number or a hash sign (#), e.g. Article #. This text serves as a template for the numbering scheme. The new counter appears in the side panel as a button. Clicking this button inserts a new numbered item (the *numberlet*) at the cursor position, with the proper number. This *numberlet* is added to the side panel and is also a *grouplet*: clicking it inserts a reference to that item in the text at the cursor position, as well as the corresponding *reference* textlet (or *reflet*) in the side panel.

Numbered items and references are updated when the content of the document changes. The numbering of items follows their order of appearance in the document, and is therefore updated when moving text around. If a numbered item is removed and there are dangling references to it, these references show the error. All updates are immediately visible in both the text and the side panel, ensuring consistent numbering at all times. An additional feature (not implemented) should let users drag a reference textlet below another numbered item to change the reference to that item. This would make it possible to re-attach dangling references.

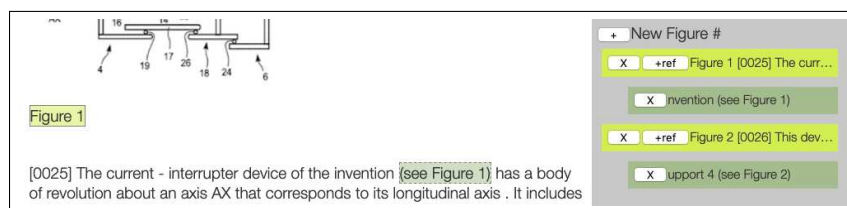


Figure 3.26. *Numberlet*: a textlet for numbering and referencing.

Searchlets

Our implementation of *searchlets* (Fig. 3.27) supports flexible search and replace by extending Beaudouin-Lafon (2000)’s search-and-replace instrument. A *searchlet* is created by clicking the creation tool then specifying the search text, or selecting the search text in the document and clicking the creation tool. Users can also create a blank *searchlet* and then enter the search string. Enabling the *search* behavior finds all occurrences of the search text, highlights them in the document and displays the number of occurrence in the panel. The usual “word matching” and “case sensitive” options become available in the menu to refine the search (Fig. 3.23, right).

Navigating Occurrences Enabling the *inspector* behavior generates the list of occurrences below the *searchlet* in the panel, highlights them in the document, and gives access to the replace capability (Fig. 3.27).

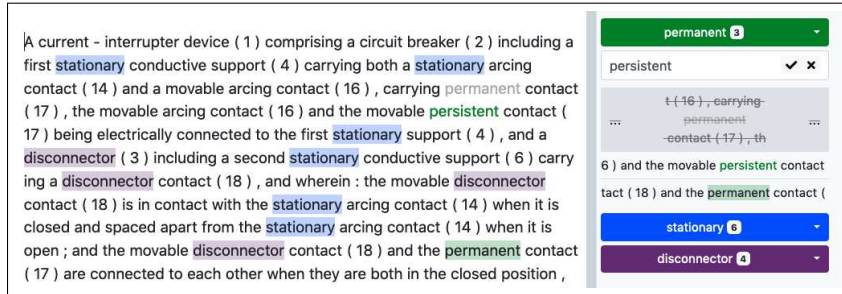


Figure 3.27. *Searchlet*: a textlet for searching and replacing text.

Changing the search string or the search settings re-runs the search and updates the list of occurrences underneath it. Editing the document also dynamically updates the list of occurrences: typing the searched text in the document creates a new occurrence and changing the text of an occurrence in the document removes it from the list of textlets if it does not match anymore.

Each occurrence is a textlet that displays the text surrounding the match in the document and updates it in real time. An occurrence can be expanded by clicking it to better show the context (Fig. 3.27). The user can then click the ellipsis buttons to show more context.

Occurrences can be moved, including under another *searchlet*, giving users flexibility to organize the search results as they see fit. For example, occurrences of different misspellings of a word can be identified with different *searchlets* and then grouped under one *searchlet*, after which they can all be replaced at once. When moved, occurrences adopt the color of their new host *searchlet*. They also “belong” to their new host for the purposes of the *replace-all* action. In the current implementation, they disappear from the list when the search string or the search settings of the new host are changed.

Replacing Text Selecting “Replace Matches” in the *searchlet* context menu (Fig. 3.23, right) shows a text input field for typing a replace string and a button for replacing all occurrences in the list. Each occurrence textlet also includes three buttons that: replace only that occurrence, revert to the previous text, or ignore this occurrence from future replace-all operations. These actions can also be performed in the document itself using keyboard shortcuts.

Replaced occurrences stay in the textlet’s occurrence list until a new search string is entered for that *searchlet*. This lets users work with the occurrences and make changes to the document after they perform a replace operation without losing track of the positions that were

originally matched.

Searchlets extend Beaudouin-Lafon’s previous work (Beaudouin-Lafon, 2000) by supporting multiple simultaneous searches. Each occurrence is reified as an item in the side panel, which supports additional functions such as disabling an occurrence in a global replace, or moving an occurrence to another *searchlet*. Our design is also grounded in our observations of the real-world challenges experienced by a group of professional users.

3.7 Structured Observation

We used our second prototype as a technology probe (Hutchinson et al., 2003) to evaluate *searchlets* with an observational study. We did not run a comparative study with, e.g., Microsoft Word as a baseline because many features that we implemented do not exist in Word or are clearly faster, e.g., a persistently-displayed character count with *countlets*, versus highlighting text and invoking Word’s word count command.

Our goals were to gather feedback, identify potential novel and unexpected uses, and discuss new ideas with the participants in order to refine our design. The study focused on *searchlets*, but we also showed the participants the other textlets from the first prototype. We incorporated suggestions incrementally so that successive participants used slightly different versions of the probe.

Participants: We recruited eight participants: three patent attorneys, one patent inventor (one woman, three men; aged 29-50 who use various versions of Microsoft Word) and four researchers (one woman, three men; aged 24-26 who use LaTeX). Three of the patent attorneys had participated in the interview study. We included researchers because we believe that textlets address the needs of a wider range of users than those in the interview study and authors of research articles must also manage consistency in their papers.

Apparatus: The prototype is a Web application accessed with the participant’s choice browser on their own computer⁸. We provided a 13” MacBook Pro laptop running macOS 10.14 and Firefox 68.0 for participants who did not have a computer at hand. We created two sets of documents to match the participant’s background: two patents and two research papers.

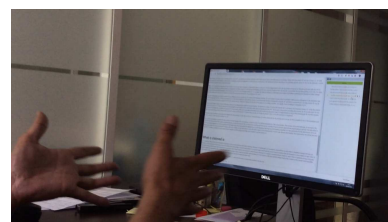


Figure 3.28: A user think-aloud in the observational study

⁸ We choose a web application just because of our prototyping skill set. From participants’ perspective, the interactions and tasks are the same as if they are in a non web document editor. We do not change the work they do.

Procedure: We started by describing the features of the *Textlets* prototype, and gave participants 10 minutes to experiment with it. We used a think-aloud protocol (Fonteyn et al., 1993) and asked participants to perform two similar tasks on two documents: one using the editor of their choice and the other using the *Textlets* prototype. We counterbalanced for document order across participants.

Each task consisted of three small exercises with increasing difficulty: 1) replace a word by another and then change it back; 2) replace a word by another but only in certain contexts; and 3) replace two words with similar meanings with another word, including all relevant variations. Thus replacing “mouse” with “rodent” also requires changing “mice” with “rodents”.

The two tasks, each with three exercises, took approximately 20 minutes. After an interview, participants completed a short questionnaire. The session ended with a debriefing to identify additional use cases and discuss ideas for improvement. We also showed participants the *countlets* and *variantlets* from the first prototype, and asked them to describe scenarios for which they might be useful.

Data Collection: We recorded audio, took hand-written notes during the session, and collected the answers to the questionnaire.

3.8 Results and Discussion

All participants successfully interacted with the textlet prototype and found the tasks representative of their everyday work. The textlet side panel was “faster to use” (P1, P3). It avoids jumping to the main text (P1, P2, P3, P6), so that they can focus on the relevant document parts, thus reducing mental workload. Most participants (6/8) preferred making changes directly with a *searchlet* over Word’s non-interactive side panel. Two participants (P1, P2) asked for even greater interactivity with *searchlets*, such as one-by-one replacement directly from the panel, which we added in a later version, and merging two *searchlets* to apply the same replacement to their occurrences. We added other small improvements based on participants’ feedback, including better colors and icons, and decluttering the textlet interface by using a menu instead of a series of buttons.

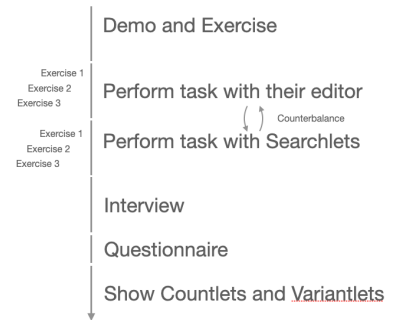


Figure 3.29: A timeline of study procedure

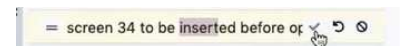


Figure 3.30: Adding the design of individual replacement, undo and ignore, based on participants feedback

Replace-all-then-correct Strategy

Most participants (7/8) used a one-by-one search-check-replace strategy with both Microsoft Word and LaTeX: They search for the word, go to each occurrence in the main document to check the context and then perform the replacement, either by clicking a button or retyping.

Participants used a different strategy with textlets, which we characterize as search-overview-replace. They started by creating one or more *searchlets*, scanned the overview of the occurrences to see the variations and assessed which ones to replace. P1 said: *“I can see immediately what variations are in the text [from the side panel]. So I see it will work by replacing all matches”*.

The combination of overview and contextual information around each match encouraged participants to spontaneously develop two different strategies for the final search-overview-replace step: Six participants used a replace-all-then-correct strategy, first replacing all occurrences, then checking each replacement in the overview list for errors, which they corrected either with the ‘revert’ button or by retyping in the document. The other two participants (P6, P7) used an ignore-replace-all strategy, first pressing the ‘ignore’ button to skip outliers, then applying ‘replace-all’, similar to the ‘perfect selection’ strategy in (Miller and Marshall, 2004). In summary, although participants were reluctant to use replace-all with their regular word processor, they felt comfortable using the *searchlets*’ replace-all and quickly developed strategies for selective replacement.

Persistent Selection: Keep Track, Individual Undo

Although both Microsoft Word and TexWorks (L^AT_EXeditor) provide an overview list of all search occurrences, they do not track them by position. By contrast, *searchlets* create persistent occurrences that help users keep track of what happened. P5 felt more confident with the prototype, saying: *“Here (pointing at the side panel) I can see the changes in context. It helps me [and] reassures me that I did the right thing.”*

Furthermore, *searchlets* let users check the results of their previous replacements. The overview of occurrences persists in the panel even as users edit the document. This differs from other word processors that clear the search whenever the user types in the document, which forces users to tediously re-enter the search text. For example, P3 said: *“I have this list of all the occurrences. When I want to do some replacements, I choose some of them and I keep the whole list that I can always check [in the side panel]. This is quite important...I do not need to proof-read the whole*

text."

Because each occurrence is also a textlet with its own history of changes, it can be undone individually and ignored in a replace-all command while still remaining in the overview list. These novel behaviors contributed to most participants (6/8) spontaneously adopting a replace-all-then-correct strategy. For example, P₃ said while performing a task: *"Maybe it is better to replace all and check the ones that do not work."* This suggests that making changes persistent, visible and reversible increases users' trust in the system.

Representing Constraints

One participant suggested embedding a group of *searchlets* as *"a highlight [feature] for forbidden words."*(P₄), arguing that making co-authors aware of these words as the document circulates would help them maintain consistency and improve collaborative editing. Textlets can thus embody constraints and serve as an active guideline when embedded in a document.

Feedback for countlet and variantlet

Participants also described situations in which they wanted to use *countlets* and *variantlets*. For example, P₃ wanted to count the words in patent abstracts: *"I think this could be very useful because many times you are going to count words and [the system] does not keep it."* P₄ wanted to use *variantlets* as a local versioning tool: *"If you can version one paragraph [instead] of the whole document, it could be very useful. In that case, you can track which part you have changed."* Future work should compare *variantlets* to "suggested edits feature" in most document editor.

Scalability and Limitations

A potential limitation of our approach is scalability: Searchlets that generate large numbers of matches or large numbers of textlets and grouplets in the side panel could cause problems when dealing with large documents. We did not observe such problems during the study, probably due to its short-term nature. Several features mitigate scalability issues: users can collapse grouplets, e.g. search results, to save space, or disable them to remove highlighting in the main text. Scrolling between the document and the side panel could also be synchronized so that the side panel can only display relevant textlets to current page. future textlets could combine behaviors, e.g. *countlet +variantlet*, to save space.

One participant found that *searchlets* might be less useful in simple cases with few matches or variations of the same word: “[With] only 3 matches, I would like to change it directly in the main text” (P₃). Another participant wanted *searchlets* to support regular expressions so that she can specify search patterns. Both features could easily be supported in a future prototype.

Summary: This study demonstrated the value of *searchlets*, the most complex textlet we developed, as well as the potential of other textlets. By turning search matches into persistent objects that users can manipulate directly, users were willing to use functions, such as replace-all, that they otherwise avoid with traditional word processors. They also spontaneously devised novel strategies and appropriated the textlet concept in unexpected ways, such as embedding *searchlets* for forbidden words. This study provides evidence for the validity of the textlet concept, and encourages us to further develop and assess the textlets we have developed, as well as design new ones.

3.9 Conclusion

This chapter investigated the common task of document editing. We interviewed 12 legal professionals about their practices to meet the constraints and consistency requirements of technical documents. This revealed the limitations of current text editing tools that technical writers are reluctant to use advanced features of their word processors, and must instead rely on their memory to manage dependencies and maintain consistent vocabulary within their documents. We introduced a simple concept called *Textlets*, interactive objects that reify text selections into persistent items. We showed five use cases where textlets can be applied to support consistent reuse, term and reference consistency, word count constraint, and exploratory writing. Our observational evaluation showed that participants successfully used the textlet to perform advanced tasks, and can quickly adapt to and appropriate textlet.

What have we learned about representation and manipulation? The original representations in the Star user interface are mostly based on physical objects in an office, such as paper and folders. This choice was based on the design principle of *Familiar User’s Conceptual Model*: “we hoped this would make the electronic “world” seem more familiar, less alien, and require less training.” (Smith et al., 1982). I believe that today’s users are already quite familiar with these representations and our interview

study has shown that they are not enough to meet the needs of legal professionals. This chapter thus introduced a new representation, reified text selection object, to support some of these unmet needs. The design of this new representation is less based on existing office objects but more on semantic objects in users' mind. The process of reification makes this semantic object *persistently*⁹ visible, which *"lets users conduct experiments to test, verify, and expand their understanding"* (Smith et al., 1982). Searchlets is a good example as one participant said: *"I have this list of all the occurrences. When I want to do some replacements, I choose some of them and I keep the whole list that I can always check [in the side panel]. This is quite important...I do not need to proof-read the whole text."* (P3)

This new representation (or object) is more like a function icon (Smith et al., 1982) that can perform actions. For example, textlets can perform actions on the main document, e.g. count words and search occurrences. Another way to look at it is to see this new representation as an instrument (Beaudouin-Lafon, 2000) that users can manipulate to act on the main content. Instead of going through a myriad of commands, users directly act on the textlets to perform a variety of actions to change the main text. I argue that this improves the users direct engagement with the main text and simplifies the user's conceptual model.

In summary, I expand our understanding of representation by introducing a new representation based on the semantic object in users' mind, rather than the existing physical objects. The persistently visible characteristic of this new representation allows users to verify and expand their understanding of the content. Users can directly act on this new representation as an instrument to engage with their content.

⁹ In a conventional word processor, when a user selects a piece of text, it is highlighted and the user interacts with it, e.g. by dragging it. It is visible but not persistently visible like the textlets

4

Representation for Document Analysis

The work reported in this chapter was joint with Junhang Yu, a master student intern, and Alexandre Ciorascu and Raphael Bournet, two undergraduate interns. I secured the collaboration with the European Patent Office, conducted the interview study with patent examiners and took the lead on the project. Junhang Yu conducted the interview study with scientists. Alexandre Ciorascu, Raphael Bournet and myself implemented the prototype. I conducted two evaluation studies together with Junhang Yu.

The work is being published at ACM CHI2022 (Han et al., 2022) (In Print).

Through the participants in Chapter 3, I had the opportunity to collaborate with the European Patent Office. This gave me a chance to study another key aspect of knowledge work: document analysis. In this chapter, I study the patent examination process in-depth and find that all examiners use specialized tools for managing text from multiple documents across various inter-connected activities, including searching, collecting, annotating, organizing, writing and reviewing, while manually tracking their provenance. I became interested in testing the generalizability of the findings. This leads me to collaborate with Junhang Yu to interview scientists about their literature review process, which also involves manipulating a set of key documents. The study shows similar findings. We created Passages, a cross-document representation that can be reused, manipulated and shared in multiple applications. Two evaluative user studies show that participants found Passages both elegant and powerful, facilitating their work practices and enabling greater reuse of information.



Figure 4.1: Analyze multiple documents

4.1 Context

Document analysis is a complex activity that involves juggling documents, most text-based. Studies of knowledge work (Adler et al., 1998; Marshall et al., 2001; O'Hara et al., 1998) highlight the complexity of this process, which involves active reading, search, retrieval, annotation and writing activities (Fig. 4.2). Many professions have developed specialized productivity software (Oleksik et al., 2012; Zhang et al., 2020) to support this process. For example, legal professionals take advantage of specialized online services to find and cite relevant law books and articles that serve as precedents for their cases (Marshall et al., 2001). Scientists follow a similar process when searching for related articles that support their arguments, using the research literature to “describe, summarize, evaluate, clarify and/or integrate the content of primary reports” (Cooper, 1998).

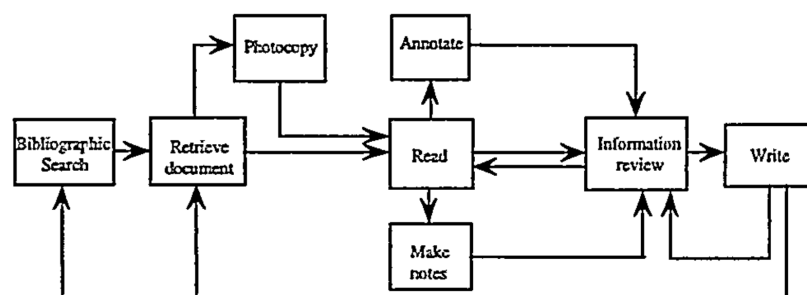


Figure 1. A model of document related activities of library users at different stages of their research work.

Figure 4.2. O'Hara et al. (1998)'s model of various document related activities of library users.

Unfortunately, current software typically traps information into *information silos* (More details in Section 5.2), making it difficult to reuse content or obtain a unified view of the material (Oleksik et al., 2012). This poses a serious design challenge, since adding yet another tool risks complicating, rather than supporting, the complex, multi-faceted nature of document management. We first need to better understand how today's knowledge workers currently perform active reading in their document management process, with particular emphasis on their use of existing software tools.

A second key problem knowledge workers face is capturing and maintaining *provenance* (Cheney et al., 2009)—tracking the source of a document and returning to it—despite varied formats and diverse, non-interconnected sources. Evans et al. (2020) argue that tracking prove-

nance is a key factor in the media industry's current "crisis in journalism" (Starbird et al., 2018), and Flinham et al. (2018) show how identifying an article's source is essential for evaluating its trustworthiness. Similarly, scientists rely on provenance to ensure reproducibility and uncover potential plagiarism (Cheney et al., 2009), and Jensen et al.'s (Jensen et al., 2010) longitudinal study of knowledge workers finds that provenance cues significantly aid recall. We thus seek to understand how today's knowledge workers manage provenance.

4.2 Related Work

Knowledge workers engage in active reading of documents, both to make sense of them and to evaluate each document's relevance to their needs. When they find useful documents, they must also keep track of their sources, or maintain the provenance of each document, in order to cite them accurately and return to them if needed. We thus review related work with respect to three key areas within knowledge work: active reading of documents, sensemaking, and information provenance.

Active Reading of Documents

Practices. Knowledge workers engage in "active reading" of documents, which involves interweaving reading with a variety of associated activities (Adler et al., 1998; Golovchinsky, 2008; Golovchinsky et al., 1999). For example, O'Hara et al. (1998) interviewed researchers about their library use and characterise scholarly research as "a complex process of searching, information retrieval, reading, information extraction and recording by annotation and note-taking, information review, and writing new compositions (such as papers or thesis chapters)". (Fig. 4.2) In a diary study of knowledge workers' reading practices, Adler et al. (1998) found that reading goes hand-in-hand with writing, and that knowledge workers often read multiple documents in parallel, a finding echoed by Tashman and Edwards (2011a)'s study of active reading (Schilit et al., 1998). Marshall et al. (2001) found that law students switch frequently and fluidly between annotating, organizing and writing. Although this research shows that reading often occurs with other activities, we do not fully understand the relationships among these activities, nor how current systems support transitions across them.

Systems. Multiple systems support document reading, including ac-

tive reading (Hinckley et al., 2012; Marshall et al., 2001; Tashman and Edwards, 2011b), active diagramming (Subramonyam et al., 2020) and active note-taking (Hinckley et al., 2007), as well as more specific tasks such as annotation (Romat et al., 2019; Yoon et al., 2013,1) and navigation (Alexander et al., 2009; Woodruff et al., 2001).

Several systems explicitly support document-reading practices, such as InkSeine (Hinckley et al., 2007), which accommodates searching during active note taking. By transforming search queries into first class objects, users can quickly capture and save search results into their ‘ink’ notebooks. GatherReader (Hinckley et al., 2012) builds on the finding that reading co-occurs with writing and cross-referencing documents (Adler et al., 1998), as well as gathering pieces of information (Marshall and Bly, 2005). It supports collecting multiple objects via a temporal visual clipboard, using pen and touch interaction. LiquidText (Tashman and Edwards, 2011b) offers readers highly flexible and malleable documents, with fluid representations and a multi-touch gesture-based interface. These systems focus on reading-related activities, particularly note-taking (Hinckley et al., 2012; Tashman and Edwards, 2011b) and searching (Golovchinsky et al., 1999; Hinckley et al., 2007). On the interface level, these systems also have similar layout: document on the right side and a note space on the right. (See Fig. 4.3, 4.4, 4.5). However, we are also interested in helping readers manage active reading while dealing with multiple document-based applications.

Sensemaking

Practices. Sensemaking is defined as creating a mental representation of an information space to support the user’s goals (Russell et al., 1993). Sensemaking research explores relationships across different knowledge work activities, especially searching, capturing and organizing information. Kittur et al. (2013) differentiate between two main phases—first information seeking then sensemaking—and highlight the cost of creating structure too early in the foraging process. Sellen et al. (2002) identify six categories of knowledge workers’ use of the web, and found that most searches lead to further activities, especially referring back to a result and incorporating it into a document. They argue that users need more flexible ways of saving text and search results. Other studies of web-based information seeking highlight users’ revisitation behavior (Adar et al., 2009; Ma Kay and Watters, 2008) and their use of multiple windows (Dubroy and Balakrishnan, 2010; Wagner et al., 2012; Weinreich et al., 2008), as well as activities beyond searching (Capra et al., 2010; Sellen et al., 2002). A key sensemaking

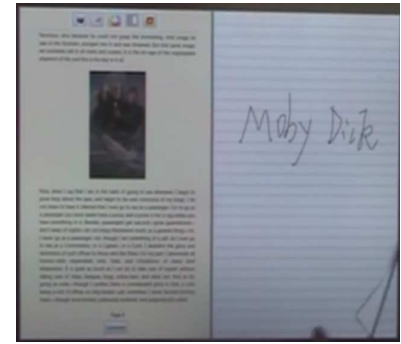


Figure 4.3: Hinckley et al. (2012)’s Gather Reader

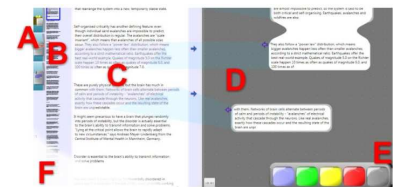


Figure 4.4: Tashman and Edwards (2011b)’s Liquid Text

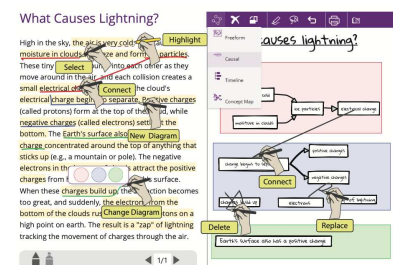


Figure 4.5: Subramonyam et al. (2020)’s TexSketch

activity involves creating external representations, such as tables (Liu et al., 2019), hierarchies (Abrams et al., 1998), diagrams (Subramonyam et al., 2020), networks (Halasz et al., 1986), canvases (Koch et al., 2020), and more (Shipman et al., 1995). Unlike reading, sensemaking focuses more on searching, capturing and organizing information. We are interested in the knowledge worker’s complete workflow, encompassing the interconnection of reading and sensemaking activities.

Systems. Multiple systems support sensemaking, including searching (Hearst, 2009; Mackinlay et al., 1995; Rao et al., 1995; White and Roth, 2009), collecting (Hong et al., 2008; Kittur et al., 2013) and organizing (Card et al., 1996; Halasz et al., 1986; Marshall et al., 1991; Robertson et al., 1998) information, as well as systems for creating tables (Pirolli and Rao, 1996), graphs and hypermedia structures (Russell et al., 1993). Hunter Gatherer (schraefel et al., 2002), Unakite (Liu et al., 2019), ScratchPad (Gotz, 2007), and (Dontcheva et al., 2006)’s web summarization system (Dontcheva et al., 2006) support searching, collecting and organizing information, but do not extend to writing or reviewing final documents, e.g. summaries and analysis reports (Tashman and Edwards, 2011a), which are also a part of knowledge work (O’Hara et al., 1998). Other systems such as CiteSense (Zhang et al., 2008) and Entity Workspace (Billman and Bier, 2007) provide an integrated environment with a multi-panel interface, but do not support the flexibility needed for the diverse and changing nature of knowledge work (Kidd, 1994; Tashman and Edwards, 2011a). More generally, such systems are not designed to support the interconnected activities required for reading and sensemaking.

