Kim Kleinert

Sharing Context

Metadata, Markup Language and Articulation Work

Thesis submitted to: the Department of Experimental Publishing, Piet Zwart Institute, Willem de Kooning Academy
in partial fulfillment of the requirements for the final examination for the degree of: Master of Arts in Fine Art & Design: Experimental Publishing.

Adviser: Marloes de Valk
Second Reader: Michael Murtaugh
Word count: 8644

Index

Introduction

This thesis examines the relationship between metadata and markup language, and how they have become formalized through the introduction of computers and the internet. We use metadata to index and retrieve data. Markup language turns data or text into structured information, rendering it legible to coworkers or machines, for storage and reproduction. In that, markup language is commonly perceived as interchange format, based on the assumption that its structure has already been written, and that it merely needs to be (re-)expressed or completed for purposes of transmission. Throughout this thesis I want to confront this notion and instead regard markup as writing practice, emphasizing its capacities to actively shape text production [1] and exchange. My inquiry here turns to collaborative work in the production of publications: How is it affected by the transformation of markup and metadata? And what does collaborative work entail, specifically what forms of coordination does it require?

[1] I use the notion of 'text'-, and 'publication production' to refer to acts of writing and assembling content manually or by using a text editor, but also include processes of rendering digital documents such as websites by a browser. Bibliographical studies, especially the work of scholar Donald McKenzie provide a useful understanding for this continuity, in that they acknowledge production as crucial to the publishing process, involving technical and material aspects next to writing work (McKenzie, 2002, 2009).

As part of a practice based research, this thesis analyzes how software coordinates collaborative work through the example of using Git as publishing protocol. Git is a software commonly used by programmers. It helps users to keep track of changes made in a file and lets them collaborate with others on the same files. The software's means of collaboration and distributed version control is enabled by a vast amount of metadata to the files contained inside a Git repository. Git builds the basis for GitPub, a tool I am developing alongside and through this research. GitPub, rendering a Git repository including its metadata into a printed or web based publication, is both subject of this text and methodological framework for producing it.

Today, collaborative writing and publishing increasingly takes place on cloud based, corporation owned frameworks and platforms which are developed under the pretense of smooth communication and efficient completion of tasks.
From my personal practice and perspective, situated between Free / Libre Open Source Software (FLOSS), feminist sever-, and permacomputing principles that consider alternative modes of sharing, maintaining, and relating to technology, this is a concerning development. Platforms and frameworks are built upon vast infrastructures, that not

only enforce extractivism and consume high amounts of energy. Through integrating them in their software, our means of production and communication are at risk of becoming subject to tech industries.

By providing insight into recent developments of metadata, this text contributes to existing knowledge of collaborative publishing work, as studied in CSCW and Bibliography and practiced by formal and informal groups. Metadata not only increases dependencies and abstraction but fundamentally shifts coordination of collaborative work by fully automizing it. The research of this thesis is of significance to practitioners who use software to write and publish collaboratively in fields related to code, arts, and design. Additionally, the here discussed findings are relevant to those who develop software for collaborative work.

This thesis is composed of three main chapters: ‘sharing context’ establishes the basis of this text, an understanding of markup and metadata as shared context. In here I further examine their historical and current uses in order to provide insight to the formalization of markup and metadata. The second chapter ‘shifting context’ analyzes the formalization processes more in depth, specifically focusing on its implications for collaborative work. I conclude this thesis with ‘articulating context’, where I discuss the dissolution of shared context today and suggest GitPub for its explicit articulation of collaborative processes through making metadata visible.

As a vast and rapidly developing field, establishing a comprehensive overview of metadata is beyond the scope of this thesis. This text will not provide an in depth discussion of personal metadata and matters of surveillance, neither will it explicitly engage with metadata in terms of machine learning.

Sharing Context

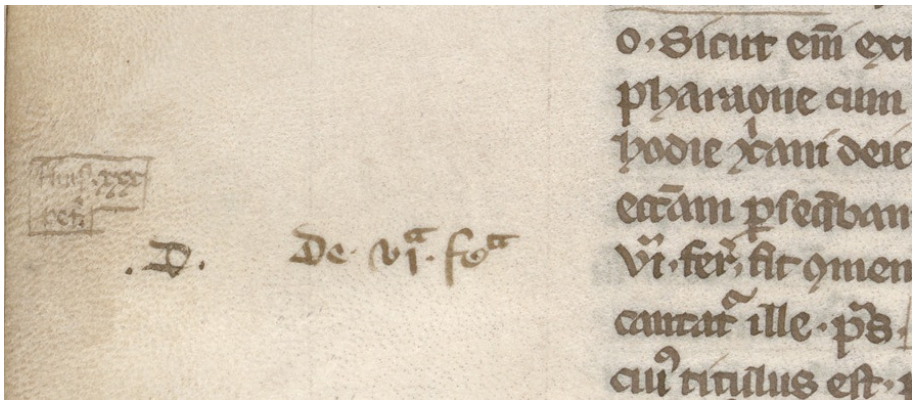
I will begin this chapter with a more detailed definition of markup and metadata, which provides the basis for understanding how they function and relate to another. Departing from this understanding, the second part of this chapter analyses the formalization of markup and metadata by looking into their origins and current uses.

Defining Markup Language

Markup languages structure text through opening and closing tags or designated characters which are wrapped around the documents contents. The tags work like instructions, indicating how the content, that they enclose, should be interpreted. Markup language is written in a source file, such as a Hypertext Markup Language (HTML) document, and rendered into a print file (or view), like a website or PDF.
`(aa768e1 - 2026-04-05 10:50:24+02:00 - Kim Kleinert - Writing this thesis in a source file, I can focus on the texts content, while leaving its presentation open to different formats. Source files can also easily be shared for editing or publishing (Coombs, DeRose and Renear, 1987).)`

The pecia system, as early form of markup language, enabled the collaborative production of a text where the pecia marks left in a document posed a means of communication between different workers. [2] The term markup is derived from the traditional publishing practice of marking up manuscripts by hand, including instructions for printing, or editorial notes (Harper, 2026). Pecia marks however are distinct from earlier occurring annotations in manuscripts. As ‘operational marks’ they are an act of logical structuring, whereas annotations are concerned with inserting content or instructions for visual appearance such as marks for illuminators indicating placement of illustrations.

[2] Until the 13th century, manuscript transcription was closely tied to monasteries, where one monk was working on one text at a time. To meet the increasing demand for manuscripts with the spread of literacy outside of the church, the pecia system was developed. Rationalizing text reproduction processes, manuscripts would be split in multiple parts called Peciae and could therefore be transcribed by multiple copiers at the same time. To communicate with another, ensuring for example the correct order of Peciae, the copiers left so called pecia marks in the manuscript they transcribed.



A pecia mark in the margin of a manuscript. Often containing the letter ‘p’, followed by numerals for the scribe to keep track of individual peciae during the rental process (Ray, 2015).

The introduction of technologies like the pecia system did not only make text reproduction more efficient but constituted another significant shift: a separation of skill and division of labor. What was previously the task of one scribe, who was responsible for an entire text, has now been split up between multiple copiers.
Further, the pecia system ensured, that the fidelity of a text was preserved when being copied in a distributed manner. The determining and repeatedly loaning out of one ‘original’ exemplar, resembling what we today know as source file, prevented the propagation of errors.
With the introduction of the printing press in the 15th century, mechanical reproduction of text moved into print shops and the pecia system was gradually replaced by so called ‘signatures’ (Febvre and Martin, 1976). [3]

[3] Visually resembling pagination methods we know today, signatures were not a service to the reader but ensured the correct order of printed pages for binding.

Like their analog predecessors, computer markup languages still facilitate communication and provide instructions between parties of production. However, not only between different workers, but human or computer and another computer.
The first computers used markup languages in the 1960 (RUNOFF at MIT) and it became standardized in the late 1980s. Emphasizing digital documents longevity, Standard Generalized Markup Language (SGML) enabled their formalized exchange, storage, and retrieval across time and different people.