Provenance

According to Pérez et al. (2018), “Provenance refers to the entire amount of information, comprising all the elements and their relationships, that contribute to the existence of a piece of data.” Jensen et al. (2010)’s investigation of the provenance of files and documents on knowledge workers’ desktops shows that it helps reveals their work patterns. Researchers have developed several systems that accommodate file provenance, e.g. for version management (Karlson et al., 2011) and file retrieval (Ghorashi and Jensen, 2012; Soules and Ganger, 2005; Stumpf et al., 2007). TaskTrail (Stumpf et al., 2007) tracks file provenance from copy-paste and save-as commands to help users re-find documents. Versionset system (Karlson et al., 2011) tracks copy relationships among files to help users handle version management. Given our currently highly networked world, Lindley et al. (2018) challenge the original file metaphor (Smith et al., 1982). They intro-

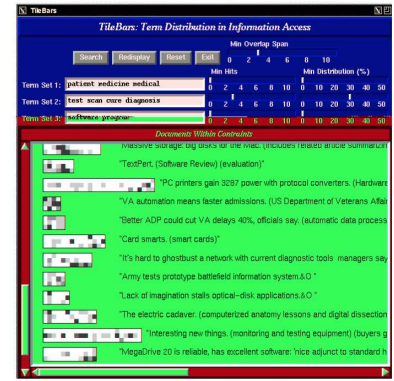


Figure 4.6: Hearst (1995)’s Tile-Bar

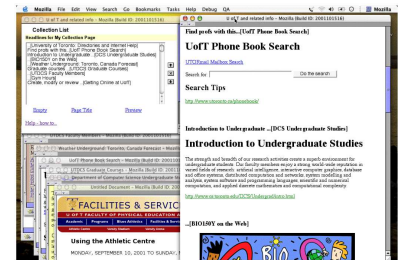


Figure 4.7: schraefel et al. (2002)’s HunterGather

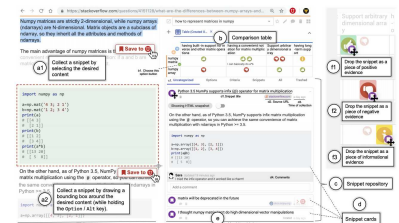


Figure 4.8: Liu et al. (2019)’s Unakite

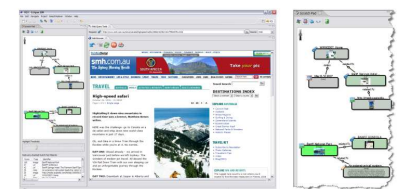


Figure 4.9: Gotz (2007)’s Scratchpad

duce the concept of ‘file biography’ to capture the provenance of each file and let users track how it propagates. Despite research showing how knowledge workers often focus on *snippets* of information (Liu et al., 2019; schraefel et al., 2002; Subramonyam et al., 2020; Tashman and Edwards, 2011a), none of these systems directly support *snippet* provenance.

Although the above systems offer useful support for different aspects of knowledge work, from reading and making sense of documents, to determining their provenance, we lack understanding of the overall knowledge work process. We decided to examine two groups of knowledge workers, patent examiners and research scientists, who read, interpret, organize and share documents across multiple applications over long periods of time, to contribute insights to the design of more advanced software tools.

4.3 Study 1: Interviews with Patent Examiners

The legal profession offers an extreme example of document-intensive knowledge work, with numerous design challenges (Adler et al., 1998; Marshall et al., 2001). Chapter 3 focused on lawyers and patent writers (Han et al., 2020). Here, I explore the issues faced by a related group—*patent examiners*—who must search for, analyze and communicate effectively about complex legal documents in order to make critical decisions about intellectual property.

Participants: We recruited 12 participants (11 men, 1 woman; mean=13 years of experience), from a major patent organization. Nine are full-time patent examiners, and three are part-time patent examiners—two software engineers and a product owner from the software development team. All use the organization’s internal software applications to search and examine patent applications, and each specializes in a particular technical field, including antennas, biotechnology, CPUs, displays, medicine, optics, and polymers.

Procedure: Each interview lasted two-three hours. In addition to collecting basic information about each patent examiner’s background, we conducted story interviews (Mackay, 2002) where they were asked to describe a recent, memorable event related to searching and examining a specific patent application. Participants were encouraged to show the specific documents and software involved, and to re-enact each search process, step by step. We also interviewed members of the

development team for a different perspective on the challenges faced by patent examiners as well as their goals for future system development.

Data Collection: All interviews were screen recorded and transcribed. We also took hand-written notes and collected examples of documents and software screens.

Analysis: Junhang Yu and I analyzed the interviews using reflexive thematic analysis (Braun and Clarke, 2019). We generated codes and themes both inductively (bottom-up) and deductively (top-down), looking for breakdowns, workarounds and user innovations. After interviewing six participants, we conducted the first analysis, grouping codes into larger categories. We discussed any disagreements and rechecked the interview transcripts to reach a shared understanding. We arrived at the final themes after two iterations.

4.4 Results and Discussion

We collected 13 unique patent examples from the 12 patent examiners. For confidentiality reasons, participants sometimes needed to hide information and would jump to another, similar example to illustrate the relevant aspect of the process, and would then return to the main example. The next sections describe the patent examiners' current document management process and six challenges that patent examiners face during the search phase, followed by a summary and implications for design.

Patent Examination Process

Awarding a new patent involves six basic steps: filing, search, examination, grant award, opposition and appeal¹. We focus on the search step, illustrated in Fig. 4.10, which judges the patentability, or novelty, of an application relative to all prior art. We chose it because involves a variety of interrelated document-based activities and is common to all patent examiners.

Patent examiners first read the patent application to understand its claimed invention (1). If the application is re-submitted, they must also read other related documents, such as previous search reports and letters from the attorneys, to understand the full context (1). Examiners actively annotate the application and issue queries (1.1) that search

¹Note that this practice differs between Europe (<https://www.epo.org/learning/materials/inventors-handbook/protection/patents.html>) and the United States (<https://www.uspto.gov/patents/basics/patent-process-overview>)

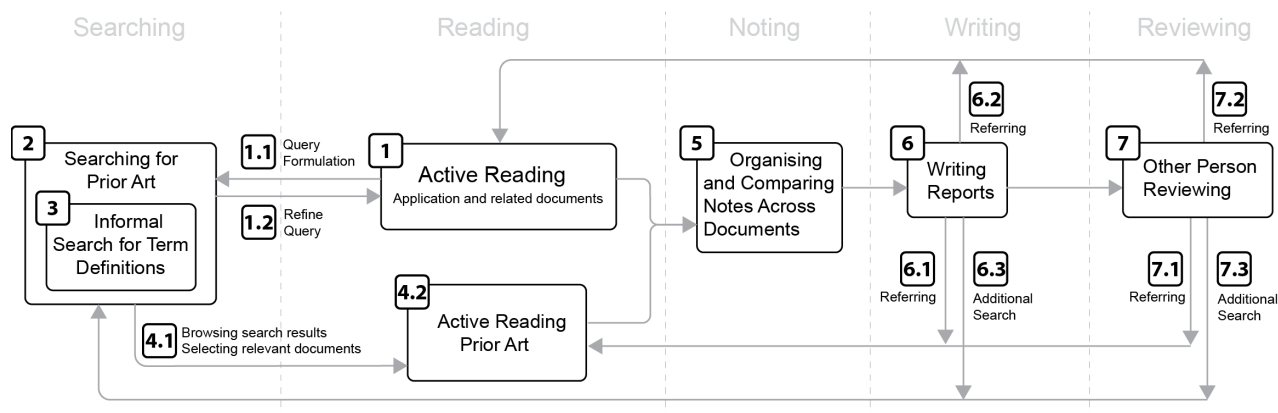


Figure 4.10. Patent examiners engage in a series of document-related activities when searching for prior art, including formulating search terms; reading and annotating documents; and writing and reviewing their own and other examiners' reports.

for prior art (2). Because of the highly technical nature of patents, examiners must often search for new terminology and term definitions using specialized web services (3). Next, they browse the search results to identify relevant documents (4.1) and then read them (4.2). Study participants reported reading from 100 to 300 documents for a single patent search. The examiners next compare selected documents to the patent application, using criteria such as novelty, inventive steps and clarity (5). This reading and searching process is highly iterative: Examiners may learn a new term from one prior art document and use it to refine subsequent queries (1.2) to search for (1.1) and read additional documents (4.2). Several participants mentioned that they take notes to keep track of their work. (5)

This process usually results in identifying three to five highly relevant prior art documents. Examiners then write a search report, drawing from their personal notes (6). The patent institute requires them to cite specific, sentence-level evidence in their report. They often refer back to the patent application (6.2) and prior art documents (6.1), and sometimes search for additional documents (6.3) if they realize that something is missing. The search report is then reviewed by the chairman of the patent division (7). This review also involves frequent references to the application (7.1) and prior art documents (7.2), and may also result in additional searches (7.3).

The search phase usually leads to the examination phase, when the patent office decides whether or not to grant the patent. Three patent examiners examine the application, one of whom maintains contact with the patent attorney. The dialogue between the examiner and the patent attorney sometimes results in modifications to the initial patent application.

Six challenges

We identified six challenges that patent examiners face during the search step: transferring information across applications is cumbersome; organizing personal notes risks duplicating information; re-finding passages from multiple sources is tedious; automatic drafting ignores human expertise; sharing knowledge has few rewards; and informal search results disappear.

Transferring information across applications is cumbersome. The patent organization has developed over 20 specialized applications for patent examiners. Most are designed for specific activities such as task management, report creation, document viewing, and searching for prior art. Others are designed for highly specific tasks. For example, a chemistry examiner (P7) showed us a specialized application for finding the CAS (Chemical Abstract Service) registry number when information found on a website or database entry is messy or incomplete.

Despite the availability of these applications, examiners must spend a great deal of time choosing among them and setting them up before they can perform any real work. P2 said: *“One of the problems that examiners have is that there are many different tools and there are many different times that you have to go here and there. So it is very cumbersome.”* They must also follow a preset process when using these systems. For example, P4 must open three different applications just to view a document *“so that it has the right dossier number and then it is synchronized.”* This produces many unnecessary windows that are hard to manage and clutter examiner’s screen. Two examiners (P3, P4) developed workarounds to avoid the rigidity of this procedure-based interaction. For example, the classification tool only sends codes to three tools, none of which is the one he wants. So P3 copy-pasted the tool’s entire classification code ² into a Microsoft Word document, which lets him easily search for and copy the codes he needs.

Organizing personal notes risks duplicating information. Examiners collect information snippets and add them to their personal notes. This not only supports the current task, but also helps them in the future. P6 called these notes a *“letter to myself”*—when the application comes back for re-examination in a year, she will not have to *“spend the same amount of time again to re-familiarize [herself] with the file.”*

Some examiners structure their notes as a comparison table (P1, P6, P9), free-form canvas (P7) or as linear text (P6, P10, P11). For example,

² Patent classification is a system for organizing all patent documents and other technical documents into specific technology groupings based on common subject matter. Source: <https://www.uspto.gov/patents/search/classification-standards-and-development>

P6 created a table in Microsoft Word to compare the application and prior art documents. She considers the table to be *“a mental help to remember what the claims are about and what are the features.”* P7 pastes text snippets and screenshots into Microsoft OneNote so that he can have *“all the information at a glance”*. Other examiners avoid organizing personal notes into an intermediate structure, and instead treat the text editor where they write the final report as their note-taking space. For example, P11 keeps the claims in his head and tries *“to start the communication [with the attorney] as soon as possible”*.

Examiners find it tedious to copy-paste text snippets from the organization’s internal applications into their personal notes. They also find it difficult to maintain links back to the original text, even though they frequently revisit these snippets when they edit or review their reports. P6 found it *“really painful...it is really nasty. It is the duplication of information...Copying is the maximum I would like to invest.”*

Re-finding snippets from multiple sources is tedious. All examiners reported manually re-finding text snippets from multiple sources, including prior art, application software and other related documents, in order to verify or cite them in their report. They developed diverse strategies, from using the search function (P2) or manually keeping a “link” (P7), to reduce the time involved in collecting snippets. This task is even more extreme for a chairman, who has to locate and verify many cited passages during the review process. P2 (a chairman) even has to manually count line numbers: *“If I see here that this claim is supported by the description on page 5, line 15-17, I go on page 5 and have to count to the line and I find [it].”*

Although citing the snippets is tedious, examiners agree on its importance for making evidence-based decisions and easing the later communication process. For example, P11 (a junior examiner) said: *“I have learned that, in the examination argumentation phase, it can be really helpful to cite the specific passage. If you read from the line that this functional feature is present, it does not mean that everybody else will read it. You have to explain it. If you don’t, you will get a lengthy letter back. You have to explain it anyway and this is a waste of time for both parties. This is why I am often quoting complete sentences, saying that this feature is really there, don’t come back to me and say it is not there.”*

Automatic drafting ignores human expertise. The organization developed an application for automatically generating a report from a form the examiners fill out. Although intended to improve report-writing productivity by reducing copy-pasting and re-writing, many

examiners have mixed feelings about the system and are reluctant to use it. They emphasize the importance of having a human select the right text, and the need for human-to-human communication in their intellectual work. P6 said that her team dislikes it, so she dropped it: *"I did not see the benefits. I was typing as much as before and the communication is much longer because it has all these standard sentences that I don't really need to repeat every time."* For her, the intellectual part is about *"selecting the passages"*. P8 and P10 emphasized that the human value lies in communication and dialogue with applicants. P8 said: *"If we are moving in the direction where the communication will be automatically generated by computer, we will be sending the message that we don't really do the intellectual work. So [the patent attorneys] will be less likely to be convinced, the same way I am less likely to be convinced."*

Sharing knowledge has few rewards. Almost half the examiners (5/12) have their own library of synonyms for their respective domains. These libraries are a form of knowledge that examiners have accumulated over many years and can be both useful to themselves and to others. The main ways examiners share knowledge is through talking to people (P4, P5), exchanging Microsoft Excel or text files (P11, P12) or using an internal system (P6, 11). P6, an experienced examiner, said that if he works in an unfamiliar field, he just goes next door and asks his colleague. This is consistent with research that shows that people prefer obtaining information from other people within an organization (Hertzum and Pejtersen, 2000). Two examiners (P11, 12) also mentioned exchanging their personal synonym stored in Excel files with colleagues.

Although the organization has an internal system for sharing synonym libraries, one participant said that examiners gradually stopped using it, since the focus on improving productivity discouraged them from taking the extra time needed to create reusable libraries. Others were willing to share their synonyms but received no benefits if they created something that exceeded their personal needs. One exception is an examiner with a huge personal synonym database—he has around 500, in contrast to his colleagues' 20 or 30—who constantly refines and improves it, and emails it to a dozen colleagues every six months. His role is similar to the "translator" identified in Mackay's study of exchange of customization files (Mackay, 1990).

Informal search results disappear. Examiners not only search for prior art documents but also for term definitions and synonyms on the web to help them understand an unfamiliar domain. The internal search tool keeps track of the main search activity, which is the

search for prior art. However, queries and results of informal searches remain ephemeral, causing repetitive re-searching for the same information. For example, after starting a search, P5 would *“go back and maybe have another look at the dictionary or the Wikipedia”* because it gives him new ideas for a search query.

Summary. This study highlights the difficulties encountered by patent examiners when managing interconnected document-intensive activities through multiple specialized but separate applications. Examiners establish multiple personal strategies for organizing information, but constantly struggle to maintain its provenance. Examiners are worried that the shift to ever more automated systems will reduce the human value that they add, and they want to maintain control of their human-to-human communication with other parties. In parallel, the push for increased productivity reduces incentives to share knowledge, which may affect the quality of their work.

4.5 Study 2: Interviews with Scientists

Reviewing and analyzing the scientific literature has a number of similarities to the patent review process (Federico et al., 2017). We decided to broaden the scope of our study to include this more open-ended form of knowledge work, and thus conducted a second series of interviews with research scientists³.

Participants. We interviewed 12 scientists (4 women, 8 men) from the following disciplines: computer science, game design, mechanical engineering, physics and psychology.

Procedure. Each interview lasted 45 - 60 minutes and was conducted by video. We conducted story interviews where each participant described a recent example of searching the research literature for a literature review or related work section.

Data Collection and Analysis. All interviews were screen-recorded and transcribed. The same two researchers analyzed the data using the same reflexive thematic analysis approach described in Study 1. We checked the interview transcripts to reach a shared understanding and arrived at agreement after two iterations.

³ Junhang Yu conducted the interviews and I participated in the analysis.

4.6 Results and Discussion

We collected a total of 33 examples of specific literature search problems encountered by the 12 scientists, approximately three stories per scientist. Although the overall process of searching for the research literature is more open-ended than that of patent examiners, it is similarly complex. We identify four challenges that scientists face when searching the literature, three of which correspond directly with those of Study 1: transferring information across applications is hard; re-finding papers from multiple sources is tedious; reusing notes is difficult; and search results are easily lost..

Transferring information across applications is hard. Scientists, similar to patent examiners, use multiple specialized applications for activities such as reading, note-taking, document management and writing. Many researchers expressed their frustration when trying to make these systems work together. For example, P5 creates a new Markdown file in Obsidian⁴ every time he decides to read a paper in depth. He must manually create a link between his notes and the source paper. Since many note-taking applications focus on supporting stand-alone reading, researchers struggled to transform their notes into their final document. Most tediously copy-paste their notes into the text editor, then re-organize them and manually add citations. For example, P10 complained that because Overleaf⁵ did not display his personal notes from JabRef⁶, he had to read the paper again to figure out the exact content. Although the details differ, the scientists face the same challenge as the patent examiners, which is to transfer information effectively across different applications.

⁴Markdown (<https://daringfireball.net/projects/markdown/syntax>) is a simple mark-up format for text, and Obsidian is a “knowledge base” (<https://obsidian.md>) for markdown files.

⁵<https://www.overleaf.com> – L^AT_EX editor

⁶<https://www.jabref.org> – bibliography management tool

Re-finding papers from multiple sources is tedious. Researchers have multiple strategies for finding papers, including keyword search (5/12), references from key papers(4/12) and recommendations from others (5/12). Similar to patent examiners, they often need to go back and re-read the papers. For example, P3 goes back to her papers and reads them again when she her notes lack sufficient context and she can no longer understand them. Others (P5, P12) developed strategies for manually adding a link to facilitate later backtracking. However, the process is tedious, especially when writing: *“Then when I need to write, I start from my note, and often go back to the article to see what is corresponding to what.”* (P11) Most researchers (10/12) track information at the document-level, unlike patent examiners who track it at the passage level. However, two researchers created more fine-grained notes: P9 recorded the page numbers of printed papers to make it easy to re-

Figure 4.11. Tables created by P2 and P5 for their literature reviews.

turn to a precise location, and P6 added comments under section titles that served as an anchor for future re-finding.

Reusing notes is difficult. All scientists (12/12) take structured personal notes as they read, which include both objective information, e.g., paper title, link, and quotations, as well as subjective information, e.g., personal summaries and comments. Although most participants (10/12) take notes that summarize the contribution(s) of the article, two (P5 and P8) also copy direct quotations.

All participants (12/12) use multiple applications for recording their notes. Most participants (10/12) comment on the articles directly, using *PDF Reader's* annotation feature, or write notes on physically printed copies (P9,P11). They also use a diverse set of additional software tools for recording personal notes, each of which imposes a corresponding file format. Some are open-ended, such as *Microsoft Word* (P1,P2,P4,P7,P10,P11), *TextEdit* (P8,P10), and *Emacs* (P6), or hand-written notes (P9) or hand-drawn tree graphs (P3). Others are designed specifically for taking notes, such as *Obsidian* (P5), *Zettlr* (P8) and *Evernote* (P12). The rest are highly structured as *Microsoft Excel* tables (P2,P3,P5,P12).

The format of the original notes directly affects the writing phase. Some (P2,P3,P5,P12) use tables to structure their notes. This provides an overview of the papers, making it easy to compare them (P12), and allowing them to discover “missing bricks in the literature” (P5). However, some uses of tables are problematic. For example, participants find it tedious to manually copy-paste information from multiple sources into each table, and maintain links to sources by hand. When writing, they must also manually copy-paste contents from the table. They also lack tools for interconnecting their data. For example, P8 wanted the text editor to automatically suggest relevant notes that he could cite. Fig. 4.11 shows how P2 uses *Excel* to categorize articles according to specified criteria. When she changed her categorization strategy, she decided to color code articles to reflect it, but found it

difficult to find relevant notes as she wrote her paper.

More flexible note-writing applications also led to frustration when participants needed to re-organize their notes or keep track of each article's quotations or provenance. For example, P2, P5 and P8 were frustrated by *Word's* lack of structure which P5 described as a "*sandbox*" that made it hard to find notes again.

In summary, reusing notes is difficult. Applications that avoid imposing structure during initial note taking make it difficult to find and reuse those notes later. Yet applications that impose an initial structure are only useful if that structure does not change during the sensemaking process.

Search results are easily lost. Given their diverse search approaches, researchers wanted to preserve traces of their search explorations. For example, P2 noted that she has no record of her history of searching for papers in the ACM Digital Library, which acts as a disincentive to further exploration. She wanted it to track "*what I had searched and what rabbit holes I had gone [down] and back*". P7 was frustrated that he could not remember where he found a particular paper, and cannot use it as a "*memory helper*" to start writing.

Summary. Together, studies 1 and 2 illustrate the challenges faced by two types of knowledge worker, patent examiners and scientists, as they try to read, search, annotate, organize, write and review text across multiple specialized applications, while manually tracking its provenance. Both groups develop personal strategies for organizing information, with comparison tables as the most common representation. A key difficulty they face is the lack of connection across the applications, which results in repetitive tasks, loss of information or both. The next section introduces *Passages*, a system we designed to address these issues.

4.7 Passages Concept

Based on these findings, we focused our design on facilitating the transfer of information across applications while tracking its provenance. Rather than creating a new, integrated application that supports the complex web of activities we observed, which would have created yet another information silo, we created a new type of interactive object that can be integrated into existing applications.

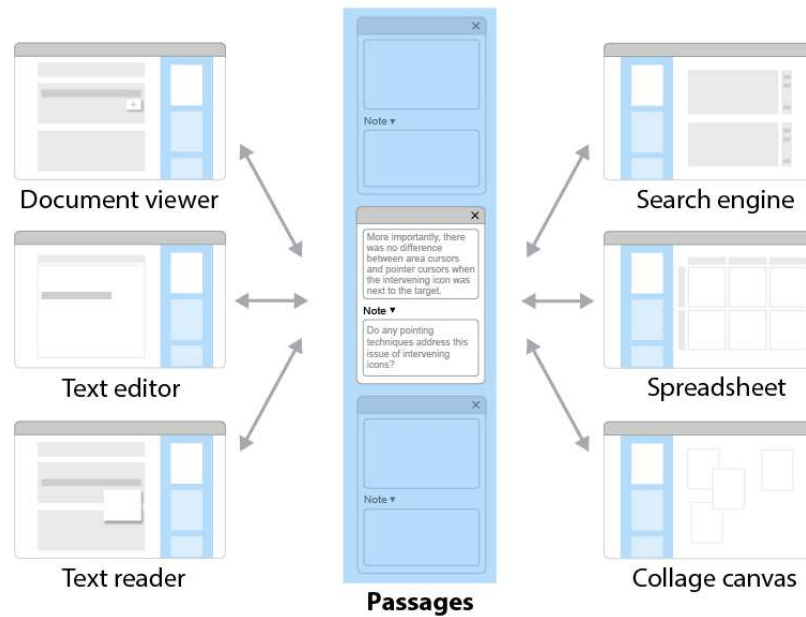


Figure 4.12. Passages reifies text selections into interactive objects that can be manipulated, reused and shared across applications

A *passage* is a snippet of text that includes metadata, such as its source document and location within that document, the time it was created, and user-defined tags and comments. A passage can be detached from its source document and reused in other documents and applications. Passages extend the concept of *textlets* in Chapter 3, which reify a text selection into a first-class object. However, while each textlet is bound to a particular document, passages can be shared **across** multiple applications (Fig. 4.12). To demonstrate the power of this concept, we created six prototype applications that each support a *Passages Side Panel* for collecting, annotating, manipulating, and sharing passages across applications, without losing their provenance.

4.8 The User Experience

We present a scenario⁷ to illustrate the user's experience of using *Passages*, with concrete examples derived from the two interview studies. Emma, a Ph.D. student in Human-Computer Interaction, works on improving pointing techniques. Her advisor, Alex, sends her three relevant papers to help her get started: *Silk Cursor* (Zhai et al., 1994), *Area Cursor* (Kabbash and Buxton, 1995) and *Enhanced Area Cursor* (Worden et al., 1997). Emma is tasked to compare them and propose new directions. As the process will involve various activities, she decides to use *Passages*.

⁷ See a video illustration: <https://youtu.be/aLC2GVVt10>

Reading

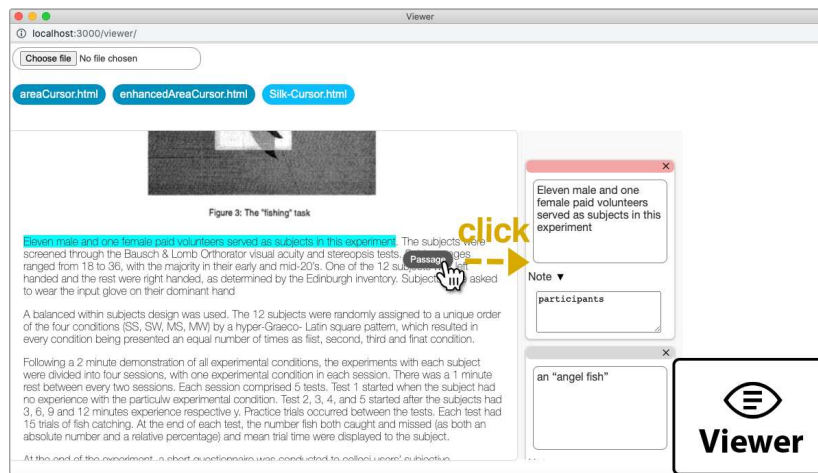


Figure 4.13. The Viewer lets Emma collect interesting text as passages, by selecting them and clicking the “Passage” button.

She opens the three papers in the Viewer (Fig. 4.13) and starts to read them, trying to find their similarities and differences. As she reads the *Area Cursor* paper, she realizes that it is inappropriate for fine positioning tasks, since selections may become ambiguous on cluttered displays. She finds this an interesting limitation and collects the corresponding text by selecting it and clicking the “Passage” button. The collected passage immediately appears in the side panel.

As she reads the *Enhanced Area Cursor* paper, she identifies several differences compared to the *Area Cursor*. For example, although the *Enhanced Area Cursor* switches dynamically between pointer and area cursor, performance become the same when an intervening icon appears next to the target. Emma collects this and other relevant text as passages, using the same interaction.

Organizing

As the number of passages grows on the side panel, she decides to organize them. She opens the Table (Fig. 4.19, Table) and drags and drop passages from the side panel into the table. She quickly tires of dragging the passages individually, and decides to move all the passages related to one paper into a column, all at once. She drags the title of the *Enhanced Area Cursor* paper and drops it into an empty column. All the passages automatically fill up the column and she names the column “Enhanced Area Cursor”.

Emma begins to see patterns in the table. For example, neither *Area Cursor* nor *Enhanced Area Cursor* improve performance when interven-

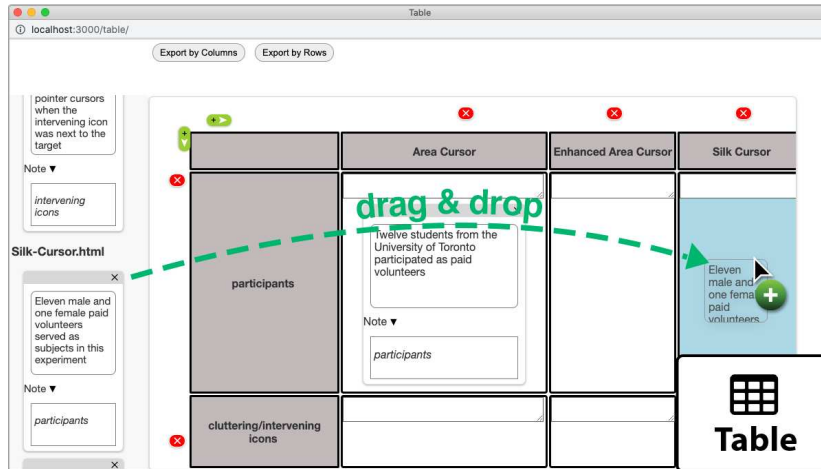


Figure 4.14. The Table lets Emma drag and drop collected passages and organize them into rows and columns

ing targets appear, and the *Silk Cursor* only works for 3D selection. She summarizes these patterns by naming the rows accordingly. Emma realizes that she has forgotten what one of the passages is about, so she double clicks it, which opens the original source document, with the relevant text selected and highlighted. This helps her examine the passage in its original context. Emma continues to build the table by moving freely among the passages.

Searching

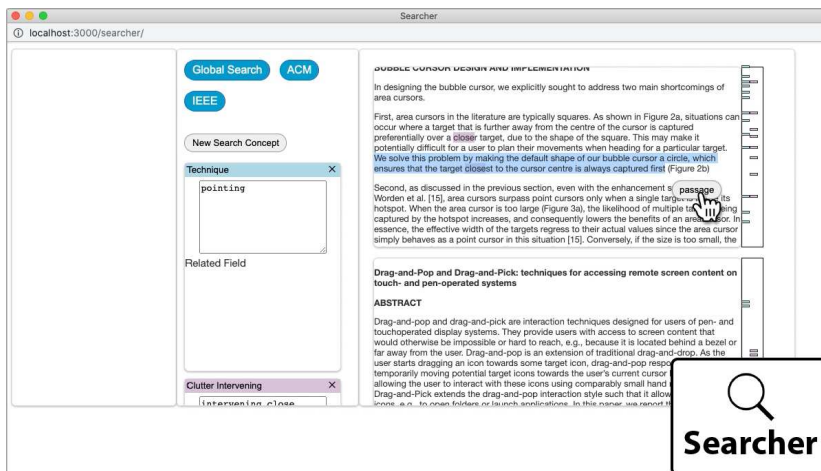


Figure 4.15. The Searcher lets Emma iteratively search for more documents by specifying multiple search terms, and keeping track of her search history.

Emma wonders if other papers address the problem of intervening icons next to the target. She returns to the Searcher application (Fig. 4.19, Searcher) and revisits her search history, where she discovers a new keyword: “intervening”. She creates a new keyword by selecting the text and clicking the “Create” button. She also adds “close” as a syn-

onym⁸ and launches a new search with these additional terms. The distribution of the different terms in the scrollbar of each resulting document helps Emma quickly locate the relevant text within the paper. After skimming the highlighted sections, she finds the Bubble Cursor paper (Grossman and Balakrishnan, 2005), which seems to address the issue of intervening icons. She creates a new passage with the relevant text description. She then decides to continue reading the document in the Viewer.

⁸ Normally, “close” is not a synonym for “intervening”. But in the context here, when multiple icons are close to one icon, usually these icons are intervening icons because it distracts users to acquire the icon she wants.

Writing

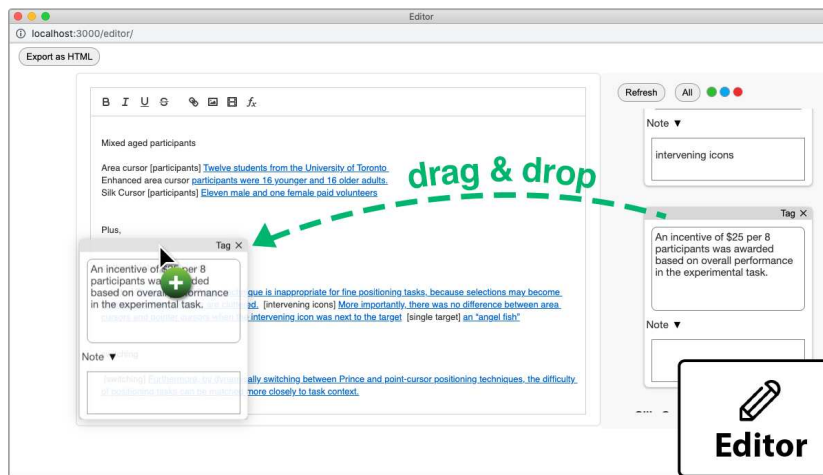


Figure 4.16. The Editor helps Emma communicate her findings, while preserving the provenance of each passage as she writes her report.

After filling the table with passages, and some additional searching and reading, Emma searches for pointing techniques that remain efficient in the presence of intervening targets. She then decides to write up her findings for Alex. She opens the Editor (Fig. 4.19, Editor) but does not want to start from a blank page. She returns to the table and uses the *Quick Drafting* feature: she clicks the “Export by Row” button (Fig. 4.17, (b)), which converts the row header and the passages (together with their notes) from this row into paragraphs in the Editor. Emma re-organizes the text into a concise evidence-based review, with a logical flow. While writing the last paragraph, Emma realizes that one piece of evidence is missing. However, she remembers that she had already read about it in the *Area Cursor Paper*. She opens the paper in the Viewer and immediately finds the related text. She collects it as a passage and drags it from Viewer and drops it into Editor’s side panel (Fig. 4.17, (a)). She then inserts it as a citation by dragging and dropping it into the text area of the Editor. Emma is satisfied with her review and exports it so she can send it to Alex.

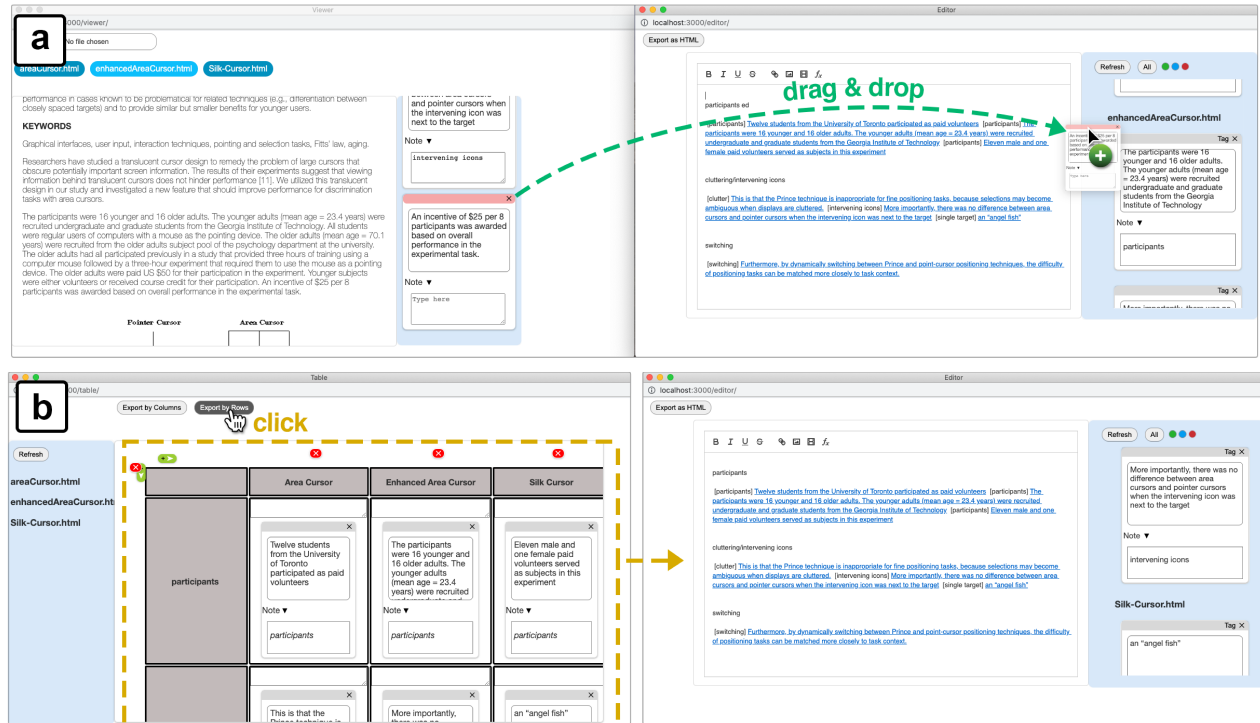


Figure 4.17. (a) Fluid transitions across applications through drag-and-drop of passages between windows; (b) Reuse of content and structure from the Table directly into the Editor

Communicating

Alex receives Emma's review and reads it with the Reader (Fig. 4.19, Reader). He wants to find out more about one passage cited as evidence. When he clicks on it, the original passage immediately appears as a yellow tooltip next to the text he is reading. He can also see the full context by double clicking on the passage tooltip, which opens the original paper at the correct scrolling position, with the relevant passage selected and highlighted. This helps him understand Emma's review. He pins two interesting passages into the side panel, as a reminder that he should discuss them further with Emma at their next meeting.

4.9 Passages User Interface

We describe the system and its implementation in greater detail. In order to demonstrate the concept of passage, we created a proof-of-concept implementation with six applications: Searcher, Viewer, Table, Canvas, Editor, and Reader that can share passages.

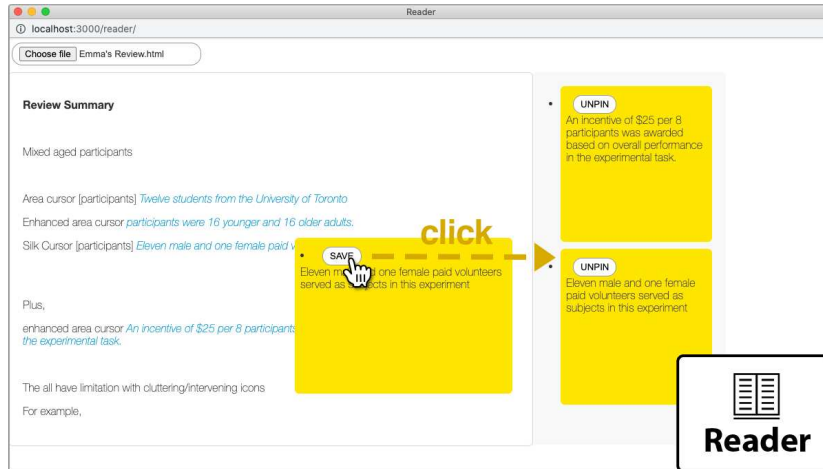


Figure 4.18. The Reader lets Alex read the report while still easily accessing the source documents to verify the claims

The *Passages* prototype is implemented as a series of web applications written in HTML, CSS and Javascript with the Vue.js framework. The back-end is implemented in Node.js and an NeDB database, except for the searcher application, which uses a PostgreSQL database for full-text search.

Passages and Passages Side Panels

Each passage is displayed as a box with the quote from the original document and a note containing text that can be edited by the user. It keeps track of both source document and its position of the text within the document, and can also be tagged with a color picked in a radial menu.

Users can always locate and re-select the text from the passage's source document simply by clicking on the passage. Since the text is re-selected instead of simply highlighted, users can immediately use normal copy-paste if needed.

A *Passages* side panel (or side panel for short) displays a collection of passages from one or more documents, automatically organized by document. Each of the applications we created includes a side panel, although passages can also exist independently of a side panel. The side panel provides an easy-to-understand, central location for holding the passages relevant to the user's current task. In addition to automatically classifying passages by their source documents, users can also filter the passages in a side panel according to their color tags.

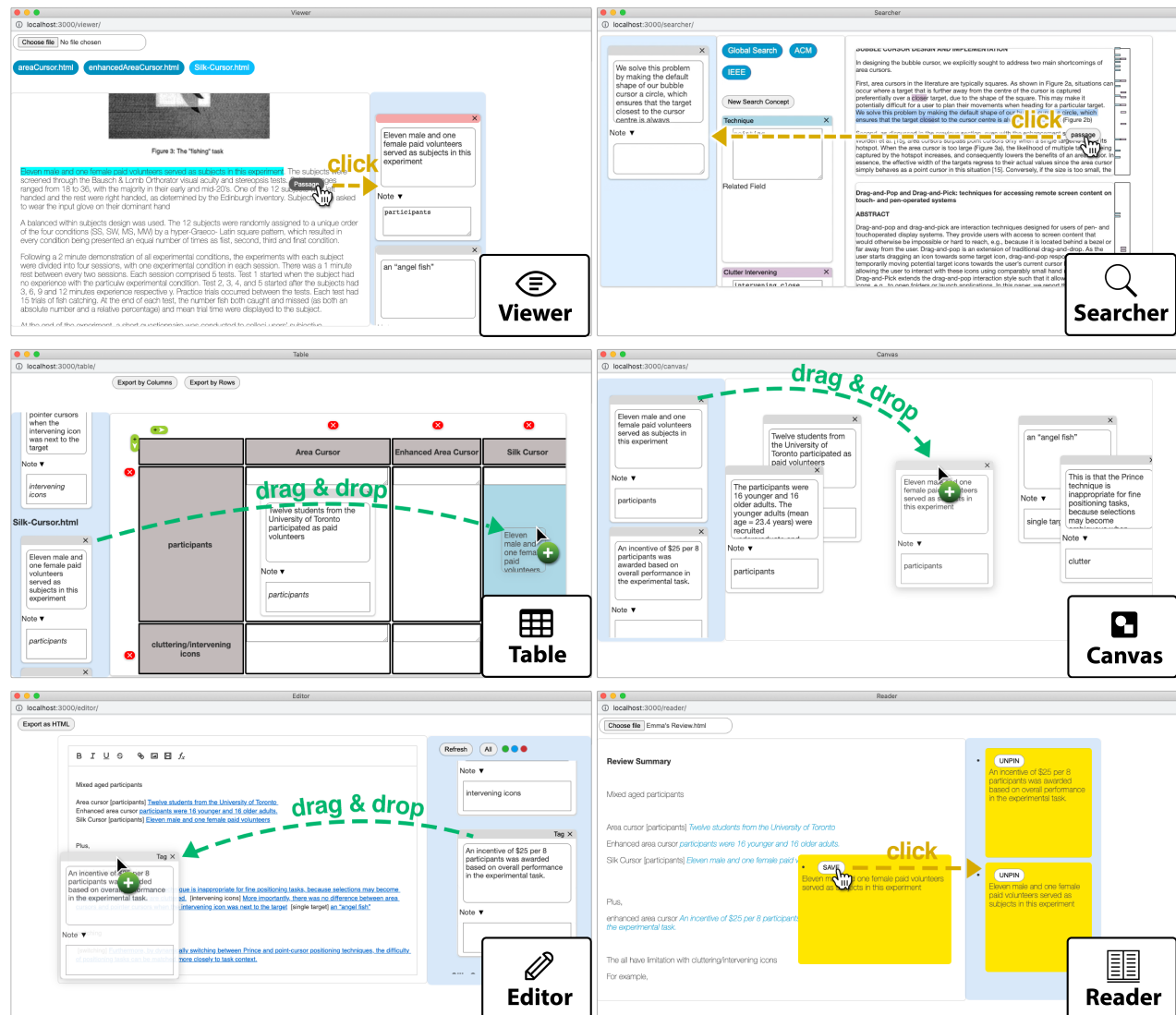


Figure 4.19. Passages overall user interface with six applications.

Viewer: Collecting Passages

The Viewer application (Fig. 4.19, Viewer) offers a lightweight way to collect passages across multiple documents. To create a passage, the user simply selects the desired text with cursor and clicks on the “Passage” button that pops up next to the cursor. The collected passage appears as a persistent interactive object in the side panel, with a small text area for personal notes. We intentionally did not provide a categorisation mechanism, such as tags, when creating passages in order to avoid the significant costs of eliciting structure early in the foraging process (Kittur et al., 2013). However, as mentioned earlier, passages can be tagged once created.

Table and Canvas: Organizing Passages

The Table and Canvas applications (Fig. 4.19, Table and Canvas) are designed to help users organize passages. The Table uses a grid structure, which is particularly useful for comparison tasks (Chang et al., 2020; Liu et al., 2019), whereas the Canvas offers a more free-form way to organize passages. The Table lets users drag-and-drop passages back and forth between the table and the side panel, as well as among table cells. A cell can contain more than one passage. If users drag a passage into a cell that already has a passage, the target cell will contain two passages. Users can populate a column with all the passages related to a specific source document simply by dragging the document title from the side panel to the column. Alternatively, users can click and hold a cell containing a passage to select it all those below it belonging to the same document, and then drag that set to a different cell.

We intentionally separated the collecting application (Viewer) from the organizing applications (Table and Canvas) for two reasons. First, this separation supports the “two-stage” model of sensemaking, where information is gathered first and then organized (Kittur et al., 2013). Second, this provides more flexibility and opens up the possibility of integrating *Passages* into other applications such as Mural ⁹, LiquidText ¹⁰ or Muse ¹¹, through a future API.

⁹ Mural: <https://www.mural.co/>

¹⁰ LiquidText: <https://www.liquidtext.net/>

¹¹ Muse: <https://museapp.com/>

Searcher: Searching with Passages

The Searcher application (Fig. 4.19, Searcher) combines the designs of TileBars (Hearst, 1995) and SearchLens (Chang et al., 2019), which let users create multiple keyword objects (Chang et al., 2019; Hearst, 1995) and visualize them in the scrollbar of each document to support rapid navigation (Hearst, 1995). Unlike SearchLens (Chang et al., 2019), the keyword object has two fields: an editable search query area and a “related field” (like a scrapbook). The search query area gives users complete control over their search queries, allowing them to apply advanced search, such as proximity search and add complex synonyms, as suggested by Studies 1 and 2.

The “related field” addresses another need identified in the interview studies, i.e. to learn about unfamiliar domains by searching for term definitions. Each keyword object offers a quick way to collect small pieces of information as passages, and attach them to the keyword using drag-and-drop. Users can capture the results of these informal searches and use them to grow their domain knowledge.

Editor: Writing with Passages

The Editor application (Fig. 4.19, Editor) lets users use drag-and-drop to embed any collected passage as a citation directly into main writing. Dropping a passage object inserts that passage's note as normal text, with the passage itself appearing as a blue citation. This interaction is based on our observations of knowledge workers who copy-paste their notes into their final writing for reuse. Hovering over the blue passage text scrolls the side panel and highlights the corresponding passage object. Users can edit the text as usual, and the link to the sidebar object is always preserved. Future work will include an additional button for selecting different citation reference styles, e.g. ACM or the American Psychological Association.

Reader: Communicating with Passages

The Reader application (Fig. 4.19, Reader) provides a quick and easy way to access any referenced documents. A single click on the blue text lets users see the exact underlying passage as a yellow tooltip. Users can then either open its source document to examine in greater detail, or pin it into the side panel for later reuse. Pinned passages can serve as reminders, e.g. for a discussion with the author. Maintaining a parallel link between the document and its reference information facilitates comparison and verification of the claims, which encourages better evidence-based practices.

We do not include the author's personal notes in the Reader because patent examiners in Study 1 expressed that seeing other people's notes might introduce bias in their own judgement. Future prototypes should offer the reader the possibility of either seeing or hiding their personal notes.

Summary

Table 4.1 summarizes the features of *Passages* that address the themes identified in the Study 1 and 2. Moving passages across applications lets users flexibly interleave knowledge work activities without being forced to follow a preset process. Moreover, the consistent representation of passages across the various applications supports a simple but powerful mental model for users.

Unlike previous systems where note-taking and writing are supported separately, *Passages* lets users reuse their notes as they write while maintaining their provenance. When exporting content from the Table to the Editor, *Passages* automatically collects the row or column head-

and to assess the overall value and relevance of the concept of *Passages*. We based the design walkthrough on a scenario that illustrates the use of *Passages* in a realistic context, with interaction snippets drawn directly from examples provided by patent examiners in Study 1. This scenario is explicitly designed to highlight challenges related to themes identified in Studies 1 and 2.

Participants: We recruited one current patent examiner, two internal software developers, two designers and one manager (two women, four men) from the same patent organization. The two designers were not patent examiners; whereas the other four all had current or previous experience as patent examiners. One developer had participated in Study 1 and was also the author of the automatic drafting application mentioned in Study 1.

Procedure: The workshop lasted one hour and featured three main sections: a 5-minute video demo, a 35-minute scenario-based design walkthrough and a 15-minute general discussion. Participants received a description of the design walkthrough procedure prior to the workshop, and a worksheet to fill out with their critiques and suggestions.

After first viewing a 5-minute video of the design scenario described in section 4.8, the first author presented or “walked through” a live demonstration of the scenario, pausing at each step to elicit critiques and suggestions from the participants. This scenario is organized into seven “interaction snippets” that illustrate how a patent examiner uses one or more key features of *Passages* to: 1) collect a passage; 2) organize passages in a table; 3) locate other passages; 4) use the table content for drafting a patent report; 5) include an additional passage; 6) insert a passage as citation; and 7) review the final report. At each step, participants discussed the pros and cons of each feature, and filled out a specially designed worksheet that recorded their critiques and suggestions for improvement. The workshop concluded with a general discussion of the strengths and weaknesses of *Passages*.

Data Collection: We video-recorded the session and collected participants’ notes about critiques and suggestions. We also took handwritten notes during the session.

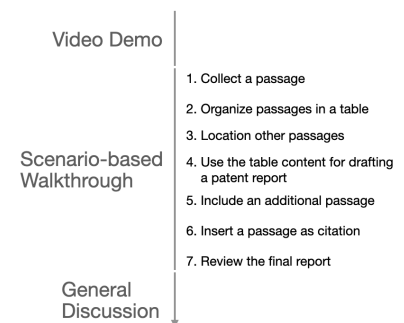


Figure 4.21: A timeline of the design walkthrough

4.11 Results and Discussion

All participants found *Passages* to be an interesting and elegant concept: *“a good way of improving the way we are working now.”* (P1). P4 said the concept of interacting with information snippets is fundamental to their work: *“Because we always deal with pieces of information. With this link, you can always navigate between the documents and to communicate etc. I think it is an elegant way of dealing with it. It is not rigid and you can always move it around. This is what I like.”* P1 and P3 also appreciated the power and flexibility offered by *Passages*. P1 said: *“The difference is really the flexibility, dragging things around, we have no way of doing that.”*

Two participants immediately wanted to integrate several features into their organization’s existing applications. For example, P2 wanted to integrate the Reader into their current reviewing applications: *“I like the connection that you can open the document. It is really good. I can imagine it working. In terms of integration, actually I could imagine this integrated into the current tools.”* P3 especially liked the concept of reusing snippets and the ability to always trace their provenance: *“It is a very interesting concept, especially this interaction of having the repository of snippets and just use them in the table and the communication. It saves a lot of time just by linking the documents and having the original version available and having an easy to compare several documents.”*

Two participants (P3, P4) appreciated the connection between reading and writing. P3 liked the ability to quickly access the reference information and the ability to quickly insert a reference in the text editor. P4 followed up saying that it is also the ability to have both information (source and reference) in parallel that makes it a great feature. P2 suggested combining reading and writing into a live thread, so that examiners could reply directly within the same passage.

Critiques. Although *Passages* lets users always go back to the source document by double clicking on it, one participant (P1) wanted more information about the source document when using a passage in the side panel, Table and Editor, so as to avoid making organization or citation mistakes. This suggests that users should have greater control over configuring the visibility of provenance information.

P1 also mentioned that the scalability of the Table may be an issue: *“I think I might get lost in a bigger table”*. The current feature of collapsible note in the passages could mitigate this concern. We are also considering supporting zooming in and out and coloring cells as in standard

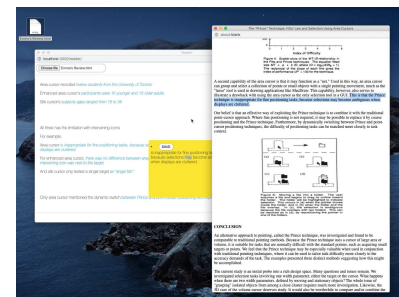


Figure 4.22: Having both source and reference side by side.

spreadsheet software.

Suggestions. Participants understood the role of the side panel as “*a repository of snippets and small fragments*” (P3) and found “*having them available all the time is really useful.*” (P6) They suggested additional *bidirectional* connections between the side panel and other applications. For example, P2 suggested that once the passage is included as citation, the side panel could indicate that it has already been used. P3 suggested a similar idea in the other direction: users could also see all the places in the editor where a particular passage has been inserted. Beyond the side panel, P2 suggested connecting the Table and Editor so that passages included as citations in the Editor would appear in the Table as well.

Two participants (P2, P3) immediately understood the power of reuse. They brainstormed about structuring mechanisms, such as tagging passages while collecting them. Examiners could then reuse this structure for other tasks, such as filtering and categorizing. P2 also suggested providing Table templates, either predefined or user-defined, with standard clauses in the Editor for examiners to reuse.

Other use cases. Participants suggested other use cases, such as coaching and quality control, where one examiner must review another examiners’ work. P4 also saw a connection with reviewing the literature: “*I will also generalize it not only to the review of patent, but also to any academic document, not just limited to patent world.*”

4.12 Study 4: Structured Observation with Scientists

In order to provide a grounded, qualitative assessment of the strengths and weaknesses of *Passages*, we also conducted a structured observation study (Garcia et al., 2014) with scientists. Participants first perform and then reflect upon a set of realistic literature review tasks, using both their usual literature review process and the *Passages* Viewer, Table and Editor applications. They then compare the details of each, both to identify problems and suggest new ideas. We also asked participants for feedback about *Passages*’s Canvas, Reader and Searcher applications.

Participants: We recruited 12 graduate-level human-computer interaction researchers (9 male, 3 female, ages 22 to 28), with an average of 3 years of research experience. They use a diverse set of applica-

tions for reviewing literature, including multiple PDF viewers, Google Docs (3/12), Microsoft Word (2/12), Overleaf (2/12), Obsidian (2/12), Google Sheets (1/12), Notion (1/12), and MarginNote (1/12).

Setup: Participants accessed *Passages* via a web application on their personal computers. We prepared two equivalent sets of HCI research papers about basic pointing techniques. Participants were already familiar with *pointing* as an HCI research domain, which is based on a fundamental model—Fitts’ Law.

The first set of three papers is based on the principle of increasing target width: *Area Cursor* (Kabbash and Buxton, 1995), *Enhanced Area Cursor* (Worden et al., 1997) and *Silk Cursor* (Zhai et al., 1994). The second set of three papers is based on the principle of reducing target distance: *Drag-and-pop* (Baudisch et al., 2003), *Pie menus* (Callahan et al., 1988), and *Object pointing* (Guiard et al., 2004). We use Balakrishnan (2004)’s cross-document comparison criteria, specifically participant profile, intervening targets, and commands for switching to normal interaction, to establish the “correct” criteria for analyzing each set of papers.

Procedure: Each session lasts approximately two hours. Participants are asked to read two groups of three research papers, once using their current set of software applications, and once using three *Passages* applications: Viewer, Table, and Editor. They then write a short review that compares the features of pointing techniques for each group of documents.

The two conditions, *document set*: increasing target width or reducing target distance, and *application choice*: with or without *Passages*, are counterbalanced for order within and across participants. Participants hear a scripted presentation that describes the functions of the Viewer, Table and Editor applications immediately before performing the *Passages* condition. We also show participants the Reader application, even though it is not necessary to perform the tasks, so they can see how a fictional colleague would read their report.

The tasks are presented in the form of the following scenario: *Your colleague wants to start a new research project with you. She sends you three documents about pointing techniques that she thinks are relevant but that she has not read. Your goal is to read and compare them. For example, what are the differences and similarities among these techniques? You will write a short paragraph at the end to tell your colleague about what you found out, so she can review and verify your findings. In the Passages condition, participants perform the task with the Viewer, Table and Editor applications.*

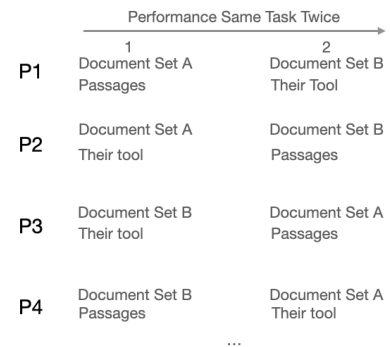


Figure 4.23: Document set and application choice are counterbalanced.