Defining Metadata

Metadata, as ‘data about data’ (ISO/IEC JTC 1, 2015), functions as a map or model that is used to index objects contained in a (symbolic) space: a library catalog is a separate document, listing and locating books contained in the collection, a map provides an overview of a landscape and specific parameters, places and routes within it. But just like the map that is not the territory it represents, metadata, with what it includes, where it draws it borders and what language it articulates itself in, is a subjective statement.

(d201a74 -2026-01-17 18:07:55+01:00 - Kim Kleinert - While writing this section about metadata, I am also manually adding and programmatically generating metadata. How so? This text is written in a git repository. In case this hasn’t come up yet, git is a software that helps you keep track of changes made in a file and lets you collaborate with others on the same files. The tracking of changes is also called version control. Git adds metadata, such as unique identifiers, to each version to store and retrieve it. Git metadata is not only crucial to version control but also for collaboration: next to unique identifiers, changes are provided with author name, timestamp and a so called commit message in which the author making the change describes what was changed. You are reading a commit message right now.) [4]

Metadata might be generated or added manually. It usually follows a schema that defines how the subject or resource and its relations are described in terminology and structure. Dublin core is such a schema, developed for describing web content in the beginnings of the Internet.

[4] the thesis Git repository can be found here: git-lab.com/km_kt/thesis

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description>
    <dc:title>Homage to Catalonia,</dc:title>
    <dc:creator>Orwell, George,1903-1950.</dc:creator>
    <dc:type>text</dc:type>
    <dc:publisher>London, Secker and Warburg</dc:publisher>
    <dc:date>[1938]</dc:date>
    <dc:language>eng</dc:language>
  </rdf:Description>
</rdf:RDF>
```

Example of a Dublin Core record encoded in XML.

Metadata can be classified into distinct types (Gartner, 2016), of which I understand the following three as most relevant to this text. Descriptive metadata is used for discovery and identification of resources, it includes elements such as title, author, and keywords of a resource. Administrative metadata provides information about maintenance and origin of an object. Lastly, structural metadata defines the framework that joins individual components into a object which conveys meaning to a user. A book's index, for instance, by providing an ordered list of chapter titles and their page numbers, renders the books individual parts comprehensible as a whole to its reader. Other types are more specifically concerned with personal data, such as use metadata which traces a users movements across digital services (Pomerantz, 2015).

The origins of metadata can be traced back to the Great Library of Alexandria where small tags attached to scrolls indicated title, subject, and author of the work. To further advance the organisation of the library, the Pinakes was composed which we today understand as the first library catalog (Phillips, 2010). Library metadata stayed in its bound form until the 1790s, which is when the first card cataloging systems appeared across Europe (Nix, 2009; Österreichische Nationalbibliothek, 2022). The new found modular approach improved the modification process of individual entries and enabled the administration of larger collections. The term metadata however was

only coined around 1968, when computers were starting to not only process figures but text documents. One of the first systems incorporating digital metadata was Machine Readable Cataloging (MARC), developed and standardized by the US library of congress. In this first digital appearance, metadata remained closely connected to libraries, but this should change with the expansion of the worldwide web.

Leader/00-23	*****nam##22*****#a#4500
001	<control number>
003	<control number identifier>
005	19920331092212.7
007/00-01	ta
008/00-39	820305s1991####nyu##########001#0#eng##
020	##$a0845348116 :$c$29.95 (£19.50 U.K.)
020	##$a0845348205 (pbk.)
040	##$a[organization code]$c[organization code]
050	14$aPN1992.8.S4$bT47 1991
082	04$a791.45/75/0973$219
100	1#$aTerrace, Vincent,$d1948-
245	10$aFifty years of television :$ba guide to series and pilots, 1937-1988 /$cVincent Terrace.
246	1#$a50 years of television
260	##$aNew York :$bCornwall Books,$cc1991.
300	##$a864 p. ;$c24 cm.
500	##$aIncludes index.
650	#0$aTelevision pilot programs$zUnited States$vCatalogs.
650	#0$aTelevision serials$zUnited States$vCatalogs.

MARC tags mainly rely on digit numbers instead of text labels. The first digits of each records line indicate the classification field, the following digits provide subfields and formatting instructions (Library of Congress, 2009).