In the *non-Passages* condition, participants are free to use their own software. Participants use a think-aloud protocol (Fonteyn et al., 1993) to describe their experiences during each task. After each task, participants complete a short questionnaire and answer questions based on the experimenter’s observations of their behavior.

After completing both tasks, participants complete a final questionnaire that compares their usual literature review process with their experiences using *Passages*. Each question uses a five-point Likert scale, ranging from strongly disagree to strongly agree. Participants are asked to assess their performance on three tasks—making annotations, assessing similarities and differences between documents, and writing an evidence-based report—with respect to their mental load, perceived success, task difficulty and level of frustration. We conclude by asking participants to identify additional use cases from their recent projects; discuss the advantages and disadvantages of using *Passages*; and make suggestions for improvement. We also show participants the Canvas and Searcher applications and ask them to describe situations for which they might be useful.

Data Collection: We collected all the passages participants created with the Viewer, Table and Editor applications, as well as the questionnaire results. We took screen recordings as participants performed each task in both conditions, as well as video of the interviews and discussions. We also took hand-written notes.

4.13 Results and Discussion

We analyzed the criteria that each participant created for comparing the papers in each condition, and counted how many are correct, according to Balakrishnan (2004). The correct criteria for the *increasing target width* condition are: participant profile; transparent or translucent; and intervening targets. The correct criteria for the *reducing target distance* condition are: types of targets; intervening targets; and commands for switching back and forth.

All participants successfully completed the task in the *Passages* condition (Fig 4.24). Participants considered provenance tracking (8/12) and table exportation (8/12) as their favorite features. Provenance tracking “enables me to work more efficiently in terms of going back to the paper (P2)”; “is really good to be able to see the location of citation in the original document (P4)” ; and “saves me a lot of time finding the context of my previous notes

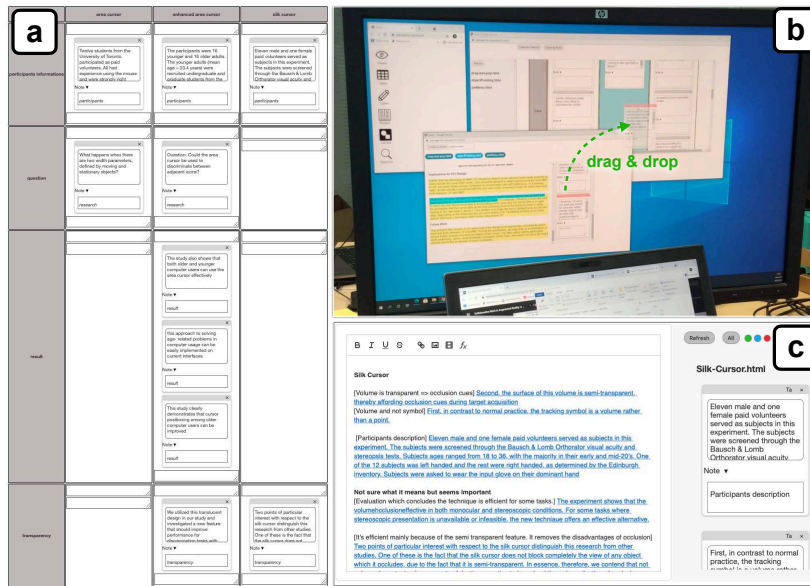


Figure 4.24. Interacting with *Passages*: Table (a) and report (c) created by participants. (b) P4 drag-dropped a newly created passage directly into the *Table*.

during writing (P11)". P2 liked Table exportation because it "enables me to compare what [I] have. I like that fact that [I] can just export it for my writing, not waste it. "

Participants appreciated the fluid combination of reading and writing: "It feels great because the [Reader is] not something that really exists [in Google Docs]...I really like the way we can click on...and trace the reference. (P1)" P3 compared *Passages* with the many note-taking apps she has tried, including LiquidText, MarginNote and Notion, and said none offers *Passages'* combination of reading and writing.

Participants immediately understood the concept of passages as generic objects that work across multiple applications: "I really like it keeping the notes as separate objects that are not explicitly linked to any PDF, making it a more flexible format (P10)." Other participants wanted to make passages work with their existing applications, such as Mural (P8) and email (P9).

Reuse Structure, Not Only Content

Participants reused passages collected in the Viewer in the Table (average reuse rate = 77.17%, SD = 0.37) and the Editor (average reuse rate = 88.36%, SD = 0.14). P9 said: "It feels great I can directly use the notes for organization and writing.". Only two participants did not use the table, since they usually write directly in a Google doc.

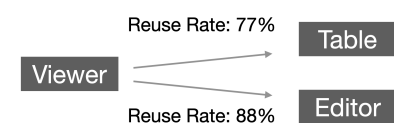


Figure 4.25: Participants largely reuse the passages collected.

Participants developed two major strategies for generating the structure for the report: task-based (5/12) and information-based (7/12). Those who used a task-based strategy took advantage of their previous experience with a similar task to generate the initial structure, e.g. pros and cons: *"Because the task is to compare interaction techniques, using pros and cons is very natural to me."* (P2). These participants collected passages as they read. By contrast, participants who used an information-based strategy first skimmed two or three papers without collecting any passages, to gain a general overview, since they did not *"know what to collect (P8)."* They then read each paper in greater detail and began collecting passages.

Despite the differences between these two strategies, all participants wanted to reuse both content and structure. P1 explained: *"For me, the important part of using a sheet is the structure."* Similar to the Study 3 examiner's suggestion to provide tags in the Viewer side panel, both P1 and P6 wanted to tag their passages in the Viewer and reuse that structure in the Table. Half of the participants (6/12) wanted to reuse structure from the Table in the Editor, either by exporting rows as separate sections or by including column titles as subsections.

Table Helps Discover New Insights

Participants were three times more likely to identify the correct comparison criteria when using *Passages* (Mean = 1.5, SD = 1, total = 18 times) compared to their own applications (Mean = 0.5, SD = 0.8, total = 6 times). Ten participants found at least one comparison criterion using *Passages*, compared to 4 participants when using their own applications. This suggests that *Passages* encourages participants to detect patterns across documents and generate new insights during comparison tasks.

Participants found the Table especially useful in discovering insights and keeping track of their sensemaking process. This includes three participants (P3, P8, P9) who had never previously used tables in their previous literature reviews. P7 said: *"The table may help me structure. The format really helps me make something apparent, by adding a table, you have two dimensions."* P6 said the table helped him to have the *"epiphany aha moment"* about the conceptual link across different domains. P9 said that, while he does not usually work with tables, he was happy he used it because it is very convenient to have all the information at a glance, and described it as *"conceptual model"*. Three participants who had used their own applications before trying *Passages* spontaneously created a table in that condition using a Markdown editor,

Google sheet or LibreOffice. This supports earlier findings that tables offer common and useful representation for analyzing sets of related documents.

Three participants perceived the Table as a temporary placeholder that helps organize their thoughts. P1 said: *“When I’m in a rush, I would completely focus on the writing instead of the sheet.”*. P3 said she had not previously used tables because she cannot reuse the content when writing, but that the export feature encouraged her to use the Table application.

We also found that several participants (5/12) returned to the Viewer to collect more passages, in order to fill in missing cells in the Table (See Fig. 4.24, b). P8 said: *“The Table feels like a basket and you are looking for eggs.”*, and explained that it serves as a description of what he is looking for and guides his search, giving him hints about which particular passages he needs to collect.

Composing and Rephrasing

Participants used a writing-oriented strategy when using their own applications, and a composition-oriented strategy when using *Passages*. In the former condition, participants write in the Editor while inspecting their notes or source documents. In the *Passages* condition, participants organise exported text directly into the Editor. Participants appreciated seeing all their notes in the same place, which limits window switching during copy-paste operations and reduces time wasted locating certain notes. P3 said: *“In Google Docs, I write mostly by myself. When I’m using Passages, I’m less likely to copy and paste contents, they are mostly from the original paper. It’s more convenient. [After importing] I would just change a little, make it more smoothly and logically [organized].”*

Participants had mixed opinions about exporting raw passages (See Fig. 4.26). Most (10/12) mentioned that they rephrase the text from the reference document in their own writing because they do not want to plagiarize. P10 said: *“I’m afraid it will be too similar to the original text if I just copy and paste. You don’t want to plagiarize stuff. Sometimes I quote but I just don’t want to do [it] all the time.”* One participant found the exported raw passage a bit *“distracting for thoughts and writing something coherent”*. Another participant wrote that pie menus have issues with large numbers of items even though the original passage states that *“Pie menus seem promising, but more experiments are needed before issuing a strong recommendation.”* As he built his table, he realized his mistake and said: *“facing the evidence forces me not to twist the words*

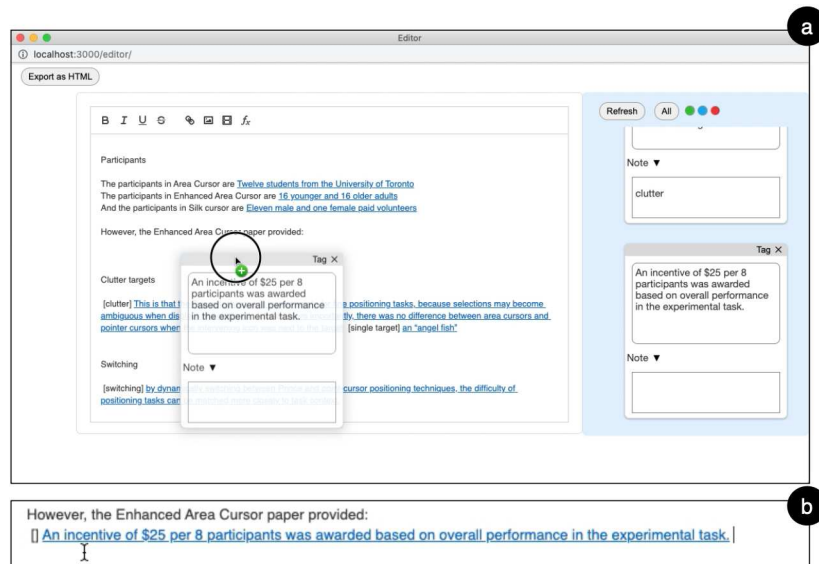


Figure 4.26. In current design, dropping a passage in the editor (a) inserts the whole raw passage (b).

and misquote.” This suggests that passages that appear in the Editor should be collapsible, to reduce clutter while still maintaining access to the original evidence.

Feedback for Searcher and Canvas

Participants quickly saw the benefits of the Searcher for their own work. For example, P6 had a problem retrieving definitions he had collected about machine learning in his papers, and found it really useful to attach definitions to keywords. P5 said he had to search his documents one by one to find papers related to “immersive technology” and “stereoscopic”, and wanted to use the Searcher to combine these two concepts, so he could search all his documents together. When shown the Canvas, three participants suggested to using it to manage themes in the thematic analysis of their qualitative studies.

Quantitative Results and Limitations

In the post-hoc questionnaire, participants ranked *Passages* as easier, less mentally demanding and less frustrating to use and more successful to accomplish the task than their existing applications. However, four participants also found it more frustrating to write the summary. Follow-up interviews showed that this was due to the Editor’s limited functionality compared to a commercial editor, e.g. lack of rich formatting and automatic formatting of lists. Participants also suggested features for improving how citations are inserted in the Editor, such

as inserting the document name and maintaining the link after copy-pasting text. Participants considered tagging in the Editor and auto-filling columns in the Table to be the two least useful features. Many participants suggested adding tagging to the Viewer, rather than the Editor, because they already have a structure in mind when writing.

Summary

Study 4 demonstrated that scientists can quickly learn to use *Passages* and take advantage of its flexibility. They largely reused the collected passages in terms of both content and structure and used the Table to discover more insights across documents. They also spontaneously devised compositing strategies by organizing exported passages into a concise writing and suggested other use cases, such as thematic analysis.

4.14 Conclusion

This chapter investigated how knowledge workers analyze and synthesis information by interacting with multiple documents. We interviewed two groups of “extreme” knowledge workers, patent examiners and scientists, and analyzed the complex interconnections among their document-related activities, especially active reading, searching, collecting, organizing, writing and reviewing. We identified six key difficulties they face, shown in Table 4.1. We found that these knowledge workers find it difficult to successfully manage their overall workflow given the set of specialized applications they use; and have trouble maintaining the provenance of the information they collect.

We then introduce the concept of a *passage*, an interactive object that reifies a selected snippet of text and maintains a link to its source document, allowing it to retain its provenance. Each passage can then be manipulated, reused, and shared across multiple applications using simple interactions such as drag-and-drop. We created a proof-of-concept collection of six applications, each with a *Passages* side panel that supports the creation and sharing of passages across them.

We conducted two studies to assess *Passages*: a design walkthrough with six participants from a major patent organization; and a structured observation study with 12 scientists. Participants found *Passages* to be both elegant and powerful, and especially appreciated its flexibility and their ability to track the provenance of individual passages.

They felt it would improve their work practices, facilitate reuse, and offer them novel strategies for analyzing and composing documents.

What have we learned about representation and manipulation? One of the design principle in the Star user interface is universal commands that can be used throughout the system, such as move, copy and delete (Smith et al., 1982). These universal commands *"strip away the extraneous application-specific semantics to get at the underlying principles"* (Smith et al., 1982). This is also the essential design concept for Passages. What I found in the interview study is that patent examiners need to deal with snippets of information across applications. This has been confirmed by one director in the patent organization: *"Because we always deal with pieces of information."* This observation motivates the design of a new representation based on information snippets that users can flexibly move across applications, which I call *"universal objects"*. These universal objects represent the users' continuous object of interest in multiple activities. The representation of these universal objects is consistent across multiple applications but the behavior is adapted to each specific context. For example, dragging a passage into a Table moves a passage from the side panel to the cell of the table. But dragging a passage into the text editor inserts a quote as linked editable text. The evaluative study has shown that users can quickly learn this context-sensitive behavior.

In summary, many current computer systems support a few universal commands since they are primarily designed based on the Star user interface. In this chapter, I expand our understanding of representation and manipulation by introducing the idea of *"universal objects"* that maintain a consistent representation but also adapt to specific contexts in different applications. I argue that universal objects simplify the user's conceptual model and improve the flexibility of the manipulation.

5

Representation for Document Management

The work reported in this chapter was joint with Julien Gori, a post-doc in the same research team. I conducted the initial interviews and worked with Julien Gori to create the interface concept. Julien Gori and Michel Beaudouin-Lafon implemented the prototype and I conducted the evaluation analysis. The work was published at ACM UIST2020 (Gori et al., 2020).

We have now seen how knowledge workers edit one document and analyze a set of related documents. To complete this thesis work about representations for digital documents, I wanted to investigate one last fundamental aspect: document management.

In this chapter, I want to explore how knowledge workers manage their documents in an age where heterogeneous data science workflows become the norm. My interviews with scientists show that instead of relying on more self-contained environments such as Jupyter Notebooks, scientists work across many diverse tools. But they have difficulties using the file system to keep track of, re-find and maintain consistency among related but distributed information. This led us to create FileWeaver, a system that automatically detects dependencies among files without explicit user action, tracks their history, and lets users interact directly with the graphs representing these dependencies and version history. We show that by making dependencies among files explicit and visible, FileWeaver facilitates the automation of workflows by scientists and other users who rely on the file system to manage their data.



Figure 5.1: Manage multiple documents.

5.1 Context

Chapter 4 showed that many knowledge workers, such as scientists, must use several specialized tools that are not designed to dialog with each other. Prior work has also shown similar findings for data analysis activity. For example, Oleksik et al. (2012) describes how researchers use a mix of standard office productivity tools and specialized tools for experimental protocols and data analysis, while Zhang et al. (2020) list up to 13 different categories of tools used by data scientists.

These specialized tools typically load and save information in proprietary and/or binary data formats, such as Matlab¹ .mat files or SPSS² .sav files. Knowledge workers have to rely on standardized exchange file formats and file format converters to communicate information from one application to the other, leading to a multiplication of files.

¹ <https://mathworks.com/products/matlab.html>

² <https://www.ibm.com/analytics/spss-statistics-software>

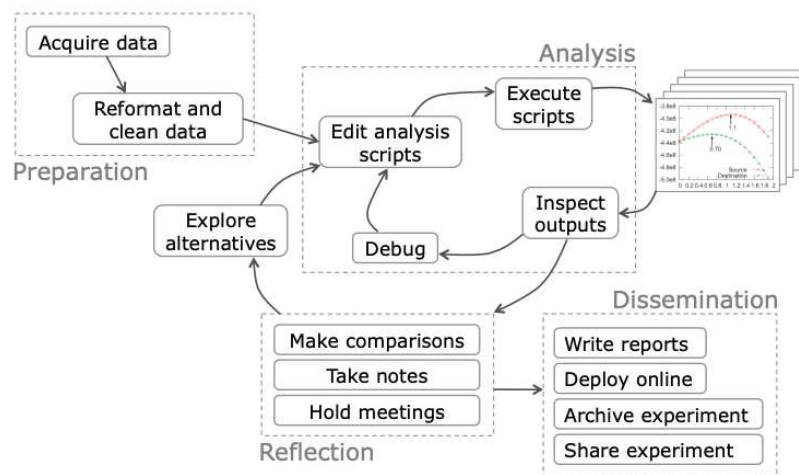


Figure 5.2. Guo (2012)'s typical data science workflow

Moreover, as exemplified by Guo's (2012) "typical" workflow of a data scientist (Fig. 5.2), knowledge workers' data analysis practices often consist of several iterations of exploratory, production and dissemination phases, in which workers create copies of files to save their work, file revisions, e.g. to revise the logic of their code, and file variants, e.g. to modify parameter values.

However, neither the file system nor file navigation tools are designed to track the *relationships* between files nor the *histories* of files, offering little support to knowledge workers for managing the numerous files and associated workflows that their work requires. Software designed to capture file *provenance* (Muniswamy-Reddy et al., 2006; Murta et al.,

2014) addresses a similar issue by capturing metadata describing the origin of the files as well as all the actions that are performed on it and the actors who perform them. However, we do not know if this fine-grained approach is adapted to the needs of knowledge workers. In this chapter, we are thus interested in investigating how knowledge workers manage their documents and how we can support the production and dissemination of the knowledge.

5.2 Related Work

Three main areas are related to document management while using many specialized tools: the issues related to *information silos*, i.e. the fact that digital information tends to be trapped in proprietary files formats and closed systems; the practices and workflows used by scientists in their data analysis tasks; and the studies and tools addressing the management of file relationships.

Information Silos

Previous work has identified the difficulties that users encounter with the rigid hierarchical organization of current file systems (Bondarenko and Janssen, 2005; Dourish et al., 1999; Ravasio et al., 2004) and the information fragmentation created by *information silos* (Karger and Jones, 2006; Ravasio et al., 2004). In a study of users' desktops, Ravasio et al. (2004) reported the interviewees' frustration with the fragmentation of information across files and most interviewees expressed the need to have their information linked together. The Placeless system (Dourish et al., 1999) partially addressed these issues by letting users create and manage their own document properties. Karger and Jones (2006) proposed three approaches to address information fragmentation: grouping (including tagging), annotating, and linking; and showed that linking related information can support users orienteering behavior (Teevan et al., 2004).

Research on scientists' work practices further highlights these issues. Oleksik et al. (2012) studied the artifact ecology of a research center and found that scientists used multiple computing applications to create information artifacts that are locked into applications, making it difficult to reuse content and get a unified view of the related research material. Their follow-up study (Oleksik et al., 2013) revealed the need to support three types of links: inheritance (source document), spatial proximity (spatial layout), and explicit links (resources and notes).



Figure 5.3: Information locked in its applications.

Tabard et al. (2008) (Fig. 5.4) studied lab notebooks and found that biologists work with a complex web of interrelated references in both the physical and digital worlds that they keep in their head, and that they struggle to map this structure into a single organizational form.

Data Analysis Tasks and Workflows

Data analysis and results sharing is an essential part of scientific discovery (Oleksik et al., 2012). The iterative and exploratory nature of data analysis (Guo, 2012) tends to produce numerous manually versioned scripts and output files (Guo and Seltzer, 2012). Computational notebooks combine code, visualizations and text, bringing fragmented but related information into a single document (Rule et al., 2018b). Scientists are rapidly adopting this new medium to document and share exploratory data analyses (Shen, 2014). Researchers have built tools for computational notebooks to support various activities such as informal and rapid version control (Kery et al., 2017), re-finding of past analysis choices (Kery et al., 2019), easy navigation (Rule et al., 2018a) and managing messes (Head et al., 2019).

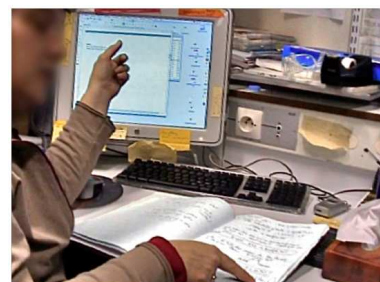


Figure 5.4: Tabard et al. (2008)'s research shows how biologists juggles complex mix of paper and computer-based information.

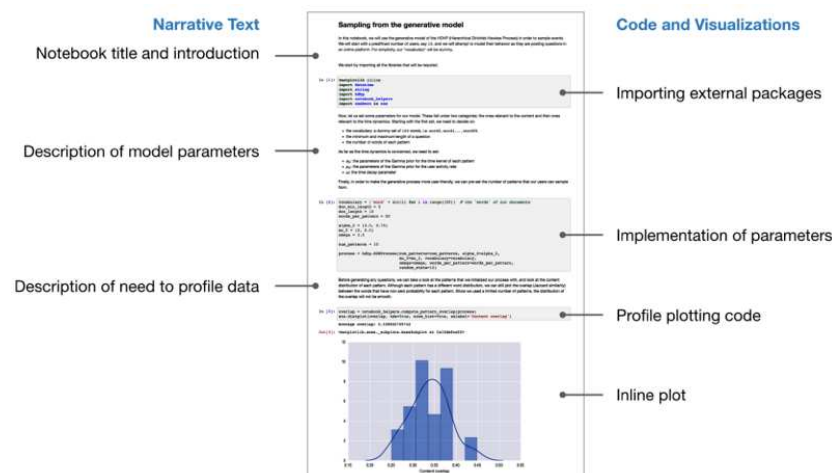


Figure 5.5. A Jupyter notebook combines code, text and visualization in a single notebook format. Image taken from (Rule et al., 2018b).

However, building these tools within computational notebooks does not fully solve the problem: studies show that users of computational notebooks need to access diverse tools (Zhang et al., 2020) and combine them with other tools and documents (Chattopadhyay et al., 2020; Rule et al., 2018b). A survey (Zhang et al., 2020) of data scientists' collaborative practices reported 13 high-level categories of tools, including code editors (e.g. Visual Studio Code), computational notebooks (e.g. Jupyter Notebook), spreadsheets, data analysis tools (e.g. SPSS), document editing tools (e.g. LaTeX), and presentation software (e.g. Microsoft PowerPoint). Rule et al. (2018b) found that "*individual note-*

books rarely tell a story by themselves but are routinely combined with other notebooks, emails, slides, and ReadMe". Furthermore, their interviewees "even transferred outputs of the analysis to an entirely different medium (e.g. slides, word processing document) for easier review."

These studies suggest that scientists use a variety of tools in their data analysis and sharing process. We are interested in how they track the inter-connections between different documents in this process and why they choose to create multiple documents in the first place.

File Relationships

Previous research has explored file relationships to devise new ways to organize digital content, including using provenance (See more in section 4.2) and linking (Tabard et al., 2007).

Provenance Karlson et al. (2011) (Fig. 5.6) described a system that tracks copy relationships among files, helping users to manage their versions. They introduced the 'versionset', a set of files that represent a user's concept of a document and found that file format is an important element of the user's conceptual model. Lindley et al. (2018) developed the 'file biography' to encompass the provenance of a file and its propagation. Commercial user interfaces for version control systems such as Sourcetree (Fig. 5.7) represent file revisions in a time-based tree visualization³, but these tools are designed primarily for coordinating work among programmers. Prior studies have found that scientists rarely use version control systems (Kery et al., 2017; Perez De Rosso and Jackson, 2013) and that knowledge workers mostly use manual versioning (Chapter 3 and (Karlson et al., 2011)) because they find version control systems too complex. We find two limitations to these systems: They require users to explicitly specify relationships among files, and the relationships are presented as non-interactive graphs. We are therefore interested in improving both automation to infer file dependencies and direct interaction to provide more user control.

Linking Tabard et al. (2007) take a different approach to support web navigation by automatically linking web pages together according to user actions. Software development tools such as webpack⁴ generate a dependency graph by analyzing source code, but do not visualize nor let users control the resulting graph.

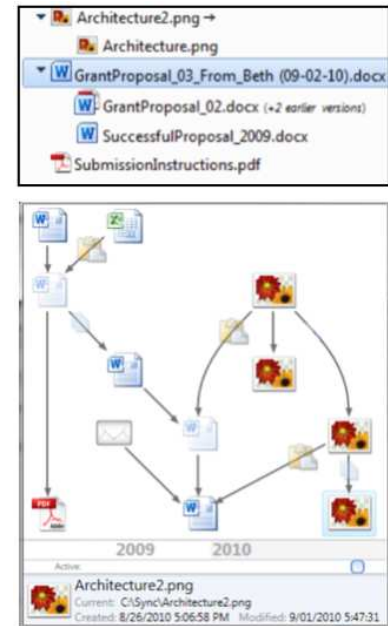


Figure 5.6: Karlson et al. (2011)'s versionset: folder view with indentation (top) and graph view showing file relationship (bottom)

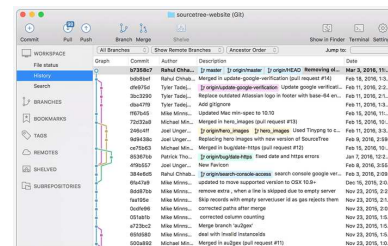


Figure 5.7: Sourcetree's time-line interface for version control. Source: <https://www.sourcetreeapp.com/>

³ See a list of GUIs at <https://git-scm.com/downloads/guis/>.

⁴ <https://webpack.js.org/>

5.3 Interviews with Scientists

Our first step was to understand scientists' document management practices. We conducted semi-structured interviews in their workplaces to examine their document artifacts and the way they manage them.

Participants: We interviewed 23 scientists (7 women, 16 men): six professors, four researchers, four Ph.D. students, five research engineers, three post-docs, and one research associate. The research topics include neuroscience, mathematics, chemistry, physics, biology and computer science.

Procedure: All interviews were conducted in English by the second author and lasted 45-60 minutes each. We visited the participants in their labs and asked them to show us the tools, e.g. notebooks, that they use to keep track of their research information. Based on the artifacts they showed us, we asked them to describe a recent, memorable event, either positive or negative, related to using that artifact. We then asked about their work practices based on these events and artifacts.

Data Collection: All interviews were audio recorded and transcribed. We also took photos and hand-written notes.

Data Analysis: We analyzed the interviews using reflexive thematic analysis (Braun and Clarke, 2019). We generated codes and themes both inductively (bottom-up) and deductively (top-down), looking for breakdowns, workarounds and user innovations. Two coders (including the interviewer) conducted the analysis. They grouped codes into larger categories, discussed any disagreements and rechecked the interview transcripts to reach a shared understanding. They arrived at the final themes after two iterations.

5.4 Results and Discussion

We generated six primary themes: taking advantage of the characteristics of artifacts; transferring information across artifacts; keeping information "just in case"; finding and re-finding; iterating the produce-communicate cycle; and expressing file relationships.

Taking Advantage of the Characteristics of Artifacts

All participants take advantage of the characteristics of different types of information artifacts to manage their information (Fig. 5.8). For example, they use paper as a thinking and reasoning tool because free-hand drawing enables rich representations, flexible layout and better memorization. This is consistent with prior studies on the advantages of paper (Mackay, 1999; O'Hara and Sellen, 1997; Sellen and Harper, 1997, 2003).

Most participants (20/23) use at least one blackboard or whiteboard in their office or in meeting rooms. These boards offer a large working space for free-hand sketching and writing, supporting fluid collocated collaboration (Pedersen et al., 1993; Tang et al., 2009).

Five participants with experimental science training also keep a physical lab notebook. This notebook follows a chronological order and makes it easy to switch between structured and free-form information. Participants create links to keep related information in context by noting down file names or sticking print-outs on the relevant pages. Participants also have at least one personal notebook for “purely temporal information” (P15). These personal notebooks are less clean and structured than the “official” lab notebook. Participants transfer the important information to the lab notebook.

Similar to the analysis of biologists’ work practice (Tabard et al., 2008), these results suggest that the nature of information being distributed among different artifacts such as paper, whiteboard and lab notebook, is driven by their unique characteristics.

Transferring Information Across Artifacts

From a first raw idea on paper to a well-written article, participants continuously structure their ideas over time as a project unfolds. We found that the process of transferring information from one artifact to another is a necessary cognitive process where participants review, rethink and re-evaluate their ideas.

P9 described how he worked with a colleague on an idea by writing on the blackboard; copying it onto paper in a cleaner version; going back and forth between blackboard and paper several times until it was mature enough to write it in LaTeX. He explained: “When I copy, I am not just copying. I try to understand it of course... We also fix errors when we transfer”.



Figure 5.8: Scientists take advantages of different artifacts such as paper, whiteboard, lab notebook and Jupyter notebook

Seven participants reported taking whiteboard photos during meetings to persist information digitally (Fig. 5.9, top). While taking a photo is quick, they often forget to do it (P9, 10) and find it cumbersome to transfer the photos to their laptop (P2), organize them (P2, 9, 21) and understand the messy content of the whiteboard after the fact (P22). As a result they prefer transcribing information onto a different artifact, such as their notebook.

Keeping Information “Just in case”

A quarter of the scientists (P1, 6, 9, 10, 14, 21) tend to hoard information (whiteboard photos, figures, versions of paper) “just in case” even though they rarely revisit it (P9, 10, 14), partially because they are not able to anticipate what will be needed in the future (Fig. 5.9). For example, P21 (computer scientist) said: *“It is really insane that amount of photos we have. We clearly do not use all of them.”*

Hoarding information adds to the challenge of future re-finding. Participants find it difficult to determine the right version among their multiple devices: *“Anyways, everything is mixed up (between home and work computers).”* (P11). Participants use time stamps and modification events as primary cues to determine the right version. While prior research has focused on supporting document versions (Karlson et al., 2011), we also found the need to support managing the versions and variants of graphs, particularly in scientists’ data analyses. For example, P6 (bioinformatician) prints several versions of the same graphs as he iterates analyses and puts the latest version in a plastic folder to easily identify it.

Finding and Re-finding

All participants develop various strategies to support their future re-finding tasks.

Summarizing Participants summarize information to avoid having to go back to the raw source. The summarized information is usually in another artifact that serves as a reference when re-finding. For example, P19 (experimental chemist) summarizes everything about a particular molecule in a physical album when she completes a series of experiments and has a clear result (Fig. 5.10, top).

Linking related information Five experimental scientists link related information together in their master lab notebook, like in (Tabard et al., 2008). If they need to link digital content, they simply note down the

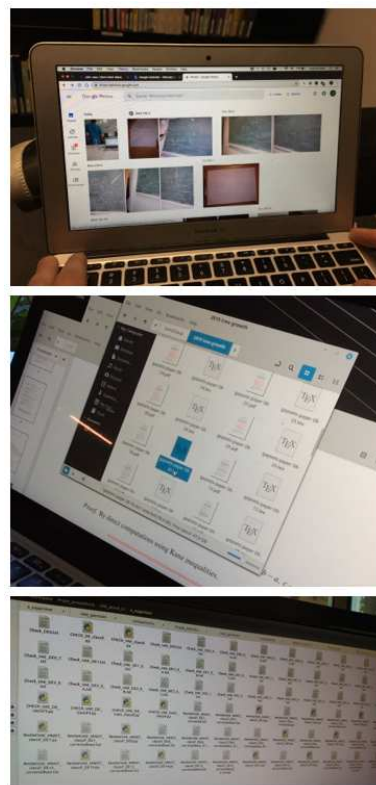


Figure 5.9: Hoard information such as whiteboard photos, paper and script versions.

file names. For example, P15 (computational chemist) writes down all the information related to a plot in his paper lab notebook, including location of the data, scripts to generate that plot, location of the Jupyter notebook of that plot and, name and location of the image file (Fig. 5.10, middle).

However, these manual links with file names break when participants change the name or location of the file, as illustrated by P16 (experimental physicist): *“I have done a lot of cleaning of my files so I don’t know if I can find it.”*

Saving data in a single place Eight data scientists use a project-based organization to save related information in a single folder, similar to (Tabard et al., 2008). But their tendency to hoard information and the messy results produced by exploratory programming (Rule et al., 2018b) result in folder clutter. For example, P18 (machine learning researcher) saves the data, the analysis scripts and the output in a single folder. As he produces variations and versions of the scripts and corresponding outputs, files accumulate, hindering re-finding (Fig. 5.9, bottom).

Several participants use computational notebooks, e.g. Jupyter notebooks, to keep track of their analysis process by saving scripts and their outputs in a single place. However, as Rules et al. (Rule et al., 2018b), we found that these notebooks are routinely combined with other documents such as slides (Fig. 5.10, bottom) and ReadMe.

This suggests that the strategy of saving data in a single place might not be flexible enough for scientists’ knowledge management. They also need a lightweight way of linking related but distributed content together.

Preserving importance The transition from one artifact to another is also a process where participants assess the importance of information. They usually preserve important information in a new artifact in a more structured way and discard information they do not need anymore. For example, P10 (mathematician) first works on paper, then selects the ideas he thinks are worth preserving and transfers them onto his paper notebook.

This process also applies to data analysis. P14 (bioinformatician) performs exploratory data analysis in one Jupyter notebook, where he produces many graphs and statistics. Once he is happy with the results, he copy-pastes them into his main Jupyter notebook, where he

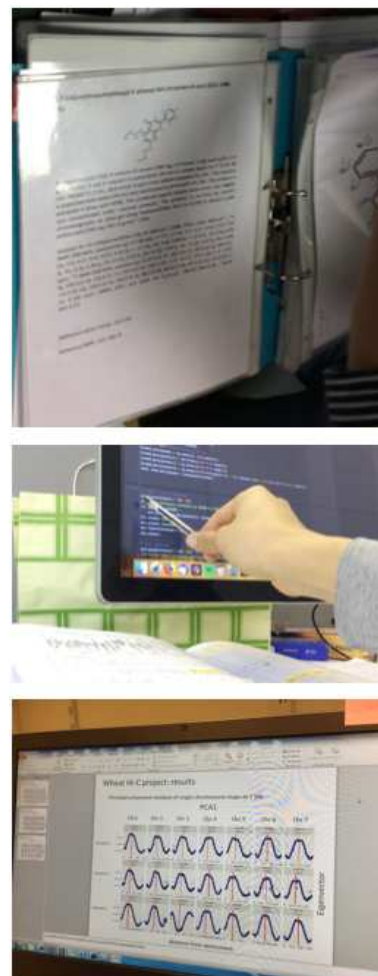


Figure 5.10: Various strategies for re-finding, e.g. summarizing, linking and combined with other documents.

collects all the important results.

Iterating the Produce-Communicate-Reproduce Cycle

Communicating results often involves manually cleaning, selecting and inputting content into another type of document or software tool (Fig. 5.11). Note that the results are often in the form of plots and graphs. Participants use a range of media for sharing their results including Microsoft PowerPoint (P22, 6, 16) and Word (P22), Printout (P6, 13), LaTeX (P13), Google doc (P21), HTML (P15, 2), Markdown (P18) and Jupyter notebook (P14, 22). This supports the finding that data analysts transfer outputs of their analyses to an entirely different medium, e.g. slides or word processing document, for easier review (Rule et al., 2018b).

Participants' choice of medium is related to the audience they target. For example, P13 prefers to give his supervisor a paper printout because she is more likely to read it and get back to him. P2 (bioinformatician) prefers to share a static HTML file instead of an interactive Jupyter notebook because *"what we have done in the project is that every group has its own expertise. If they want to change the parameters, they will ask you to change it. They won't do it directly."*

Some participants (P6, 22) complain that it is time-consuming to tediously copy-paste content into slides for easy sharing: *"The problem is that you have to drag and drop into PowerPoint. So when you have 15 images, it is a bit time consuming."* (P6)

Scientists often face re-finding challenges when they need to modify content after getting feedback. For example, P18 had to recreate a graph after discussing with coauthors but could not find the data for that graph: *"I did so many experiments that I could have accidentally overwritten the file..."*

After a modification, the new results need to be communicated again. Scientists need to manually re-execute the update pipeline by cleaning, selecting and putting content into another medium. P14 (bioinformatician) said: *"There is obviously a problem of synchronisation. At some point, I generate the new version of a figure with slightly different parameters. I need to reload it in Overleaf. It happens often."* Managing different file formats or variants adds more overhead, as P11 illustrated: *"Including external files in LaTeX can get messy very very quickly. What? I want to use jpeg; I am pretty sure this thing is jpeg. But the browser is suggesting that it is not a jpeg."*

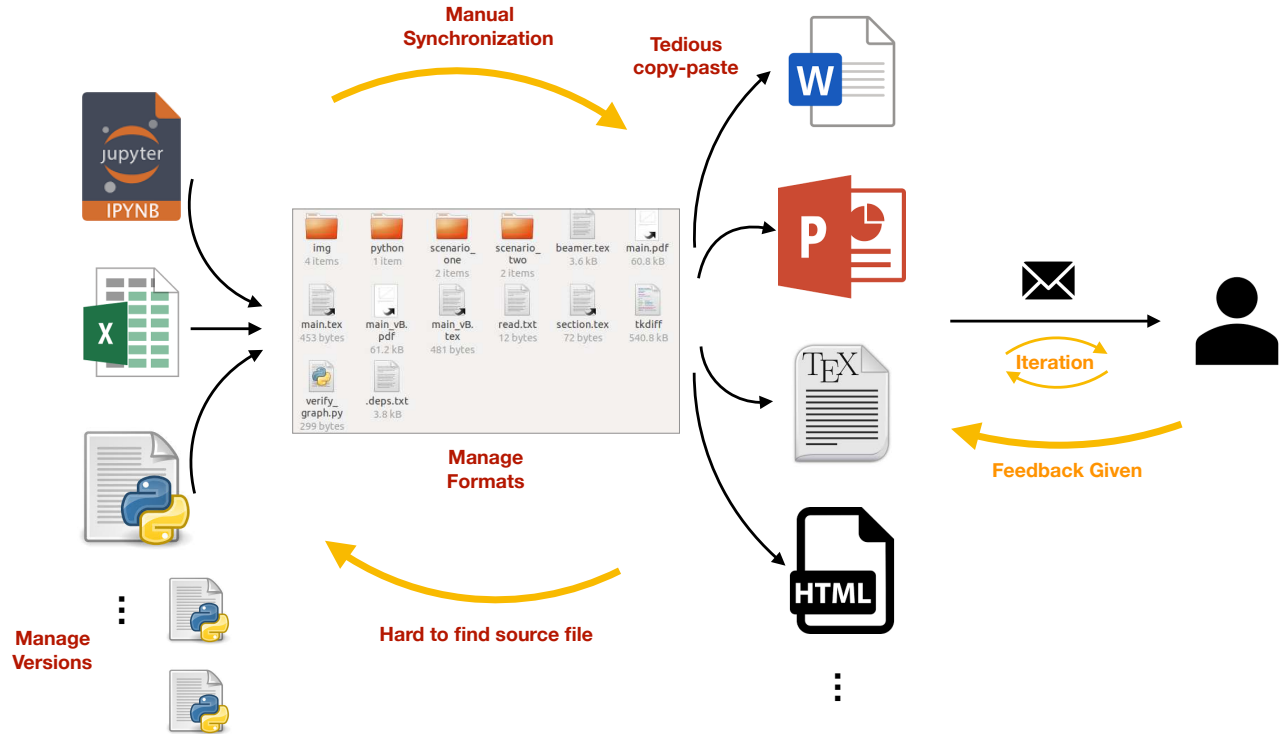


Figure 5.11. Rapid lifecycle of various documents in scientists' workflow.

In summary, the produce-communicate-reproduce cycle requires scientists to successfully re-find the original content, reproduce the results with new settings while keeping the previous versions and manually synchronizing the updated changes to another medium to communicate these results again.

Expressing Inter-File Relationships

We identified three types of inter-file relationships that scientists want to express: description, dependency and coexistence.

Description To help re-find and re-understand in the future, participants often use text files to add explanations to other files, e.g. tables, images, pictures and code. Participants use ReadMe files (P14), tables of contents in notebooks (P15) and code comments (P6). For example, P21 (computer scientist) writes plain text files for each image in a folder to help re-find them.

Re-finding code is particularly challenging because it requires understanding the code again. P14 (bioinformatician) finds computational notebook useful but still writes additional ReadMe files for each Jupyter

notebook to explain the analysis to his future self (Rule et al., 2018b). Other participants complain that out-of-order execution and unclear cell dependencies hinders the reproducibility of previous results. For example, P15 does not use a Jupyter notebook as a full-fledged lab notebook because he can accidentally re-run a code cell that overwrites the figure he wants to save. He still uses his paper notebook.

Dependency Participants need to keep track of files that are generated by other files and manually keep them in sync. This often happens in produce-communicate-reproduce cycle when scientists need to manually modify and remix content across applications.

Coexistence All eight data scientists put the data and analysis scripts (including Jupyter notebook) in sub-folders of the same folder to link the data with the corresponding analyses. P14 (bioinformatician) and his student created two separate but linked Jupyter notebooks because they use two languages (R and Python), that Jupyter does not support simultaneously.

Participants rely on file names to express inter-file relationships. However, links with file names are easy to break in case of renaming. File names are also used for other information such as experiment parameters, manual version suffix and file format, resulting in long, hard-to-understand file names.

Issue	Empirical Evidence
Hard to re-find related but distributed information	Related information is often distributed by necessity because scientists take advantage of the characteristics of different information artifacts (both physical and digital), resulting in difficult information re-finding.
Lack of tools to track inter-file relationships	Among various strategies to support future re-finding, linking is effective but limited because participants want to create different types of relationships among files and using file naming conventions is error-prone.
Manual synchronization	To share results, participants manually re-execute the pipelines that create the target documents.
Difficult version management	Participants' hoarding behavior and use of multiple devices adds to the challenges of re-finding and versioning. They have difficulty keeping track of versions of their documents, such as graphs.
Lack of support to manage file variants, e.g. format	Participants produce variants of a file for different purposes. Different variants of the same file clutter the folder, making it hard to manage.
Tedious creation of shared document	Participants complain that it is time-consuming and tedious to re-create shared document for review.

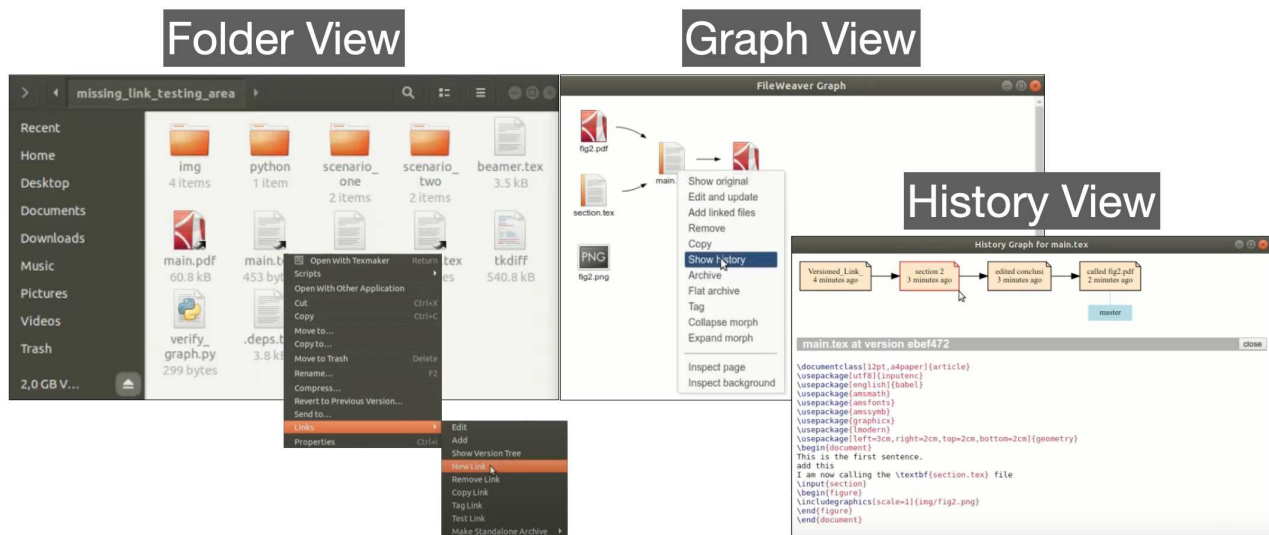
Table 5.1. Six issues observed in the interview study.

Summary

This study shows that related information is often distributed *by necessity* because scientists take advantage of the characteristics of different information artifacts. Their hoarding behavior complicates their versioning practices. They struggle to keep track, re-find and maintain consistency of this related but distributed information, particularly when collaborating asynchronously with students or colleagues.

5.5 FileWeaver User Interface

Based on the findings from the user study, we decided to focus on the file-related issues listed in Table 5.1. We designed *FileWeaver*, a prototype that augments traditional file management tools with automatic tracking and interactive visualization of file dependencies and histories. *FileWeaver* detects file dependencies and can update the dependents of a file when that file is changed. It also records file histories and can manage simultaneous versions and variants of a file.



FileWeaver's interface has three interactive views (Fig. 5.12):

- A Folder View that displays all files in the current directory, similar to a typical file browser;
- A Graph View that displays files as nodes and dependencies as directed edges. For example, a file containing a dataset loaded by a script is connected to the file holding the script;

Figure 5.12. The *FileWeaver* User Interface. The Folder View is a standard file browsing window. Users can add files together with their dependencies to the Graph View by clicking “Add File”. The Graph View was displayed by selecting *main.tex* in the Folder View, and shows the dependency graph for that file. The History View shows the history of versions of *main.tex*

- A History View that displays the history of a file as an interactive tree, where each new *version* is connected to the previous one.

In all three views, a menu with *FileWeaver* commands is accessible by right clicking a target or the background.

Folder View

FileWeaver uses the standard Folder view of the user's operating system (Fig. 5.12), with an added sub-menu in the context menu of each file to invoke *FileWeaver* commands. The key command is *New Link*, which lets users add files to *FileWeaver* together with their dependencies. Files managed by *FileWeaver* have a little arrow on their icons to provide a visual cue.

File Dependencies and the Graph View

Each file added to *FileWeaver* is a node in a graph, and the dependencies between files are directed edges between the nodes. This graph is constructed and updated automatically. Whenever the user selects a file managed by *FileWeaver* in the Folder view, she can immediately see the dependency graph of that file in a separate Graph view. Most *FileWeaver* commands are available in both the Folder and Graph views.

Black edges represent regular, up-to-date dependencies. Other colors represent a different status: a green dashed edge indicates a *FileWeaver* copy (Fig. 5.13); a red edge indicates a stale dependency; a gray edge indicates a manual dependency, e.g. between a ReadMe file and a script; and a blue edge indicates the morph group of a file (Fig. 5.14).

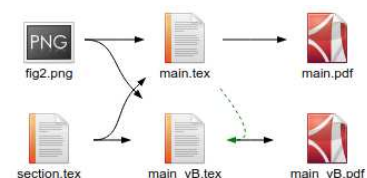


Figure 5.13: Graph View with a copy (green arrow).

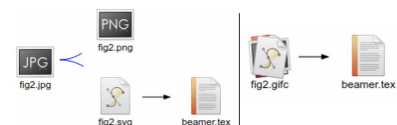


Figure 5.14: A morph file

Version Control and the History View

FileWeaver puts each file it manages under version control. When the user edits the file via the *Edit File and Update* command and saves it, *FileWeaver* asks her to enter a commit message and records a new version upon closing that file.

The *Show History* command (Fig. 5.12) opens a History view where each node represents a version, labeled with the first line of the commit message and the time it was created. Branches corresponds to the use of the *FileWeaver Copy* command. Using the context menu, the user can open any version of a file and compare two versions with color-highlighted differences.

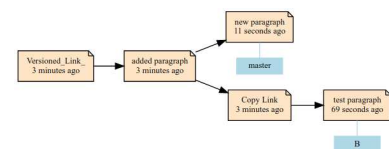


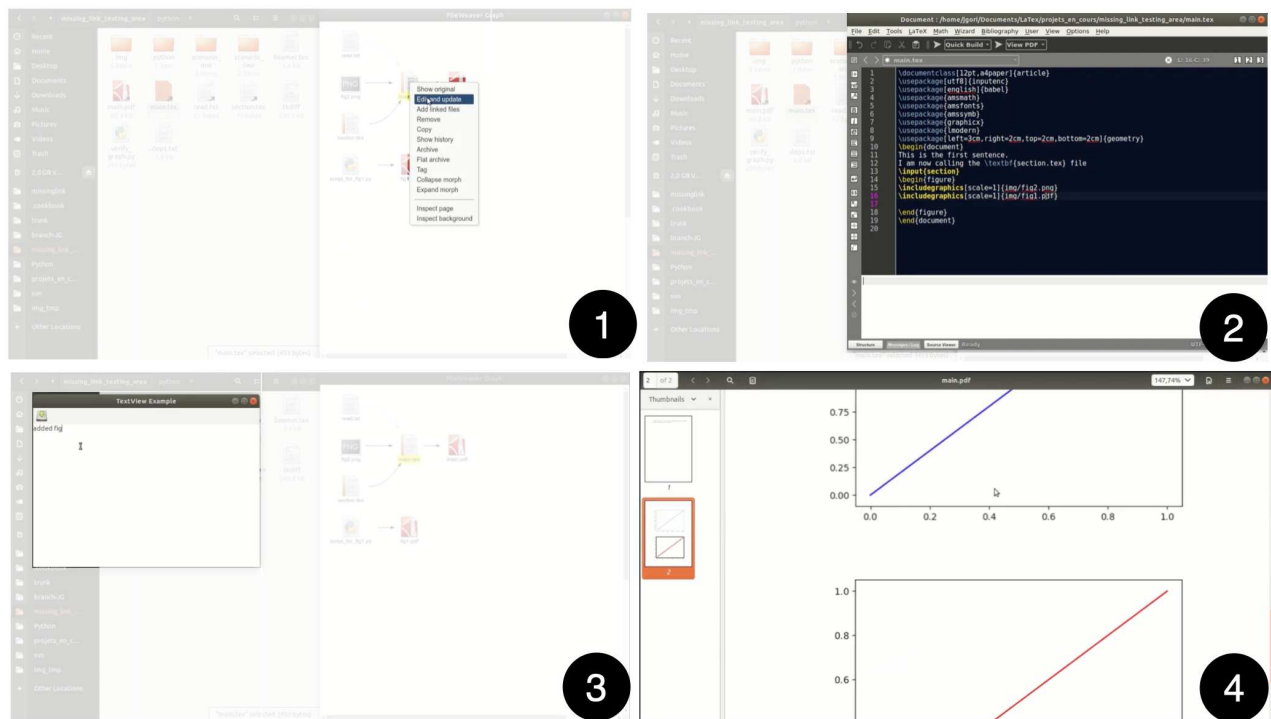
Figure 5.15: History View.

Synchronization among Views

The Folder, Graph, and History views are synchronized and kept consistent. When the user selects a file in the Folder view, the Graph view updates to display the relevant graph. When *FileWeaver* detects that the dependencies of a file have changed, the graph is animated. The user can open a new Folder view at the location of any selected file in the Graph view, and a History view from either the Folder or Graph view.

This makes navigating the file system flexible: the Folder view provides traditional navigation of the file hierarchy, the Graph view displays the dependencies of a file, irrespective of their location in the hierarchy, and the History view lets users explore the evolution of a given file over time.

5.6 The User Experience



The combination of synchronized views, automatic dependency tracking and updating, and automatic versioning opens up new possibilities for managing files. We introduce the main features that we have implemented in *FileWeaver* through several use scenarios inspired by the

Figure 5.16. Selected user interaction to detect and maintain file dependencies. (See text and video for a complete interaction)

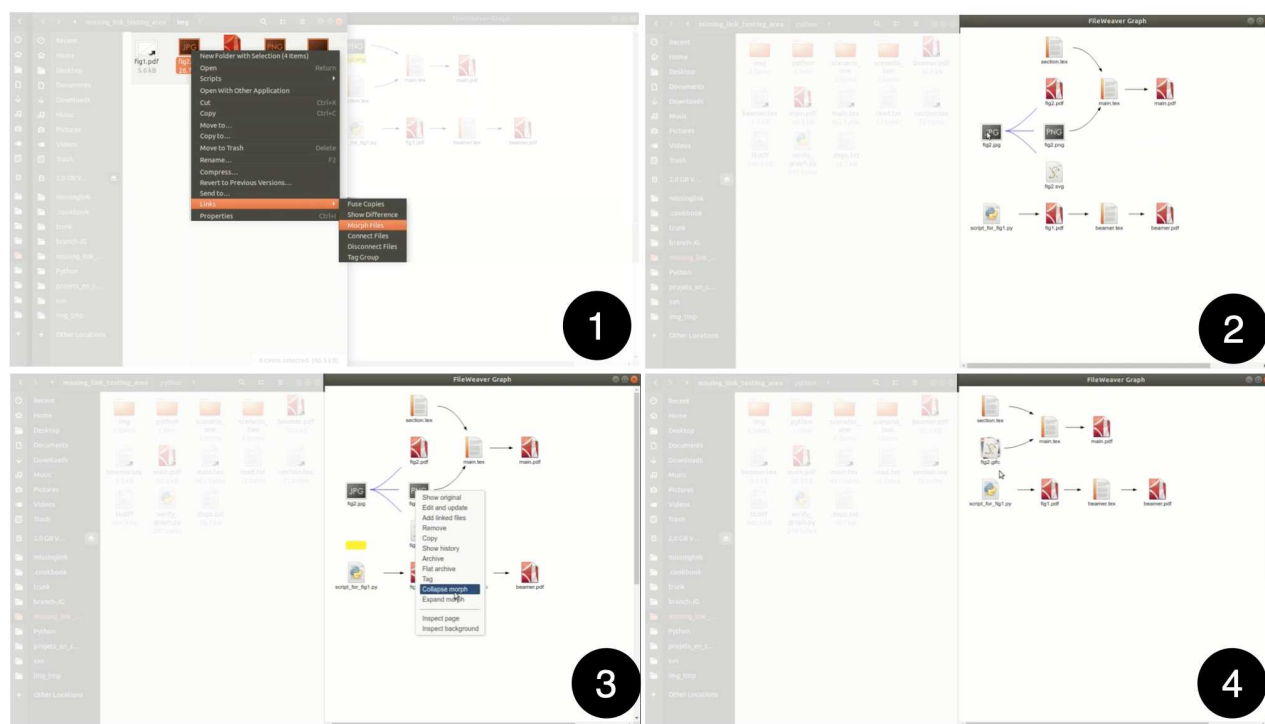
interview study, starring Jane, a data scientist ⁵.