Initially, a browser was simply a tool to access a file on the internet. As the web grew in terms of web pages and amount of users, the necessity for methods of navigation became increasingly urgent. This is when the first search engines appeared, which, just like the first cataloging systems, utilized metadata to index and retrieve files (HTML documents) from a collection (the worldwide web). The above mentioned Dublin Core schema here paved the way for the HTML `<meta>` tag, and with that the success of early web search tools.

Interpreting the Index, Indexing the Interpretation

Throughout their technical examination we could see that supposedly metadata and markup serve different purposes: we use metadata for indexing and retrieval and markup languages to state intended interpretation of text across entities of humans and computers. I here propose an understanding of markup and metadata as a shared context [5], enabling collaborative production. This understanding blurs markup and metadata boundaries and distinct histories of origin, allows us to

[5] The notion of 'context' relates to both: physical and social environment but also the text that surrounds a word or phrase to give it meaning.

see them as interrelated: Metadata schemas and elements are often encoded in markup languages (to interpret the index) while markup tags might also contain metadata (to index the interpretation).

(8800cfc - 2026-02-28 11:42:07+01:00 - Kim Kleinert - You can already sense, that this is where things get a bit more complicated.)

Collaboration includes not only exchanging change information or work on a specific task, but as importantly the coordination of the work across different collaborators and their knowledges and environments. Etymologically, 'context' derives from the Latin contexo, which stands for weaving, joining, connection, or structure (Wiktionary Contributors, 2026). An understanding of markup and metadata as shared, interwoven structure that actively shapes collaborative production counters the common assumption of marks and data as a technical byproduct.

(61917b2 - 2026-03-28 21:29:53+01:00 - Kim Kleinert - I choose to focus on markup as part of coordination and articulation work, rather than mere interchange format. In that, I regard it as writing practice, which from an engineering perspective is not a given.)

Markup and metadata's origins and relation to another provide the necessary foundation for understanding how they have become formalized. In the following I will take into account their current uses to examine this process more closely.

Generating Exchange

Markdown, Extensible Markup Language (XML) and HTML as the most used markup languages today, all represent small dependencies, high resilience and compatibility and enable easy document exchange between collaborators and software.

(28f87fa - 2026-03-28 21:32:36+01:00 - Kim Kleinert - I am writing this text in markdown)

Markdown utilizes ASCII characters such as hashtags or underscores instead of tags, which makes text formatting more minimal compared to other markup languages and enhances readability of source files.

Two examples showing the formatting of a headline in markup languages. The first line uses Markdown (a hashtag and space character indicating a header) compared to the below HTML (the header text is wrapped in h1 tags).

XML, due to its many simpler alternatives, today mainly remains in use for software configuration files and the exchange and storage of data.
(cc0f5a8 - 2026-02-01 20:47:32+01:00 - Kim Kleinert - This is not only a commit message, but an in-text citation, technically similar other other references in this text, they are enabled by Citation Style Language (CSL). CSL is an XML based markup language that enables computer readable formatting of scholarly citation styles such as Harvard or APA. I modified the harvard referencing CSL file and can now cite this documents metadata.)

HTML is the basis of the web, and with that poses the most common application of markup language. In its beginning, the worldwide web was comprised of simple markup files on web servers that could be rendered by web browsers. This was a two dimensional process in which a single entity (the browser) is responsible for interpreting a single source (the markup file) from one specific location (the web server).
In the late 1990s the web gained wider popularity. A point of departure for this development was the legalization of commercial activity on the web in 1995. Simultaneously the availability of internet con-

nection in offices and individual households increased, as were computers becoming more accessible in price, size, and usability. On the technical side, the threshold for publishing on the web lowered and possibilities for user interaction increased, all enabled by dynamic generation and retrieval of content through server side scripting. While previously, a web browser was interpreting a markup file from a server, now this file does not exist in its static form anymore. Instead, HTML is often supplemented with or entirely generated by JavaScript components. They assemble the markup file on demand, which can then be rendered by the browser.