⁵I suggest to follow the scenario with this video illustration: <https://youtu.be/PrcuF1MG1to>

Scenario 1: Detecting and Maintaining Dependencies

Jane wants to keep track of the LaTeX file `main.tex` where she writes the report. She adds the LaTeX file `main.tex` to *FileWeaver* and then adds the files linked to it. She also adds a Python script. *FileWeaver* runs the script and adds the generated figure. Jane wants to add this new Figure to her report. She edits the LaTeX file to add a figure (Fig. 5.16, 1 and 2) and is prompted for a commit message (3). After saving the message, *FileWeaver* automatically updates all the dependent files, and opens the resulting PDF file. The new Figure has been added and the graph has been updated (4). Jane opens the original folder of the Python script and edits it to change the color to green. After saving it, *FileWeaver* runs the scripts, updates the linked files, and opens the updated PDF. The figure becomes green.

Scenario 2: Managing Variants with Polymorphic Files

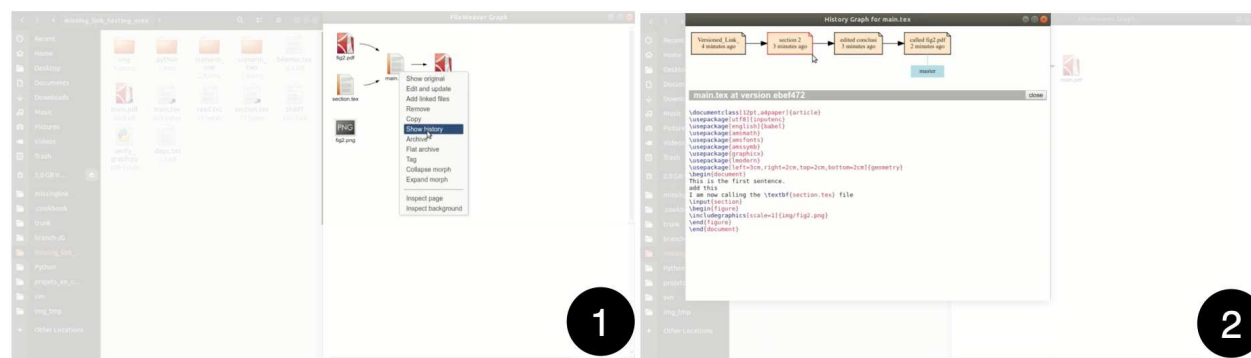


Jane opens the folder of an image file and sees four variants of the same figure. She merges them into a polymorphic file, or morph (Fig. 5.17, 1) The Graph View connects the variants with blue lines (2). Jane collapses the morph to simplify the graph and reduce clutter (3 and

Figure 5.17. Selected user interaction to manage variants of files. (See text and video for a complete interaction)

4). When she calls the morph from the Beamer file (for presentation), *FileWeaver* automatically calls the preferred extension.

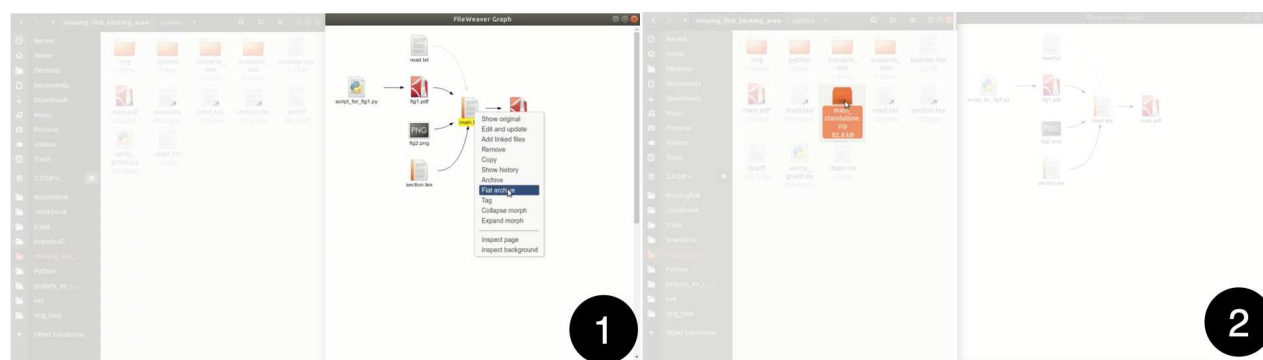
Scenario 3: Managing Histories and Versions



Jane opens the history of the main TeX file. (Fig. 5.18, 1) She first opens a previous version (2), and then selects two recent versions and displays their differences. Jane creates a copy of the file with its linked files. In the Graph View, a green arrow links the original to the copy. She updated the file linked to the copy. She first edits the copy and sees the result. She then edits the original and checks the result. When she opens the History View again, she can see the branch with the original and the copy.

Figure 5.18. Selected user interaction to manage history and version of a file. (See text and video for a complete interaction)

Scenario 4: Sharing files



Jane wants to share all files related to the main TeX file with her colleague Alex. She selects "Flat Archive", (Fig. 5.19, 1), which gathers all relevant files into a single folder (2). Jane can also create a runnable archive that also copies the hierarchy of folders. This lets Alex directly edit and compile these files.

Figure 5.19. Selected user interaction to easily share all related files. (See text and video for a complete interaction)

5.7 System Implementation

FileWeaver runs on Linux Ubuntu 18.04 LTS and was implemented by Julien Gori and Michel Beaudouin-Lafon. This section reflects their work and is included for completeness.

Graph Attributes

Nodes and edges of the graph have attributes that are stored in a separate database. Some attributes are maintained automatically, e.g. *edge update time*, which tracks when the dependency between two files was last brought up to date. *FileWeaver* compares it to the file's UNIX `mtime` attribute to decide whether to update the dependency. Each node also stores the version ID of the corresponding file, and each edge the version ID of the source side of the edge. If two files (A and B) depend on different versions of a third file (C), the edge (say, from C to A) representing the dependency that is not up to date is shown in red. This tells the user that A should be updated. If instead A and B were produced from C as part of the produce-communicate-reproduce cycle, it tells the user that the version of C that was used to produce A is accessible from the History View, whereas it would normally have been overwritten by the version that produced B.

Other attributes are under user control. For example, The user can set *node update* to false to specify that this node should not be automatically updated, e.g. because it is too computationally expensive to run the update. The user can also change the *recipes* for a given file (see next).

Recipes: Tracking and Updating Dependencies

FileWeaver uses file-dependent scripts called *recipes*, stored as node attributes, to track and maintain dependencies. These recipes specify how each file is processed (*update*), displayed to the user (*interact*) or its dependencies tracked (*trace*), allowing a high level of task automation and the ability to always keep the Graph view up to date.

The *update* recipe processes the file and produces its output if there is one. For example, the recipe for a LaTeX file can simply be `latexmk $filename`. The *update* recipe is run whenever a file needs to be updated. When a user triggers a *FileWeaver* command on a given file, *FileWeaver* checks that the dependent files are up to date, in a topological sorting order, and launches the *update* recipe on the target file if an update is indeed needed.

The *trace* recipe is optional and should produce the same file accesses as *update*, only faster. If unspecified, the *update* recipe is used instead. For example, the *trace* recipe for a LaTeX file can be simply `pdflatex $filename`. *FileWeaver* traces the file accesses resulting from running the *trace* recipe when the file dependency list needs updating, e.g. when a file is edited or newly added. *Trace* recipes are motivated by files with long compilation procedures, such as LaTeX files: while the *update* call to `latexmk` will usually result in at least three calls to `pdflatex`, the *trace* recipe only needs a single one. *Update* and *trace* recipes should include clean-up commands, if needed, to remove temporary files and avoid cluttering.

The *interact* recipe opens an editor or viewer for the file. For a LaTeX file, it calls the user's preferred LaTeX editor, e.g. `texmaker -n $filename`. *FileWeaver* runs the *interact* recipe when the user edits the file through *FileWeaver*, or when *FileWeaver* wishes to display a file to the user, e.g. to show the result of an update. When the user closes the editor (which exits the *interact* recipe), changes are automatically versioned. Note that the user can always edit a file outside of *FileWeaver*; it will then be recognized as out of date on the next call to a *FileWeaver* command.

When a file is added to *FileWeaver*, the recipes are initialized according to the file's type. Each file type has a default set of recipes stored in a `.rcp` file. *FileWeaver* currently features recipes for 9 popular file types: `.tex`, `.py`, `.pdf`, `.png`, `.jpg`, `.jpeg`, `.csv`, `.svg`, `.txt`. Users can edit `.rcp` files, share them, and specify custom recipes for any given file.

We acknowledge that writing recipes may require some computing knowledge. We expect to provide more recipe to cover more file types, and a simple editor to facilitate creating, editing and sharing them.

Links and the Cookbook

The *FileWeaver* backend runs in the background and strives to be as transparent as possible to the user. It uses a hidden folder called the *cookbook* to store its files.

Each file managed by *FileWeaver* is attributed a folder, called a *cookbook page*, in the cookbook. When *FileWeaver* starts tracking a file, that file is moved to a new cookbook page, and a symbolic (soft) link is created at the original file location, pointing to the file in its cookbook page. This leaves the user's folder virtually unchanged, except for the fact that the user now sees a symbolic link rather than a standard file. We use soft

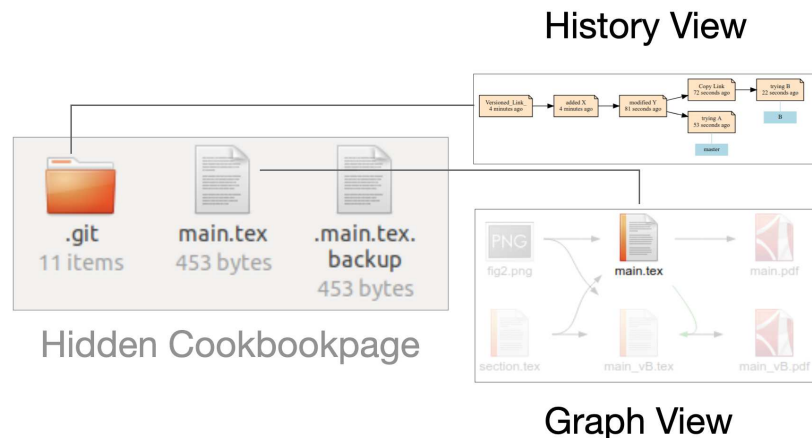


Figure 5.20. Each file managed by *FileWeaver* has a cookbook page, in a hidden folder called *cookbook*. A cookbook page also holds the version control repository.

links rather than hard links because the latter do not span filesystems (including partitions). Also, most file browsers display symbolic links with an arrow on top of the file icon, giving the user a clue that the file is managed by *FileWeaver*.

A cookbook page also holds the version control repository for that particular file (Fig. 5.20). Cookbook pages are named after the device and i-node number of the file that they store, making for a cheap and memoryless, hashtable-like one-to-one mapping between files and cookbook pages⁶.

Tracking File History Through Version Control

Whenever a file is edited via the *Edit and Update File* command, *FileWeaver* prompts the user for a message and automatically commits the file to the version control repository located in the file's cookbook page. While *FileWeaver* could automatically detect file changes and perform actions, we avoided this behavior to give users a better sense of control. The *FileWeaver Copy* command creates a new cookbook page for the copy but shares the same repository as the original file so that the user can merge the two files with the *Merge Files* command. All edges to parent nodes and attributes are copied from the original file, preserving custom settings.

Polymorphic files

When the user selects several files and invokes the *Morph Files* command, *FileWeaver* adds these files to its cookbook as usual, and assigns them to a *morph group*. Instead of creating a soft link for each file, it creates a single soft link for the morph with a `.gifc` extension

⁶ Some text editors use temporary files that are renamed into the original file after saving, therefore changing the i-node number of the file. *FileWeaver* creates an additional, hidden hard link to the file when it is added to the cookbook page to prevent the i-node from changing. This hard link can also serve as a backup access to the file.

(generic image format container). When a file calls a `.gifc` extension, *FileWeaver* goes through the graph and looks if there is an edge between that file and a member of the morph group. If so it redirects the soft link with the `.gifc` extension to that morph group member; otherwise it goes through the default image formats associated with that file, e.g. `pdf/png/jpg` for a LaTeX file, and creates an edge with the corresponding morph group member. Whenever a file is used by *FileWeaver*, e.g. as part of an update, *FileWeaver* checks if that file uses a morph. If so, it makes sure that the soft link is pointing to the right member of the morph group. Since some systems, e.g. LaTeX, use the file extension of included files, *FileWeaver* creates a second, temporary soft link with the right extension, and runs a stream editor to search and replace the morph file name with the right extension. When the update is complete, the original file content is restored and the second soft link is removed.

Implementation Details

FileWeaver runs on Linux Ubuntu 18.04 LTS. The backend is written in just under 5000 lines of Python with calls to standard UNIX tools such as `bash` and `sed`⁷. File dependencies are detected by running `Tracefile`⁸ on the `trace` recipe to track file accesses. The graph database is managed via the `graph-tool` library⁹. The backend currently runs only on Linux due to the use of `Tracefile` and `strace`, but we are considering alternatives for Mac OS.

The Folder view is the GNOME Nautilus file manager¹⁰ (Fig. 5.12 left). We use `nautilus-python`¹¹ to write extensions to Nautilus for running the backend and adding the menu of *FileWeaver* commands to the standard Nautilus file menu.

Both the Graph and History views (Fig. 5.12 center) are implemented with `NWJS`¹², a platform to create web-based desktop applications. The Graph View sends commands to the backend through a simple pipe, and receives updates to the graph through a shared file. It uses `dot`¹³ for the layout of the graph. The version control system is `Git`¹⁴. Although not designed to deal with binary files such as figures, it can handle a moderate amount of them without problem. We plan to explore extensions to `Git` that deal with large or numerous binaries¹⁵. The History View directly calls `Git` commands for the different version management commands. It uses `git2dot`¹⁶ to create the version tree and `diff2html`¹⁷ to display version diffs. The *FileWeaver Copy* command uses the `git-worktree` feature to check out several branches at a time.

⁷ <https://www.gnu.org/software/>

⁸ <https://github.com/ole-tange/tangetools/blob/master/tracefile/tracefile.pod>, which is a wrapper around Linux's standard `strace` (<https://linux.die.net/man/1/strace>).

⁹ <https://graph-tool.skewed.de/>

¹⁰ <https://github.com/GNOME/nautilus>

¹¹ <https://github.com/GNOME/nautilus-python>

¹² <https://nwjs.io/>

¹³ <https://www.graphviz.org>

¹⁴ <https://git-scm.com/>

¹⁵ such as `git-lfs`, `git bup`, `git-annex`

¹⁶ <https://github.com/jlinoff/git2dot>

¹⁷ <https://diff2html.xyz>

5.8 Discussion and Evaluation

Issue	<i>FileWeaver</i> Features
Hard to re-find related but distributed information.	Visualize file relationship in interactive graphs.
Manual synchronization.	Automatically update dependent files.
Difficult version management.	Visualize version history in a timeline.
Lack of support to manage file variants, e.g. format.	Abstract file format into generic file.
Tedious creation of shared document.	Automatically collect all dependencies in an archive.

Table 5.2. How different features of *FileWeaver* address issues observed in the interview study

Table 5.2 lists the most important features implemented in *FileWeaver* and how they address the issues identified in the formative study (see Table 5.1). We complement this analysis with a more general qualitative evaluation of *FileWeaver* based on Green et al.’s cognitive dimensions (Green, 1990). We leave a formal, longitudinal user study to future work.

Visibility and Hidden Dependencies

Visibility is the ability to view components easily, while hidden dependencies reflects whether important links between entities are visible or not. *FileWeaver* achieves high visibility and low hidden dependencies by making dependencies among files explicit and visible in the Graph view. Dependencies are represented by edges that can be manipulated, e.g. change the update rule. Automatically running recipes also reveal dependencies as files are edited, unlike tools such as, e.g. Makefiles, that require manual editing to describe the dependencies.

In typical folder-based views, all non-hidden files are visible, which may clutter the folder. *FileWeaver*’s polymorphic files let users view the multiple variants of a file as a single entity, making their relationship more explicit and reducing clutter.

Viscosity

Viscosity refers to resistance to change. *FileWeaver* has low viscosity because it automatically propagate the user’s changes to a file to its dependents, achieving consistency between files without explicit user action.

FileWeaver also automatically stores successive versions of a file so that users can examine and revert to previous versions using the History

view. It supports exploration of alternatives by making it easy to create parallel copies and merge them.

FileWeaver prompts the user for a message when a file is edited via *Edit and Update File*, which has high viscosity. Although encouraging users to document their changes can help them re-find the right version in the future, this requires extra effort. We plan on simplifying this process, e.g. by suggesting auto-generated commit messages.

Error-proneness

FileWeaver reduces errors by automating the processes of dependency update and versioning. The former reduces the risk of running the wrong command to update files, and the latter makes it possible to get back to a former version in case of an error. In future work we plan to use version control also for the graph itself, which would make it possible to easily recover deleted files and dependencies.

Secondary Notation

Secondary notation refers to the ability to carry additional information. *FileWeaver* lets users tag files to rename the node labels while keeping the underlying files with their original name. Users can also edit the *recipe* and *interact* scripts of individual files to tailor them to their needs. In future work we will make it easier to define file types and edit default recipes, as they are not currently readily accessible to novice users.

Role Expressiveness

Role expressiveness refers to how obvious the role of each component is. Each view in *FileWeaver* has a specific role: Folder view for regular file management, Graph view for managing dependencies, and History view for versioning. Role expressiveness could be further improved by decoupling components and giving users more flexible ways to combine them.

Premature Commitment

Premature commitment refers to constraints on the order to complete tasks. *FileWeaver* limits premature commitment by letting users add files to the system whenever they want. File names, contents and locations can also be modified at any time. Asking for a commit message when saving a file is a form of premature commitment, and should be

made more flexible.

5.9 Conclusion

This chapter studies how knowledge workers manage their information. I interviewed 23 scientists and showed that the information they manage is often distributed, because they take advantage of the characteristics of different information artifacts and rely on specialized tools. Although scientists develop strategies to cope with re-finding information, they still struggle to manage versions, maintain consistency and keep track of related distributed information. We introduced *FileWeaver*, where the traditional folder view is augmented by an interactive Graph View that displays dependencies among files and a History View that lets users interact with the different versions of a file, supporting navigation and re-finding tasks. *FileWeaver* also features polymorphic files, which groups the different variants of a file into a single, generic format.

We plan to evaluate *FileWeaver* in a realistic setting to gather feedback and improve the interface. We also need to assess its scalability on larger sets of files. Future improvements include making recipes editable by novice users, including through automatic capture of commands; simplifying the commit process, e.g. by suggesting commit messages; and detecting dependencies in files such as zip archives and Word documents. Finally, *FileWeaver* has great potential for collaboration, since the file contents and its relationships are under version control and can therefore be pushed to a remote server for sharing.

What have we learned about representation and manipulation? One power of visual representation is that it gives users the ability to see invisible things. Once users see them, they can understand and learn to use them. The invisible things described in this chapter are the *relationships among files* that users typically have to manage in their heads. We make these invisible file relationships visible so that “*the display screen relieves the load on the short-term memory by acting as a sort of “visual cache.*” (Smith et al., 1982)”. We created three new representations based on three types of file relationships: dependency, variation and version. The dependency relationship is represented by a graph; the variation relationship is represented by a group; and the version relationship is represented by a tree.

In summary, I expand our understanding of representation by intro-

ducing three new representations based on the invisible file relationships that users typically have to manage in their heads. These new representations demonstrate the power of visualizing the invisible. I argue that these new representations improve users' understanding of their own files and enable a flexible way to manipulate them.

6

Design Principles

The design principles of reification, polymorphism, and reuse introduced by Beaudouin-Lafon and Mackay (2000) strongly influenced the design of the new representations investigated in this thesis. I now analyze these new representations and reflect on my experience designing them. I propose three new design principles, with the goal of enriching existing design principles and providing designers with a new perspective to think about designing digital objects. These three design principles are:

1. Granularity
2. Individuality
3. Synchronization

6.1 Granularity

Beaudouin-Lafon and Mackay (2000) refer to reification as “*the process by which concepts are turned into objects*” and describe several existing examples including a *style* in a text editor, a *group* in a graphical editor and a scrollbar for documents. The first principle I propose is that *The reified representations should be adapted to the granularity of the users’ objects of interest*. Using an analogy with objects in the physical world, I categorize granularity into two types: *material* and *scale*.

Granularity of Material refers to the property of the distinguishable pieces. I will illustrate this using text as an example. In a text editor, text is represented at a character level ¹. The Granularity of Material is the character. This is appropriate for text editing but may not be for font design. When designing a new font, font designers need to manipulate the strokes, the glyphs, the angles between two lines etc. If the text were still represented in the character level, it would not be possible; the designer simply cannot manipulate these essential elements to design their font. The text is thus represented as a graphical shape (Fig. 6.1, right). The Granularity of Material is finer than for text editing; it is at the level of a shape. This level is appropriate because designers can now manipulate these elements to define a font. Although the object in both examples is the text, it has different representations in terms of granularity of material in their digital environments. An appropriate representation should match the granularity of the users' object of interest.

¹ In text editing, users manipulate text at several levels of granularity such as character, words, paragraphs, etc. I argue that these levels are in the category of granularity of scale, not of materials because the atomic level is still the character. Furthermore, despite the fact that these levels are in the category of granularity of scale, the principle still applies because text editing tools are adapted to these different levels, e.g. by making selection.

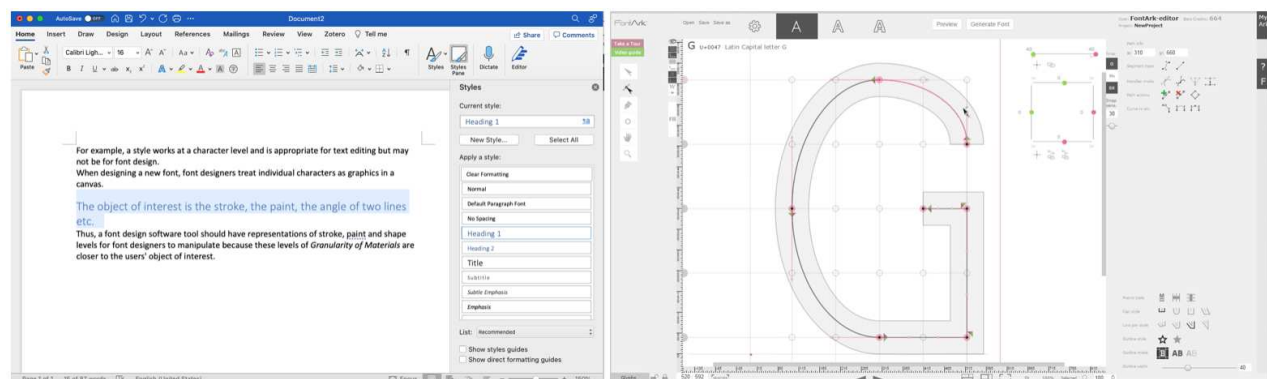


Figure 6.1. Text is treated as a character in a word processor (left) but as a graphic in a font design software. Image taken from: Microsoft Word and FontArk (Source: <https://fontark.net/farkwp/>).

Granularity of Scale refers to the size of the distinguishable pieces. In Beaudouin-Lafon's examples, a style (in a text editor) works at the level of character; a group works at the level of individual shape and a scrollbar works at the level of a row of pixels spanning the document. Now imagine a situation where you can only apply style to the whole text, not just part of it, in a text editor; a situation where you can only group objects by layers, not arbitrary objects in the same layer, in graphical editor; and a situation where you can only scroll the document by a whole paragraph, not a line of text. Although these are just imaginary examples, it helps understand the importance of designing representations whose *Granularity of Scale* is matched to the size of users' object of interest. Another research example of this approach is MoveOn (Rivière et al., 2019), which lets dancers segment the seeker bar of the video into short, repeatable clips to support the

dance learning process. When learning a movement, the dancers' object of interest is a short clip of a single, digestible movement rather than the entire video. This shorter clip thus has a finer-grained scale to better support the dancers' learning process.

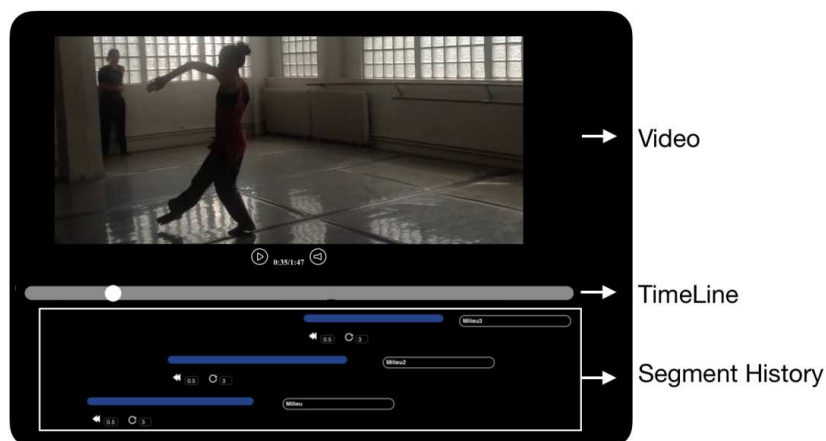


Figure 6.2. Rivi re et al. (2019)'s MoveOn interface with short clips representing a digestible movement.

Example: Textlets

In the Textlets project, we observed that many users' editing tasks are at the level of the selection (Granularity of Scale). The action of selecting indicates the user's intention of specifying the object of interest ². This led to the design of a new representation at the level of the selection, namely the textlets. The benefits of such a design is demonstrated in the examples of *countlets* and *variantlets*. Technical writers often need to constantly count words and characters for a single or multiple sections in a document. But standard word processors only constantly display the count for the whole document. Word count of a specific range of text requires users to repetitively select the text. Because Textlets work at a *Granularity of Scale* closer to the users' actual object of interest during an editing task, users can easily keep track of their word counts in real-time without repetitive selection. Similarly, in the example of *variantlets*, technical writers often want to explore alternatives at the granularity of word, sentence or paragraph. *Variantlets* let users keep track of the changes made to a selection which could be a word, sentence or paragraph, rather than the entire document.

² A shift from prefix syntax to suffix syntax helps systems to get rid of the mode, thus improving the usability (Tesler, 2012)

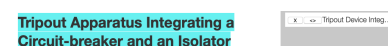


Figure 6.3: Variantlets

Example: Passages

When working with patent examiners, we found that they not only work with multiple documents but also *snippets of text* from multiple resources and remix them as needed. For example, they iteratively

take keywords from prior art documents to search for more relevant documents; they compare text snippets from multiple documents to understand the differences; they also cite specific text snippets (or passages) when writing the final report. These observations informed our design to focus on creating a finer granularity of representation, the passage. Both textlets and passages give users the ability to create selection-level flexible representations, which are closer to the users' objects of interest in their daily tasks.