In order to assemble the markup file into a meaningful order from individual components, structural metadata is required.

Metadata Beyond the Index

Information and Library scientist Jeffrey Pomerantz notes that 'Metadata has become infrastructural, like the electrical grid or the highway system' (2015, 3). 'Infrastructural' suggests that metadata is more ubiquitous, yet opaque, than when the concept was first coined. For staying within the scope of this text I want to outline two tendencies I see in metadata's current use and purpose. Firstly, metadata becomes more fine grained and complex. The technical capability of computers and the complexity of digital documents has notably increased since 1968, when the term metadata was first described. This allows for much higher granularity of metadata: Instead of indexing full objects or documents, today each individual component of an object contains its own metadata. This high granularity tends to exceed human comprehension in its amount and level of abstraction, and reflects a shift towards being merely intended for programmatic extraction and processing. Collaborative real time editors and how they synchronize edits between different users illustrate this point clearly: Merge algorithms like Conflictfree Replicated Data Types (CRDTs) assign each character a unique ID to determine its position in a text (Gentle and Kleppmann, 2025). Consequently, text is stripped of its context and turned into a dataset with metadata generated at the granularity of a single keystroke. Remaining illegible to human collaborators, coordination of the writing process is left to the algorithm. [6]

Secondly, digital resources are increasingly stored in and rendered

[6] For a more detailed account of CRDTs I recommend reading the software's documentation website: crdt.tech and Aljoscha Meyer's 'Notes on CRTDs' (2026).

from a database architecture. This transformation was conceptualized by digital culture theorist Lev Manovich, who presents the database as prevalent form of cultural expression (2001). In this paradigm shift, database logic, which stores information in separate pieces to be assembled at runtime, succeeds narrative, in which elements exist by being fixed into a particular order.
This becomes apparent when looking at digital documents that today are not stored in one piece, but separate components, which can be patched together into flexible shapes on demand. Every item stored in a database architecture contains descriptive metadata to retrieve it and structural metadata to render it into a whole.
(1c84621 - 2026-01-10 09:05:07+01:00 - Kim Kleinert - In git, which I am writing this text in right now, versions of a file are not stored as a whole. Instead the changes made in a file are indexed. To retrieve a specific version, it will be reassembled from the indexed metadata using hash driven reference clusters.)

JavaScript frameworks like React are exemplary for this structural transformation. Frameworks enable programmers to build dynamic web applications that exceed the standard browser capabilities while guaranteeing a streamlined developer experience (MDN Contributors, 2025). React entirely omits markup in the production process of web based applications. I noted this earlier when describing the current state of HTML, which is now generated by JavaScript components. Instead of relying on HTML documents and browser native JavaScript for interaction, React dynamically constructs the document object model (DOM) from individual components containing functions and given parameters. It thereby creates a virtual DOM and ‘fiber tree’ (internal data structure) in which an algorithm renders the final HTML from metadata. Database architectures break with the tradition of text production processes as narrative weaving under the pretense of absolute flexibility.
I will come back to this. For now, an understanding of metadata’s high granularity and the database as key form of cultural expression enable us to recognize that today, metadata’s role exceeds its original indexing purposes.

Thus far, I have argued that markup and metadata form a shared con-

text that enables collaborative production of publications. Both, metadata, and markup languages transformed notably since their concepts were introduced and first digitized. Markup language is written less by humans but remains in use for certain applications. Metadata moves from mere indexing to taking over important parts in the rendering of text.

Those transformations do not exist in a technological vacuum but impact the work processes that they enable. If the introduction of markup language brought forth distributed modes of production, how does this form change under prevalence of metadata today?

Shifting Context

This chapter firstly outlines two consecutive shifts of markup and metadata's roles in collaborative production of text. The second part examines how the most recent transformation informs collaboration. I will study this question on the basis of technological dependencies, abstraction and lastly look more closely at collaborative work: what it entails and how it is mediated.

Linear to Distributed Production

The pecia system brought forth a separation of skill and division of labor into distributed tasks. This is not an individual event but, as physicist and activist Ursula Franklin reminds us, a technological and social development recurring throughout history, significantly impacting not only the way we work but our relation to technology and each other (1990).

Let us look again, at the current state of web writing. Formerly static documents, today many sites use frameworks, like REACT, which distribute the composition of HTML across both the server and each users web browser. Digital humanities scholar Helen Burgess notes that 'This movement, from presentation to dynamic generation and retrieval, echoes the medieval transition from religious manuscript production to secular copy production' (2010, 178). Both transitions bring forth distributed modes of production that involve several steps and entities, thereby opening up the shared context of collaborative production through new possibilities of dissemination. Server side scripting, enabled publishing on the web to become more accessible and interactive, which can be seen with the rise of blogs and social media platforms in the early 2000s (O'Reilly, 2005). In order to manage the new wide contexts, individual tasks become more narrow and prescriptive. In the pecia system, which Peciae could be transcribed and in what rhythm where prescribed by the stationer, who oversaw the renting and selection of available texts (Febvre and Martin, 1976).

Burgess argumentation and the history of operational marks identify markup as closely linked to the production of text. But taking into account the findings from the previous chapter: an increasing importance of metadata for the dynamic generation of text and documents in real

time, and the declining use of markup languages, makes apparent that the shift from linear to distributed production requires a reconsideration.

Distributed to Detached Production

Today we meet in Google Docs or Etherpads to work on shared documents, we navigate versions using Git and build websites upon databases and JavaScript frameworks: tools and platforms which's text processing mechanisms rely on metadata rather than markup languages. In this currently ongoing transformation, metadata exceeds its prior role for indexing and retrieval and has to be considered the new basis for publication production processes.

Metadata based production further separates what previously were distributed processes, enabled by markup, into detached entities. Fundamentally, markup language operates from within a document. Metadata, in comparison, as its prefix 'meta' for 'higher' or 'beyond' indicates (Harper, 2026), is separated from the data it refers to. The notion of detachment exceeds this structural observation: In the following I will discuss how metadata based production not only disengages collaborators from another, but through abstraction also separates the worker and their process from the object of work itself. This abstraction results from a complex interrelation of technological dependencies and a transition from human directed to computer optimized legibility.

Deeply Dependent, Highly Abstract

Metadata based production requires higher technological dependencies compared to methods that rely on markup.
This is exemplified in the work of the engineers and researchers Martin Kleppman and Joseph Gentle. Their introduction of Eg-walker, a new text collaboration algorithm, points out the weaknesses metadata heavy algorithms such as CRDTs. While they allow for the operation under high concurrency, CRDTs use 10 times as much computer memory than comparable applications (Gentle and Kleppmann, 2025). Another example for high dependencies are JavaScript frameworks, which require the installation of specific environments (NODE) and package managers (NPM) in order to be used.

Depending on larger amounts of existing software projects carries consequences for access, centralization, and maintenance of digital documents. How does a combination of the three impact the modes of production, exchange, and the knowledge they circulate?

Access to means of (re)production increasingly becomes subject to experts (programmers) and big corporations who have the technical capacities to navigate complex and computer memory intensive software dependencies. This might feed into a wider and ongoing power imbalance over text production and circulation, reinforcing dominant narratives over marginalized knowledges and practices. By affecting how accessible sources are and to whom, an increase in dependencies could consolidate centralized control by big tech companies, while taking agency away from small scale operations and individual users.