Example: FileWeaver

While the above two examples create representations that are more fine-grained, the representation of a "*morph*", where multiple file variants can be merged together, is more coarse-grained. We observed that when a scientist talks about, e.g. a figure in her experimental results, she refers to "that box plot she created the other day". She does not need to know the format (e.g. png, jpg and svg) or resolution of that image file except when it is really needed. This is the rationale behind the design of polymorphic files: a group of similar files forming the user's concept of a document that is meaningful to them.



Figure 6.4: A morph represents a group of similar files.

6.2 Individuality

Polymorphism (Beaudouin-Lafon and Mackay, 2000) refers to *the ability to apply a command to a group of objects, of the same or different types*. It is a powerful way to manipulate multiple objects beyond just mimicing the reality (Jacob et al., 2008). Many research tools leverage the power of polymorphism. For example, Draco (Kazi et al., 2014) allows users to add granular motion by directly manipulating example objects. The performed transformation is recorded and is applied to each of the individual, repeated objects generated from the patch (Fig 6.5, first row). DataInk (Xia et al., 2018) allows users to interact with the subgroup of an object collection by specifying the mapping between data dimensions and visual properties of the glyph (Fig 6.5, second row). Unlike the focus on expressivity and creativity in these contexts, knowledge work requires accuracy and consistency. For example, patent examiners are required to cite a specific quote in order to accurately communicate and justify their decisions with patent attorneys. Scientists need to keep a specific set of experimental conditions and the corresponding results consistent. These differences raise other challenges to apply polymorphism and lead us to the second design principle: *in addition to the ability to apply a command to a group of objects, also support the ability*

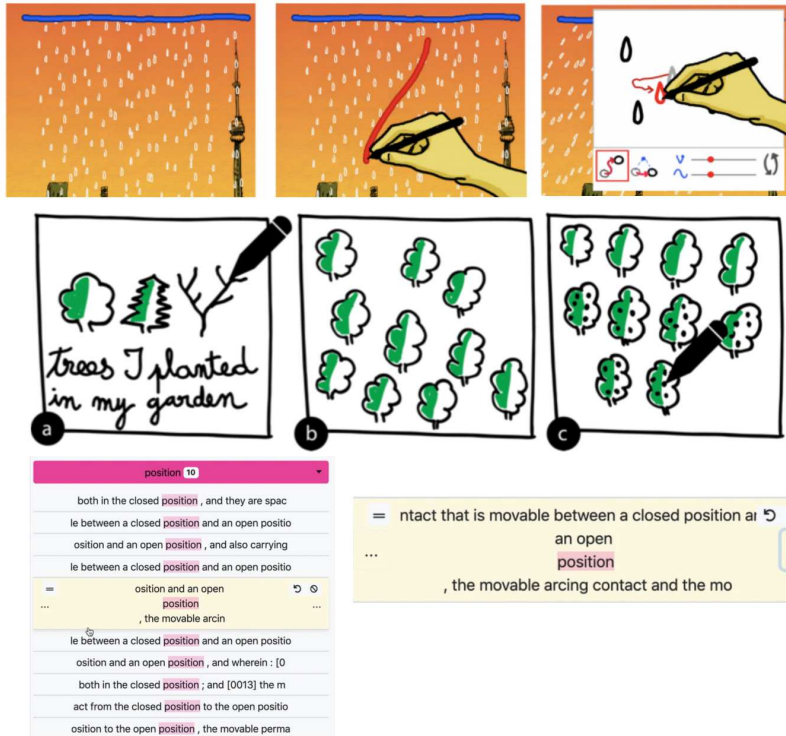


Figure 6.5. Interactions to manipulate groups of objects in Draco (Kazi et al., 2014), DataInk (Xia et al., 2018) and Textlets.

to interact with individual and subgroups of objects. Enabling interaction with individual element of a group further extends the flexibility of polymorphism. This is linked to the principle of Granularity because it is about decomposing a group into its element. It is also different because Granularity focuses on the object itself while Individuality focuses on the relationship among similar objects, i.e. the group.

Example: Textlets

In text editing tools, “replace all” is a command that replaces all the occurrences of a search term with another in the whole text. However, one interesting observation that emerged from the interview study is that most legal professionals do not use the “replace all” command. They prefer to replace occurrences one by one instead. A deeper investigation revealed that it is because they need to ensure the correctness of each match depending on the individual context. For example, in a situation where they change to a noun that has a different gender or number, they also may need to change the verb and article of that sentence. This is especially true for our users who primarily work on French documents.

The current “*replace all*” command is a limited polymorphic command. The limitations are due to three reasons. First, users cannot see the context of each individual occurrence to make decisions on which ones can be replaced directly and which ones need further edits. Second, users cannot see what has been replaced before. They cannot check the previous replacement unless they restart the search tool with the replaced term. Finally, users cannot select a sub-group of all occurrences to apply different strategies ³.

Searchlets, one type of Textlets, extends the flexibility of polymorphism by enabling individual-level commands. Three individual-level commands of *searchlet* encourage users to perform “*replace all*”: 1) extendable context, 2) individual undo, 3) partial selection (Fig 6.5, third row). Extendable context allows users to see and assess the context of an individual occurrence to make decisions. Individual undo lets users revert back to the text before the replacement to make comparisons. Partial selection gives users the ability to select a subset of the occurrences that can be replaced all at once.

Example: Passages

To facilitate the process of manually moving each passage from the side panel to the table, we designed a feature where users can drag the document title to the header of the table. This inserts all the passages from this document into the column at once. However, the current design does not allow users to just move a subgroup of passages. A design improvement is to allow users to specify the sub-group of passages and then move them together.

Example: FileWeaver

We designed “*morphs*” to let users manage a group of conceptually-similar files. But we also maintain the ability for the users to interact with individual files, e.g. by expanding the morph, when they need to.

³ Miller and Marshall (2004)’s also realized this problem and intended to improve it. Their cluster-based search and replace prototype automatically groups the matches so that users can replace similar matches at once.

6.3 Synchronization

The last design principle is to *always synchronize new and existing representations*. The rationale is that whatever representations users interact with, if the underlying information they are interested in is the same, changes in the information should be reflected in the representations accordingly and consistently. Designers might also consider provid-

ing linking mechanisms between these representations. This could enhance users' conceptual model about the relationship between two representations and a smooth shift between the two when the task changes. For example, most LaTeX editor, such as Overleaf ⁴, provides at least two representations, the source plain text editor and the output PDF. Although Overleaf allows users to navigate to the location of corresponding content between these two representations by clicking a button, the selection is still not synchronized. Also, these representations are not synchronized in real time and users still need to compile each time to see the updated PDF. These can be areas of improvement using the principle of Synchronization.

⁴ <https://www.overleaf.com>

Example: Textlets

When designing textlets, we needed to make a choice about the location of the new selection objects. We considered several alternatives: integrated in the main text itself, represented in a separate panel on the side or integrated into the scroll bar. The advantage of having a separate place for new reified text selection objects is that users can have an independent space to work on these objects without interfering with other tools, e.g. the scroll bar. But we were also concerned that the new representation on the side panel would shift users' attention away from the main text. In the end, we decided to present them in a side panel because we wanted to test them independently from other tools and wanted to study how these new representations would affect the users' editing behavior. We keep the main text and the Textlets side panel synchronized and provide ways for users to switch between these two representations. For example, clicking on the textlet navigates the main document to the location of the occurrence and highlights it. Similarly, clicking on an occurrence in the main document highlights the corresponding textlet in the side panel.

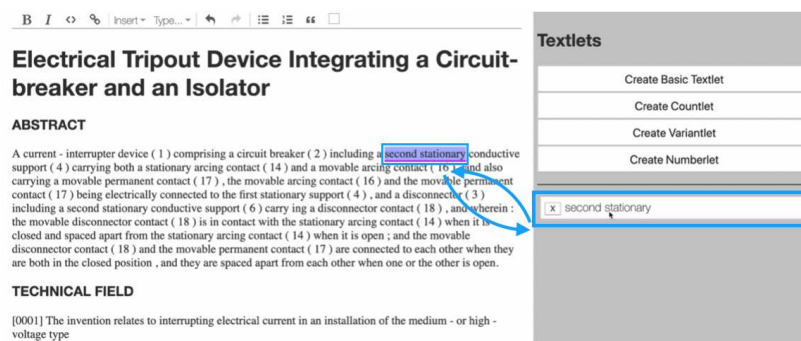


Figure 6.6. Synchronization between textlets in the side panel and the main text

In the evaluation of *Searchlets*, participants found the side panel surprisingly useful because they did not need to go back to the main text. This suggests that participants indeed shift their attention to the side panel but for a good reason. Their attention is now focused on more relevant information. The side panel provides a filtered view of the document with relevant information for the user's task at hand, leaving the extra information in the main text accessible only when needed. We also observed that several users did go back to the main text by clicking on the textlet on the side panel to locate it in the main text, i.e. using it as a navigation tool.

Example: Passages

The synchronization principle is especially relevant for Passages. Passages need to maintain two types of synchronization: the passage itself and the side panel. Currently, the passage itself is always synchronized across applications but the side panels are not. Our original argument is that the side panel is task- and application- specific. For example, users might want to use different sets of passages in different applications, e.g. Table and Editor. However, this causes inconsistencies among multiple side panels and a lack of single source of truth, making it hard for users to manage the passages. If we were to apply the principle of Synchronization to the side panel, we would synchronize all the side panels across applications but preserve the structures of the passages for that specific application, e.g. if users have tagged them. We could also reduce the number of side panels to one, and make it standalone and work with other applications. In this case, we would only need to synchronize one side panel instead of multiple ones. This would also simplify the user's conceptual model.

Example: FileWeaver



Figure 6.7. Synchronization between the Graph View and the Folder View

FileWeaver is not intended to replace the existing file manager but to

provide a new representation by visualizing the relationships among files of users' interest. We applied the principle of Synchronization so that every user command, either from the Folder view or the Graph view, affects both views consistently.

6.4 Summary

As described in Chapter 2, Bill Verplank defines interaction design as *representation for manipulation*. In graphical user interfaces, the representations are the graphical objects and the manipulations are the commands to act on the them. An interaction design's job can be broadly considered as to choose the appropriate representations and commands. In this chapter, I proposed three design principles that can help guide interaction designers to create appropriate representations. Beaudouin-Lafon et al. (2021) introduced Generative Theories of Interaction and recognized the idea of using concepts and principles to create new designs. Following their method, I created a summary table (Table 6.1) with sample questions, to help interaction designers apply the concepts and principles introduced in this chapter.

Designing Representations	Analytical	Critical	Constructive
<i>Concept</i>			
Representation	What are the existing representations that users can manipulate?	Do these representations match the objects of interest in users' minds?	Are there other objects of interest that have not been represented in the user interface? Should new representations be designed?
<i>Principle</i>			
Granularity	Which level of granularity (both scale and material) are the representations at?	Are the granularity of representations adapted to users' objects of interest?	Which granularity of representations should we create? Finer-grain or coarser-grain?
Individuality	Which representations can be grouped? Which commands apply to individual and group representations?	Should commands apply to individual elements and subsets in the group?	How to make commands apply to individual elements without breaking the group or losing the ability to apply to the whole group?
Synchronization	Which representations describe the same underlying information? Are these representations synchronized?	Should representations be synchronized?	How to make representations synchronized?

Table 6.1. Questions that can be asked to apply the concepts and principles in this chapter analytically, critically and constructively.

These principles are based on my own design experience and analysis of existing interfaces. The scope of these principles should be further tested. Beaudouin-Lafon and Mackay (2000) validated their principles in the interface redesign Coloured Petri Net. I plan to apply the proposed principles in other areas such as presentation software, spreadsheet, etc. Presentation software is an interesting example because it already has a rich set of representations such as the slide deck, the slide and the shape and text fields in each slide. I believe that by studying specific examples and applying the proposed principles to redesign them, we can test the power and limitation of these principles and possibly learn new ones. Another way to test these principles is to collect and analyze representations in the existing literature. I have analyzed several representations in this thesis and I would like to conduct a more systematic and exhaustive literature review focusing on representations.

7

Conclusion and Future Work

7.1 Conclusion

This thesis explores how to design representations for today’s document-related knowledge work. It challenges some fundamental and taken-for-granted representations in graphical user interfaces. My intent is to create a new perspective on knowledge work and demonstrate that we can still design innovative interfaces to support it.

I study today’s knowledge work from three main activities around documents: editing (Chapter 3), analyzing (Chapter 4) and managing (Chapter 5). By studying extreme users (von Hippel, 1986) for each activity, I reveal the limits of current representations. By taking on a principled and theory-driven approach, I designed three flexible and effective representations to support these activities.

Document Activity	Extreme User	Representation
Editing	Lawyers	Textlets
Analyzing	Patent Examiners	Passages
Managing	Scientists	FileWeaver

Table 7.1. Thesis overview with three aspects of document-related knowledge work.

My first focus is document editing, one of the most common document activities. We conducted contextual interview with lawyers and showed that they struggle to manage the internal dependencies and to maintain consistent use of vocabulary within their documents. They do not use the automatic features, e.g. “replace all”, because they do

not trust it. This leads to the design of Textlets (Fig. 7.1), a persistent representation of text selection. The proof-of-concept implementation demonstrates how Textlets address identified user issues. We studied the use of Textlets in an observational study where participants found it useful and effective.

My second focus is document analysis. Through a collaboration with the European Patent Office, I conducted contextual interviews and observations with patent examiners. The results showed that patent examiners need to analyze information snippets from multiple documents across various interconnected activities, including searching, collecting, annotating, organizing, writing and reviewing, while manually tracking their provenance. I designed Passages, a representation of text selection that can be reused across multiple tools while maintaining the ability to retrieve its source document when needed. We conducted a walkthrough workshop and an observational study to evaluate the use of Passages. The results show that that Passages facilitate knowledge workers practices and enable greater reuse of information.

I finally turn my focus to document management. My interviews with scientists from a broad variety of backgrounds revealed that information is distributed among many artifacts by necessity because scientists take advantages of different characteristics of these artifacts. They find it difficult to keep track of, re-find and maintain consistency among these related but distributed information. This leads us to create FileWeaver, a system that automatically detects dependencies among files without explicit user action, tracks their history, and lets users interact directly with the graphs representing these dependencies and version history. We qualitatively evaluate FileWeaver based on Green (1990)'s cognitive dimensions to identify areas for improvement.

The design of three document representations are inspired by the same theoretical principles: reification, polymorphism and reuse (Beaudouin-Lafon and Mackay, 2000). I reflect on our experience designing and evaluating these representations and propose three new design principles: granularity, individuality and synchronization.

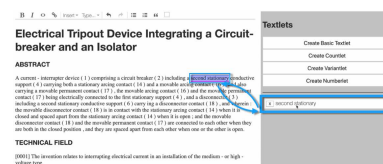


Figure 7.1: Textlets

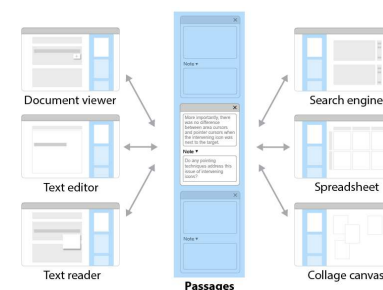


Figure 7.2: Passages

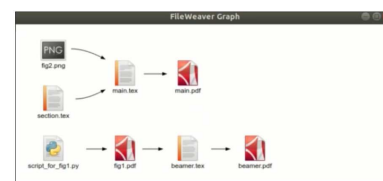


Figure 7.3: FileWeaver

7.2 Future Work

From Extreme Users to General Users

While this thesis has focused on designing for extreme users, one next step is to find out how we can generalize to more ordinary users. This question has two levels.

1. How are the empirical findings still relevant to ordinary users?
2. Would our designs still work for ordinary users?

To address the first question, one method is to survey a large population of ordinary users based on the findings of this thesis. For example, one finding with lawyers is that they need to keep consistent use of terms but it is not clear if other user populations have the same need. To generalize this finding, the survey should focus on the search and replace experience of a variety of users.

The results of the first question could provide us with some initial hints about the second question. If the results show that ordinary users have similar needs, it is likely that our designs could still work for them. If the results show that they have different needs, we then need to study them in more detail. Another method is to let a variety of users try out or “experience” our designs. By doing so, we not only use our designs to further probe their needs and also evaluate the usefulness of our designs. For example, I am interested in making Passages a stand-alone tool ¹, and deploy it to various users for them to use for a period of time².

Human Value in AI

All three prototypes in this thesis are purely based on direct manipulation and are agnostic of the past interaction history. Prior work such as the Lumiere project (Horvitz et al., 1998) have proposed adaptive systems that offer tailored assistance by identifying the user’s goals and needs. More recent work, e.g. BIGFile (Liu et al., 2018), uses Bayesian Information Gain framework to assist the user in navigating to a desired target (file or folder). One future direction is to explore how to bring both direct manipulation and adaptive systems together, especially from a user’s perspective.

In our Passages project with patent examiners, we observed their reluctance to use the tool that automatically generates a report. From

¹ Yoink is a good example of a visible clipboard: <https://apps.apple.com/us/app/yoink-improved-drag-and-drop/id457622435?mt=12>

² This is also related to literature on Product Development such as Eric Ries’s Lean Startup: <https://www.amazon.com/Lean-Startup-Entrepreneurs-Continuous-Innovation/dp/0307887898>

this, we learned that the nature of their intellectual work is based in the human-human communication, as one patent examination manager said:

If we are moving in the direction where the communication will be automatically generated by computer, we will be sending the message that we don't really do the intellectual work. So [the patent attorneys] will be less likely to be convinced, the same way I am less likely to be convinced.

This illustrates the importance of maintaining user control in this human-human communication. At the interface level, the difference can be very subtle. For example, instead of generating the full report and asking the users to post-edit it, our Passages editor gathers all the relevant passages and waits for them to be selected by the user. The subtle interface difference between post-editing and active composing maintains the user in control to decide how they want to write their documents.

Final Thoughts

During my mid-term presentation of my Ph.D, Jean-Daniel Fekete asked me:

What is a document?

I was not able to answer that question. It was not because I do not know how to do a quick search on Google but I did not know how my thesis could answer that question. I have been thinking about this question since then. Two years later, I came across him on the train and I told him that I might have the answer for that question. He was ready to listen. I said:

A document is an artifact that persists information and is intended for human-to-human communication.

There are two elements in my definition: persistent and human-to-human communication. First, a document must hold and persist information. The traditional way is to have it written. From this aspect, human speech is not a document because it disappears with time. Second, a document is intended for human-to-human communication. This implies the involvement of at least two parties, the writer and the reader, and they are both humans. From this aspect, the computer code is not a document if it is intended to be read only by the computer (non human). The computer code together with comments and documentation is a document because it is intended to be read by a hu-

man. Together, these are the two essential elements of my definition of a document. He thought about it for a bit and said:

OK. I accept this answer.

References

- Abrams, D., Baecker, R., and Chignell, M. (1998). Information archiving with bookmarks: Personal web space construction and organization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, page 41–48, USA. ACM Press/Addison-Wesley Publishing Co.
- Adar, E., Teevan, J., and Dumais, S. T. (2009). Resonance on the web: Web dynamics and revisitation patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 1381–1390, New York, NY, USA. Association for Computing Machinery.
- Adler, A., Gujar, A., Harrison, B. L., O'Hara, K., and Sellen, A. (1998). A diary study of work-related reading: Design implications for digital reading devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, page 241–248, USA. Association for Computing Machinery.
- Alexander, J., Cockburn, A., Fitchett, S., Gutwin, C., and Greenberg, S. (2009). *Revisiting Read Wear: Analysis, Design, and Evaluation of a Footprints Scrollbar*, page 1665–1674. Association for Computing Machinery, New York, NY, USA.
- Atkin, A. (2013). Peirce's Theory of Signs. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2013 edition.
- Baecker, R. M., Nastos, D., Posner, I. R., and Mawby, K. L. (1993). The user-centered iterative design of collaborative writing software. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 399–405, New York, NY, USA. ACM.
- Balakrishnan, R. (2004). "beating" fitts' law: Virtual enhancements for pointing facilitation. *Int. J. Hum.-Comput. Stud.*, 61(6):857–874.
- Bardram, J., Bunde-Pedersen, J., and Soegaard, M. (2006). Support for activity-based computing in a personal computing operating

- system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 211–220, New York, NY, USA. Association for Computing Machinery.
- Barreau, D. and Nardi, B. A. (1995). Finding and reminding: File organization from the desktop. *SIGCHI Bull.*, 27(3):39–43.
- Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. (2003). Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Human-Computer Interaction—INTERACT '03*, pages 57–64. IOS Press.
- Beaudouin-Lafon, M. (2000). Instrumental interaction: An interaction model for designing post-wimp user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 446–453, New York, NY, USA. ACM.
- Beaudouin-Lafon, M. (2004). Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 15–22, New York, NY, USA. ACM.
- Beaudouin-Lafon, M., Bødker, S., and Mackay, W. E. (2021). Generative theories of interaction. *ACM Trans. Comput.-Hum. Interact.*, 28(6).
- Beaudouin-Lafon, M. and Mackay, W. E. (2000). Reification, polymorphism and reuse: Three principles for designing visual interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pages 102–109, New York, NY, USA. ACM.
- Bederson, B. B. and Hollan, J. D. (1994). Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, UIST '94, page 17–26, New York, NY, USA. Association for Computing Machinery.
- Bergin, T. (2006). The origins of word processing software for personal computers: 1976-1985. *Annals of the History of Computing, IEEE*, 28:32–47.
- Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. (2015). Soylent: A word processor with a crowd inside. *Commun. ACM*, 58(8):85–94.
- Bertin, J. (1983). Semiology of graphics; diagrams networks maps. Technical report.
- Bier, E. A., Ishak, E. W., and Chi, E. (2006). Entity quick click: Rapid text copying based on automatic entity extraction. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 562–567, New York, NY, USA. ACM.
- Billman, D. and Bier, E. A. (2007). *Medical Sensemaking with Entity Workspace*, page 229–232. Association for Computing Machinery, New York, NY, USA.

- Bondarenko, O. and Janssen, R. (2005). Documents at hand: Learning from paper to improve digital technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, page 121–130, New York, NY, USA. Association for Computing Machinery.
- Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., and Klemmer, S. R. (2009). Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1589–1598, New York, NY, USA. ACM.
- Braun, V. and Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health*, 11(4):589–597.
- Bush, V. (1945). As We May Think. *Atlantic Monthly*, 176(1):641–649.
- Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. (1988). An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, page 95–100, New York, NY, USA. Association for Computing Machinery.
- Capra, R., Marchionini, G., Velasco-Martin, J., and Muller, K. (2010). *Tools-at-Hand and Learning in Multi-Session, Collaborative Search*, page 951–960. Association for Computing Machinery, New York, NY, USA.
- Card, S. K., Moran, T. P., and Newell, A. (1987). Computer text-editing: An information-processing analysis of a routine cognitive skill. In Baecker, R. M. and Buxton, W. A. S., editors, *Human-computer Interaction*, chapter Computer Text-editing: An Information-processing Analysis of a Routine Cognitive Skill, pages 219–240. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Card, S. K., Robertson, G. G., and York, W. (1996). The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, page 111–ff., New York, NY, USA. Association for Computing Machinery.
- Chang, J. C., Hahn, N., and Kittur, A. (2020). Mesh: Scaffolding comparison tables for online decision making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 391–405, New York, NY, USA. Association for Computing Machinery.
- Chang, J. C., Hahn, N., Perer, A., and Kittur, A. (2019). Searchlens: Composing and capturing complex user interests for exploratory search. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 498–509, New York, NY, USA. Association for Computing Machinery.

- Chapuis, O. and Roussel, N. (2007). Copy-and-paste between overlapping windows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 201–210, New York, NY, USA. ACM.
- Chattopadhyay, S., Prasad, I., Henley, A. Z., Sarma, A., and Barik, T. (2020). What's wrong with computational notebooks? pain points, needs, and design opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA. Association for Computing Machinery.
- Cheney, J., Chong, S., Foster, N., Seltzer, M., and Vansummeren, S. (2009). Provenance: A future history. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, page 957–964, New York, NY, USA. Association for Computing Machinery.
- Cherubini, M., Venolia, G., DeLine, R., and Ko, A. J. (2007). Let's go to the whiteboard: How and why software developers use drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 557–566, New York, NY, USA. ACM.
- Churchill, E. F., Trevor, J., Bly, S., Nelson, L., and Cubranic, D. (2000). Anchored conversations: Chatting in the context of a document. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 454–461, New York, NY, USA. ACM.
- Cohen, A. L., Cash, D., and Muller, M. J. (1999). Awareness, planning and joint attention in collaborative writing: From fieldwork to design. *LOTUS CODE# 1999.02*, pages 94–101.
- Conlen, M. and Heer, J. (2018). Idyll: A markup language for authoring and publishing interactive articles on the web. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 977–989, New York, NY, USA. Association for Computing Machinery.
- Cooper, H. M. (1998). *Synthesizing research: A guide for literature reviews*, volume 2. Sage.
- De Saussure, F. (2011). *Course in general linguistics*. Columbia University Press.
- Dearman, D. and Pierce, J. S. (2008). It's on my other computer! computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 767–776, New York, NY, USA. Association for Computing Machinery.
- Dontcheva, M., Drucker, S. M., Wade, G., Salesin, D., and Cohen, M. F.

- (2006). Summarizing personal web browsing sessions. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, page 115–124, New York, NY, USA. Association for Computing Machinery.
- Dourish, P., Edwards, W. K., LaMarca, A., and Salisbury, M. (1999). Presto: An experimental architecture for fluid interactive document spaces. *ACM Trans. Comput.-Hum. Interact.*, 6(2):133–161.
- Dragicevic, P., Jansen, Y., Sarma, A., Kay, M., and Chevalier, F. (2019). Increasing the transparency of research papers with explorable multiverse analyses. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 65:1–65:15, New York, NY, USA. ACM.
- Dubroy, P. and Balakrishnan, R. (2010). A study of tabbed browsing among mozilla firefox users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 673–682, New York, NY, USA. Association for Computing Machinery.
- Ducheneaut, N. and Bellotti, V. (2001). E-mail as habitat: An exploration of embedded personal information management. *Interactions*, 8(5):30–38.
- Evans, N., Edge, D., Larson, J., and White, C. (2020). News provenance: Revealing news text reuse at web-scale in an augmented news search experience. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–8, New York, NY, USA. Association for Computing Machinery.
- Farkas, D. K. (1985). The concept of consistency in writing and editing. *Journal of Technical Writing and Communication*, 15(4):353–364.
- Federico, P., Heimerl, F., Koch, S., and Miksch, S. (2017). A survey on visual approaches for analyzing scientific literature and patents. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2179–2198.
- Fertig, S., Freeman, E., and Gelernter, D. (1996). Lifestreams: An alternative to the desktop metaphor. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, page 410–411, New York, NY, USA. Association for Computing Machinery.
- Flintham, M., Karner, C., Bachour, K., Creswick, H., Gupta, N., and Moran, S. (2018). Falling for fake news: Investigating the consumption of news via social media. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–10, New York, NY, USA. Association for Computing Machinery.
- Fonteyn, M. E., Kuipers, B., and Grobe, S. J. (1993). A description of think aloud method and protocol analysis. *Qualitative Health Research*, 3(4):430–441.
- Garcia, J., Tsandilas, T., Agon, C., and Mackay, W. E. (2014). Structured

- observation with polyphony: A multifaceted tool for studying music composition. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, page 199–208, New York, NY, USA. Association for Computing Machinery.
- Gaver, W. W. (1989). The sonicfinder: An interface that uses auditory icons. *Hum.-Comput. Interact.*, 4(1):67–94.
- Ghorashi, S. and Jensen, C. (2012). Leyline: Provenance-based search using a graphical sketchpad. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*, HCIR '12, New York, NY, USA. Association for Computing Machinery.
- Goffin, P., Boy, J., Willett, W., and Isenberg, P. (2017). An exploratory study of word-scale graphics in data-rich text documents. *IEEE Transactions on Visualization and Computer Graphics*, 23(10):2275–2287.
- Golovchinsky, G. (2008). Reading in the office. In *Proceedings of the 2008 ACM Workshop on Research Advances in Large Digital Book Repositories*, BooksOnline '08, page 21–24, New York, NY, USA. Association for Computing Machinery.
- Golovchinsky, G., Price, M. N., and Schilit, B. N. (1999). From reading to retrieval: Freeform ink annotations as queries. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 19–25, New York, NY, USA. Association for Computing Machinery.
- Gori, J., Han, H. L., and Beaudouin-Lafon, M. (2020). Fileweaver: Flexible file management with automatic dependency tracking. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 22–34, New York, NY, USA. Association for Computing Machinery.
- Gotz, D. (2007). The scratchpad: Sensemaking support for the web. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 1329–1330, New York, NY, USA. Association for Computing Machinery.
- Green, T. R. G. (1990). Cognitive dimensions of notations. In *Proceedings of the Fifth Conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and Computers V*, page 443–460, USA. Cambridge University Press.
- Grossman, T. and Balakrishnan, R. (2005). *The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area*, page 281–290. Association for Computing Machinery, New York, NY, USA.
- Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. (2004). Object pointing: A complement to bitmap pointing in guis. In *Proceedings of Graphics Interface 2004*, GI '04, page 9–16, Waterloo, CAN. Canadian Human-Computer Communications Society.

- Guo, P. J. (2012). *Software tools to facilitate research programming*. PhD thesis, Stanford University Stanford, CA.
- Guo, P. J. and Seltzer, M. (2012). Burrito: Wrapping your lab notebook in computational infrastructure. In *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance, TaPP'12*, page 7, Berkeley, CA, USA. USENIX Association.
- Halasz, F. G., Moran, T. P., and Trigg, R. H. (1986). Notecards in a nutshell. *SIGCHI Bull.*, 17(SI):45–52.
- Han, H. L., Junhang., Y., Ciorascu., A., Bournet., R., Mackay, W. E., and Beaudouin-Lafon, M. (2022). *Passages: Reading and Interacting with Text Across Documents*. Association for Computing Machinery, New York, NY, USA.
- Han, H. L., Renom, M. A., Mackay, W. E., and Beaudouin-Lafon, M. (2020). Textlets: Supporting constraints and consistency in text documents. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA. Association for Computing Machinery.
- Harper, R., Lindley, S., Thereska, E., Banks, R., Gosset, P., Smyth, G., Odom, W., and Whitworth, E. (2013). What is a file? In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13*, page 1125–1136, New York, NY, USA. Association for Computing Machinery.
- Hartmann, B., MacDougall, D., Brandt, J., and Klemmer, S. R. (2010). What would other programmers do: Suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10*, pages 1019–1028, New York, NY, USA. ACM.
- Head, A., Hohman, F., Barik, T., Drucker, S. M., and DeLine, R. (2019). Managing messes in computational notebooks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19*, New York, NY, USA. Association for Computing Machinery.
- Hearst, M. A. (1995). Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95*, page 59–66, USA. ACM Press/Addison-Wesley Publishing Co.
- Hearst, M. A. (2009). *Search User Interfaces*. Cambridge University Press.
- Hertzum, M. and Pejtersen, A. M. (2000). The information-seeking practices of engineers: Searching for documents as well as for people. *Inf. Process. Manage.*, 36(5):761–778.
- Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T. (1992). Edit wear and read wear. In *Proceedings of the SIGCHI Conference*

- on *Human Factors in Computing Systems*, CHI '92, pages 3–9, New York, NY, USA. ACM.
- Hinckley, K., Bi, X., Pahud, M., and Buxton, B. (2012). Informal information gathering techniques for active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 1893–1896, New York, NY, USA. Association for Computing Machinery.
- Hinckley, K., Zhao, S., Sarin, R., Baudisch, P., Cutrell, E., Shilman, M., and Tan, D. (2007). Inkseine: In situ search for active note taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 251–260, New York, NY, USA. Association for Computing Machinery.
- Hollan, J. and Stornetta, S. (1992). Beyond being there. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, page 119–125, New York, NY, USA. Association for Computing Machinery.
- Hong, L., Chi, E. H., Budiu, R., Pirolli, P., and Nelson, L. (2008). Spartag.us: A low cost tagging system for foraging of web content. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, page 65–72, New York, NY, USA. Association for Computing Machinery.
- Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 256–265, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Hutchins, E. L., Hollan, J. D., and Norman, D. A. (1985). Direct manipulation interfaces. *Hum.-Comput. Interact.*, 1(4):311–338.
- Hutchinson, H., Mackay, W., Westerlund, B., Bederson, B. B., Druin, A., Plaisant, C., Beaudouin-Lafon, M., Conversy, S., Evans, H., Hansen, H., Roussel, N., and Eiderbäck, B. (2003). Technology probes: Inspiring design for and with families. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 17–24, New York, NY, USA. Association for Computing Machinery.
- Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., and Zigelbaum, J. (2008). Reality-based interaction: A framework for post-wimp interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 201–210, New York, NY, USA. Association for Computing Machinery.
- Jalal, G., Maudet, N., and Mackay, W. E. (2015). Color portraits: From color picking to interacting with color. In *Proceedings of the 33rd*

- Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 4207–4216, New York, NY, USA. ACM.
- Jensen, C., Lonsdale, H., Wynn, E., Cao, J., Slater, M., and Dietterich, T. G. (2010). *The Life and Times of Files and Information: A Study of Desktop Provenance*, page 767–776. Association for Computing Machinery, New York, NY, USA.
- Johnson, J., Roberts, T. L., Verplank, W., Smith, D. C., Irby, C. H., Beard, M., and Mackey, K. (1989). The Xerox Star: A retrospective. *Computer*, 22(9):11–26, 28–29.
- Jung, H., Stolterman, E., Ryan, W., Thompson, T., and Siegel, M. (2008). Toward a framework for ecologies of artifacts: How are digital artifacts interconnected within a personal life? In *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*, NordiCHI '08, page 201–210, New York, NY, USA. Association for Computing Machinery.
- Kabbash, P. and Buxton, W. A. S. (1995). The “prince” technique: Fitts’ law and selection using area cursors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 273–279, USA. ACM Press/Addison-Wesley Publishing Co.
- Kapser, C. J. and Godfrey, M. W. (2008). “cloning considered harmful” considered harmful: Patterns of cloning in software. *Empirical Softw. Engg.*, 13(6):645–692.
- Karger, D. R. and Jones, W. (2006). Data unification in personal information management. *Communications of the ACM*, 49(1):77–82.
- Karlson, A. K., Smith, G., and Lee, B. (2011). Which version is this? improving the desktop experience within a copy-aware computing ecosystem. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 2669–2678, New York, NY, USA. Association for Computing Machinery.
- Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S., and Fitzmaurice, G. (2014). Draco: Bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 351–360, New York, NY, USA. Association for Computing Machinery.
- Kery, M. B., Horvath, A., and Myers, B. (2017). Variolite: Supporting exploratory programming by data scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1265–1276, New York, NY, USA. Association for Computing Machinery.
- Kery, M. B., John, B. E., O’Flaherty, P., Horvath, A., and Myers, B. A. (2019). Towards effective foraging by data scientists to find past analysis choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA. Association for Computing Machinery.

- Kery, M. B., Radensky, M., Arya, M., John, B. E., and Myers, B. A. (2018). The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA. Association for Computing Machinery.
- Kidd, A. (1994). The marks are on the knowledge worker. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 186–191, New York, NY, USA. Association for Computing Machinery.
- Kim, M., Bergman, L., Lau, T., and Notkin, D. (2004). An ethnographic study of copy and paste programming practices in oopl. In *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, ISESE '04, pages 83–92, Washington, DC, USA. IEEE Computer Society.
- Kittur, A., Peters, A. M., Diriyee, A., Telang, T., and Bove, M. R. (2013). Costs and benefits of structured information foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 2989–2998, New York, NY, USA. Association for Computing Machinery.
- Klemmer, S. R., Hartmann, B., and Takayama, L. (2006). How bodies matter: Five themes for interaction design. In *Proceedings of the 6th Conference on Designing Interactive Systems*, DIS '06, page 140–149, New York, NY, USA. Association for Computing Machinery.
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and Team, J. D. (2016). Jupyter notebooks - a publishing format for reproducible computational workflows. In *International Conference on Electronic Publishing*.
- Ko, A. J., Aung, H., and Myers, B. A. (2005). Eliciting design requirements for maintenance-oriented IDEs: A detailed study of corrective and perfective maintenance tasks. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 126–135, New York, NY, USA. ACM.
- Ko, A. J. and Myers, B. A. (2004). Designing the whyline: A debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 151–158, New York, NY, USA. ACM.
- Ko, A. J. and Myers, B. A. (2006). Barista: An implementation framework for enabling new tools, interaction techniques and views in code editors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 387–396, New York, NY, USA. ACM.

- Koch, J., Taffin, N., Beaudouin-Lafon, M., Laine, M., Lucero, A., and Mackay, W. E. (2020). Imagesense: An intelligent collaborative ideation tool to support diverse human-computer partnerships. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW1).
- Laakso, S. A., Laakso, K. P., and Saura, A. J. (2000). Improved scroll bars. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 97–98, New York, NY, USA. ACM.
- Larsen-Ledet, I. and Korsgaard, H. (2019). Territorial functioning in collaborative writing. *Computer Supported Cooperative Work (CSCW)*, 28(3):391–433.
- Lindley, S. E., Smyth, G., Corish, R., Loukianov, A., Golembewski, M., Luger, E. A., and Sellen, A. (2018). Exploring new metaphors for a networked world through the file biography. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA. Association for Computing Machinery.
- Liu, M. X., Hsieh, J., Hahn, N., Zhou, A., Deng, E., Burley, S., Taylor, C., Kittur, A., and Myers, B. A. (2019). Unakite: Scaffolding developers' decision-making using the web. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 67–80, New York, NY, USA. Association for Computing Machinery.
- Liu, W., Rioul, O., McGrenere, J., Mackay, W. E., and Beaudouin-Lafon, M. (2018). Bigfile: Bayesian information gain for fast file retrieval. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA. Association for Computing Machinery.
- Lottridge, D. and Mackay, W. E. (2009). Generative walkthroughs: To support creative redesign. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, page 175–184, New York, NY, USA. Association for Computing Machinery.
- Ma Kay, B. and Watters, C. (2008). Exploring multi-session web tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 1187–1196, New York, NY, USA. Association for Computing Machinery.
- Mackay, W. E. (1988). Diversity in the use of electronic mail: A preliminary inquiry. *ACM Trans. Inf. Syst.*, 6(4):380–397.
- Mackay, W. E. (1990). Patterns of sharing customizable software. In *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work*, CSCW '90, page 209–221, New York, NY, USA. Association for Computing Machinery.
- Mackay, W. E. (1999). Is paper safer? the role of paper flight strips in air traffic control. *ACM Trans. Comput.-Hum. Interact.*, 6(4):311–340.
- Mackay, W. E. (2002). Using video to support interaction design. *DVD*

Tutorial, CHI, 2(5).

- Mackay, W. E. (2004). The interactive thread: Exploring methods for multi-disciplinary design. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, page 103–112, New York, NY, USA. Association for Computing Machinery.
- Mackay, W. E. (2019). Designing with sticky notes. In Christensen, B., Halskov, K., and Klokmoose, C., editors, *Sticky Creativity: Post-It Note Cognition, Interaction and Digitalization*, pages 231–256. Academic Press.
- Mackay, W. E. and Fayard, A.-L. (1997). HCI, natural science and design: A framework for triangulation across disciplines. In *Proceedings of the 2nd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '97, page 223–234, New York, NY, USA. Association for Computing Machinery.
- Mackinlay, J. D., Rao, R., and Card, S. K. (1995). An organic user interface for searching citation links. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 67–73, USA. ACM Press/Addison-Wesley Publishing Co.
- Malone, T. W. (1983). How do people organize their desks? implications for the design of office information systems. *ACM Trans. Inf. Syst.*, 1(1):99–112.
- Marshall, C. C. (1997). Annotation: From paper books to the digital library. In *Proceedings of the Second ACM International Conference on Digital Libraries*, DL '97, pages 131–140, New York, NY, USA. ACM.
- Marshall, C. C. and Bly, S. (2005). Saving and using encountered information: Implications for electronic periodicals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 111–120, New York, NY, USA. Association for Computing Machinery.
- Marshall, C. C., Halasz, F. G., Rogers, R. A., and Janssen, W. C. (1991). Aquanet: A hypertext tool to hold your knowledge in place. In *Proceedings of the Third Annual ACM Conference on Hypertext*, HYPERTEXT '91, page 261–275, New York, NY, USA. Association for Computing Machinery.
- Marshall, C. C., Price, M. N., Golovchinsky, G., and Schilit, B. N. (2001). Designing e-books for legal research. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '01, page 41–48, New York, NY, USA. Association for Computing Machinery.
- McCloud, S. (1993). *Understanding Comics: The Invisible Art*. Tundra Publishing Ltd.
- McGrenere, J., Baecker, R. M., and Booth, K. S. (2002). An evaluation of

- a multiple interface design solution for bloated software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, page 164–170, New York, NY, USA. Association for Computing Machinery.
- Miller, R. C. and Marshall, A. M. (2004). Cluster-based find and replace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 57–64, New York, NY, USA. ACM.
- Miller, R. C. and Myers, B. A. (2002a). Lapis: Smart editing with text structure. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 496–497, New York, NY, USA. ACM.
- Miller, R. C. and Myers, B. A. (2002b). Multiple selections in smart text editing. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 103–110, New York, NY, USA. ACM.
- Muniswamy-Reddy, K.-K., Holland, D. A., Braun, U., and Seltzer, M. I. (2006). Provenance-aware storage systems. In *Usenix annual technical conference, general track*, pages 43–56, Berkeley, CA, USA. USENIX Association.
- Murta, L., Braganholo, V., Chirigati, F., Koop, D., and Freire, J. (2014). noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*, pages 71–83. Springer.
- Myers, B. A. (1991). Text formatting by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 251–256, New York, NY, USA. ACM.
- Noël, S. and Robert, J.-M. (2004). Empirical study on collaborative writing: What do co-authors do, use, and like? *Comput. Supported Coop. Work*, 13(1):63–89.
- Norman, D. A. (1991). Cognitive artifacts. *Designing interaction: Psychology at the human-computer interface*, 1(1):17–38.
- O'Hara, K. and Sellen, A. (1997). A comparison of reading paper and on-line documents. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 335–342, New York, NY, USA. ACM.
- O'Hara, K., Smith, F., Newman, W., and Sellen, A. (1998). Student readers' use of library documents: Implications for library technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, page 233–240, USA. ACM Press/Addison-Wesley Publishing Co.
- Oleksik, G., Jetter, H.-C., Gerken, J., Milic-Frayling, N., and Jones, R. (2013). Towards an information architecture for flexible reuse of digital media. In *Proceedings of the 12th International Conference on*

- Mobile and Ubiquitous Multimedia*, MUM '13, New York, NY, USA. Association for Computing Machinery.
- Oleksik, G., Milic-Frayling, N., and Jones, R. (2012). Beyond data sharing: Artifact ecology of a collaborative nanophotonics research centre. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 1165–1174, New York, NY, USA. Association for Computing Machinery.
- Oleksik, G., Wilson, M. L., Tashman, C., Mendes Rodrigues, E., Kazai, G., Smyth, G., Milic-Frayling, N., and Jones, R. (2009). Lightweight tagging expands information and activity management practices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 279–288, New York, NY, USA. Association for Computing Machinery.
- Oney, S. and Brandt, J. (2012). Codelets: Linking interactive documentation and example code in the editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2697–2706, New York, NY, USA. ACM.
- O'Hara, K. and Sellen, A. (1997). A comparison of reading paper and on-line documents. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, page 335–342, New York, NY, USA. Association for Computing Machinery.
- Pedersen, E. R., McCall, K., Moran, T. P., and Halasz, F. G. (1993). Tivoli: An electronic whiteboard for informal workgroup meetings. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, page 391–398, New York, NY, USA. Association for Computing Machinery.
- Perez De Rosso, S. and Jackson, D. (2013). What's wrong with git? a conceptual design analysis. In *Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*, Onward! 2013, page 37–52, New York, NY, USA. Association for Computing Machinery.
- Perlin, K. and Fox, D. (1993). Pad: An alternative approach to the computer interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 57–64, New York, NY, USA. Association for Computing Machinery.
- Pirolli, P. and Rao, R. (1996). Table lens as a tool for making sense of data. In *Proceedings of the Workshop on Advanced Visual Interfaces*, AVI '96, page 67–80, New York, NY, USA. Association for Computing Machinery.
- Pérez, B., Rubio, J., and Sáenz-Adán, C. (2018). A systematic review of provenance systems. *Knowledge and Information Systems*, 57.
- Rao, R., Pedersen, J. O., Hearst, M. A., Mackinlay, J. D., Card, S. K., Masinter, L., Halvorsen, P.-K., and Robertson, G. C. (1995). Rich

- interaction in the digital library. *Commun. ACM*, 38(4):29–39.
- Ravasio, P., Schär, S. G., and Krueger, H. (2004). In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact.*, 11(2):156–180.
- Rivière, J.-P., Alaoui, S. F., Caramiaux, B., and Mackay, W. E. (2019). Capturing movement decomposition to support learning and teaching in contemporary dance. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW).
- Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., and van Dantzich, M. (1998). Data mountain: Using spatial memory for document management. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, page 153–162, New York, NY, USA. Association for Computing Machinery.
- Rodden, K. and Wood, K. R. (2003). How do people manage their digital photographs? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 409–416, New York, NY, USA. Association for Computing Machinery.
- Romat, H., Pietriga, E., Henry-Riche, N., Hinckley, K., and Appert, C. (2019). Spaceink: Making space for in-context annotations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 871–882, New York, NY, USA. Association for Computing Machinery.
- Rosson, M. B. (1983). Patterns of experience in text editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '83, pages 171–175, New York, NY, USA. ACM.
- Rule, A., Drosos, I., Tabard, A., and Hollan, J. D. (2018a). Aiding collaborative reuse of computational notebooks with annotated cell folding. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW).
- Rule, A., Tabard, A., and Hollan, J. D. (2018b). Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA. Association for Computing Machinery.
- Russell, D. M., Stefik, M. J., Pirolli, P., and Card, S. K. (1993). The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, page 269–276, New York, NY, USA. Association for Computing Machinery.
- Schilit, B. N., Golovchinsky, G., and Price, M. N. (1998). Beyond paper: Supporting active reading with free form digital ink annotations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, page 249–256, USA. ACM Press/Addison-Wesley Publishing Co.
- schraefel, M. C., Zhu, Y., Modjeska, D., Wigdor, D., and Zhao, S.

- (2002). Hunter gatherer: Interaction support for the creation and management of within-web-page collections. In *Proceedings of the 11th International Conference on World Wide Web, WWW '02*, page 172–181, New York, NY, USA. Association for Computing Machinery.
- Sellen, A. and Harper, R. (1997). Paper as an analytic resource for the design of new technologies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '97*, page 319–326, New York, NY, USA. Association for Computing Machinery.
- Sellen, A. J. and Harper, R. H. (2003). *The Myth of the Paperless Office*. MIT Press, Cambridge, MA, USA.
- Sellen, A. J., Murphy, R., and Shaw, K. L. (2002). How knowledge workers use the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '02*, page 227–234, New York, NY, USA. Association for Computing Machinery.
- Shen, H. (2014). Interactive notebooks: Sharing the code. *Nature*, 515(7525):151–152.
- Shipman, F. M., Marshall, C. C., and Moran, T. P. (1995). Finding and using implicit structure in human-organized spatial layouts of information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95*, page 346–353, USA. ACM Press/Addison-Wesley Publishing Co.
- Shneiderman (1983). Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69.
- Simon, H. A. (1996). *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 3 edition.
- Smith, D. C., Irby, C., Kimball, R., and Harslem, E. (1982). The Star user interface: An overview. In *Proceedings of the June 7–10, 1982, National Computer Conference, AFIPS '82*, page 515–528, New York, NY, USA. Association for Computing Machinery.
- Soules, C. A. N. and Ganger, G. R. (2005). Connections: Using context to enhance file search. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles, SOSP '05*, page 119–132, New York, NY, USA. Association for Computing Machinery.
- Srgaard, P. and Sandahl, T. I. (1997). Problems with styles inword processing: Aweak foundation for electronic publishing with sgml. In *Proceedings of the 30th Hawaii International Conference on System Sciences: Digital Documents - Volume 6, HICSS '97*, pages 137–, Washington, DC, USA. IEEE Computer Society.
- Starbird, K., Dailey, D., Mohamed, O., Lee, G., and Spiro, E. S. (2018). Engage early, correct more: How journalists participate in false rumors online during crisis events. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–12,

- New York, NY, USA. Association for Computing Machinery.
- Stumpf, S., Fitzhenry, E., and Dietterich, T. G. (2007). The use of provenance in information retrieval. In *Workshop on Principles of Provenance (PROPR)*.
- Stylos, J., Myers, B. A., and Faulring, A. (2004). Citrine: Providing intelligent copy-and-paste. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, UIST '04*, pages 185–188, New York, NY, USA. ACM.
- Subramonyam, H., Seifert, C., Shah, P., and Adar, E. (2020). Texsketch: Active diagramming through pen-and-ink annotations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA. Association for Computing Machinery.
- Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication*. Cambridge university press.
- Tabard, A., Mackay, W., Roussel, N., and Letondal, C. (2007). Pagelinker: Integrating contextual bookmarks within a browser. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07*, page 337–346, New York, NY, USA. Association for Computing Machinery.
- Tabard, A., Mackay, W. E., and Eastmond, E. (2008). From individual to collaborative: The evolution of prism, a hybrid laboratory notebook. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work, CSCW '08*, page 569–578, New York, NY, USA. Association for Computing Machinery.
- Tang, A., Lanir, J., Greenberg, S., and Fels, S. (2009). Supporting transitions in work: Informing large display application design by understanding whiteboard use. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work, GROUP '09*, page 149–158, New York, NY, USA. Association for Computing Machinery.
- Tashman, C. S. and Edwards, W. K. (2011a). Active reading and its discontents: The situations, problems and ideas of readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 2927–2936, New York, NY, USA. Association for Computing Machinery.
- Tashman, C. S. and Edwards, W. K. (2011b). Liquidtext: A flexible, multitouch environment to support active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 3285–3294, New York, NY, USA. Association for Computing Machinery.
- Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. (2004). The perfect search engine is not enough: A study of orienteering behavior in directed search. In *Proceedings of the SIGCHI Conference*

- on *Human Factors in Computing Systems*, CHI '04, page 415–422, New York, NY, USA. Association for Computing Machinery.
- Teevan, J., Iqbal, S. T., and von Veh, C. (2016). Supporting collaborative writing with microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2657–2668, New York, NY, USA. ACM.
- Tesler, L. (2012). A personal history of modeless text editing and cut/copy-paste. *Interactions*, 19(4):70–75.
- Toomim, M., Begel, A., and Graham, S. L. (2004). Managing duplicated code with linked editing. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, VLHCC '04, pages 173–180, Washington, DC, USA. IEEE Computer Society.
- Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 2 edition.
- Tversky, B. (2015). The cognitive design of tools of thought. *Review of Philosophy and Psychology*, 6:99–116.
- Tyler, S. W., Roth, S., and Post, T. (1982). The acquisition of text editing skills. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, CHI '82, pages 324–325, New York, NY, USA. ACM.
- van Dam, A. (1997). Post-wimp user interfaces. *Commun. ACM*, 40(2):63–67.
- Voida, S. and Mynatt, E. D. (2009). It feels better than filing: Everyday work experiences in an activity-based computing system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, page 259–268, New York, NY, USA. Association for Computing Machinery.
- von Hippel, E. (1986). Lead users: A source of novel product concepts. *Manage. Sci.*, 32(7):791–805.
- Wagner, J., Mackay, W. E., and Huot, S. (2012). Left-over windows cause window clutter... but what causes left-over windows? In *Proceedings of the 2012 Conference on Ergonomie et Interaction Homme-Machine*, Ergo'IHM '12, page 213–216, New York, NY, USA. Association for Computing Machinery.
- Weinreich, H., Obendorf, H., Herder, E., and Mayer, M. (2008). Not quite the average: An empirical study of web use. *ACM Trans. Web*, 2(1).
- Wexelblat, A. and Maes, P. (1999). Footprints: History-rich tools for information foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 270–277, New York, NY, USA. ACM.
- White, R. W. and Roth, R. A. (2009). *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool Publishers, [San Rafael, Calif.].

- Whittaker, S. and Sidner, C. (1996). Email overload: Exploring personal information management of email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 276–283, New York, NY, USA. Association for Computing Machinery.
- Woodruff, A., Faulring, A., Rosenholtz, R., Morrisson, J., and Pirolli, P. (2001). Using thumbnails to search the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 198–205, New York, NY, USA. Association for Computing Machinery.
- Worden, A., Walker, N., Bharat, K., and Hudson, S. (1997). Making computers easier for older adults to use: Area cursors and sticky icons. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, page 266–271, New York, NY, USA. Association for Computing Machinery.
- Xia, H., Henry Riche, N., Chevalier, F., De Araujo, B., and Wigdor, D. (2018). *DataInk: Direct and Creative Data-Oriented Drawing*, page 1–13. Association for Computing Machinery, New York, NY, USA.
- Yeh, R., Liao, C., Klemmer, S., Guimbretière, F., Lee, B., Kakaradov, B., Stamberger, J., and Paepcke, A. (2006). Butterflynet: A mobile capture and access system for field biology research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 571–580, New York, NY, USA. Association for Computing Machinery.
- Yoon, D., Chen, N., and Guimbretière, F. (2013). Texttearing: Opening white space for digital ink annotation. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 107–112, New York, NY, USA. ACM.
- Yoon, D., Chen, N., Guimbretière, F., and Sellen, A. (2014). Richreview: Blending ink, speech, and gesture to support collaborative document review. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 481–490, New York, NY, USA. Association for Computing Machinery.
- Yoon, Y. S. and Myers, B. A. (2015). Supporting selective undo in a code editor. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 223–233, Piscataway, NJ, USA. IEEE Press.
- Zhai, S., Buxton, W., and Milgram, P. (1994). The “silk cursor”: Investigating transparency for 3d target acquisition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 459–464, New York, NY, USA. Association for Computing Machinery.
- Zhang, A. X., Muller, M., and Wang, D. (2020). How do data science workers collaborate? roles, workflows, and tools. *Proc. ACM*

Hum.-Comput. Interact., 4(CSCW1).

- Zhang, X., Qu, Y., Giles, C. L., and Song, P. (2008). Citesense: Supporting sensemaking of research literature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08*, page 677–680, New York, NY, USA. Association for Computing Machinery.
- Zheng, Q., Booth, K., and McGrenere, J. (2006). Co-authoring with structured annotations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06*, pages 131–140, New York, NY, USA. ACM.