Additionally, the centralization of services creates a high degree of instability for the sources and processes that rely on them. Recent outages of internet service providers such as Amazon Web Services and Cloudflare (Gross, 2025; Ray, 2025) illustrate the extend of this and its affect on the ability to maintain sources on creators side. That maintenance becomes increasingly complex and time intensive can also be observed through continuing software rot, which poses a risk for longevity of digital objects that rely on large amount of software.

But dependencies are not merely an infrastructural concern. By abstracting underlying processes, they also fundamentally affect how we use and relate to technology. This becomes apparent with the previously introduced example of JavaScript frameworks. Omitting markup language in the production process, they render entire web applications from metadata in favor of seamless user [7] experiences. But like all conveniences, this one does not come without cost: The user's experience is based on the fact that JavaScript frameworks run their own code behind the scenes. For the sake of simplicity, this process is abstracted. Firstly, this abstraction results in the projects using frameworks being larger and computationally more expensive to operate. Secondly, the use of JavaScript frameworks might change the programmers relation to the web and its materiality.

[7] 'user' here refers to both, the developer, using the framework as a tool, and the user who visits the website

(bae0b6b - 2026-04-02 12:28:56+02:00 - Kim Kleinert - Much can be said about the webs materiality: its grain (Chimero, 2015),

how it is handmade (Carpenter, 2015), vernacular (Lialina, 2005) or just words (Jackson), powered by the sun (Decker, 2024) or a house we live in (Schwulst, 2018). But most important right now is to remember, that HTML documents build the core of the web.)

And no matter what tool or language a website is originally built with, the final form that a browser displays and a user views is HTML. Creating an entire website in a JavaScript framework disconnects the production process from the final form and user who interacts with it. This separation raises concerns for semantic HTML and accessibility of websites. HTML consists of various tags, each of them having a specific purpose. Semantic HTML describes the intentional use of those tags to define the semantics (meaning) of their content. On the bottom line, the use of semantic HTML ensures readability and compatibility for humans, devices, and software alike. Being clearly structured and descriptive by nature, semantic HTML also enhances accessibility for users who visit websites with assistive technology such as screen readers. When omitting HTML in the production process of a website by using a JavaScript framework, the developer might loose sight of HTML tags purposes. What results are websites with meaningless structures that are more complex to read and maintain by humans and render by software such as browsers of assistive technology.

Thus far this section has argued, that detached production methods based on metadata require higher technological dependencies. This might increase work time spent on maintaining underlying structures to ensure longevity of documents and brings forth a centralization of access and control on the side of corporations. Further, dependencies create not only larger projects but add layers of abstraction to processes, separating production of a project from its final form. Metadata based production increases abstraction, but also legibility.

Metadata, a Question of Legibility?

Legibility is not just readability, but operates through combined methods of simplification, classification, and standardization such as earlier mentioned metadata schemas. Together these methods enable to interpret that what is readable to give it meaning (Münchmeyer, 2026). In the first chapter I have shown how granularity and amount of metadata drastically increase today which implies heightened levels of legibility

for machines. Commonly framed as technological improvement, Pomerantz argues that ‘more [metadata] is indeed more, on the larger scale of creating an environment that encourages innovation’ (2015) while entirely disregarding risks of surveillance and human incomprehensibility.
Initially, I introduced markup and metadata as a shared context, coordinating collaboration between both humans and machines. But as the legibility question demonstrates, the metadata prevalent today is clearly directed towards machines. Too abstract for humans, the context it once invoked, becomes meaningless amongst collaborators.
(cfbd33d - 2026-04-05 11:07:22+02:00 - Kim Kleinert - wait, but this is metadata and you CAN read it, right?)
How do underlying dependencies and shifting legibility affect the communication layer of collaborative work? In the following I will move from the prior technical understanding of collaboration to a social one, to look more closely at what collaborative work entails and how this work is represented and coordinated by technology.

Implicit Articulation

Collaboration defies clear boundaries of the individual worker, operates on asynchronous, nonlinear timelines and breaks with expectations of total efficiency.
(0f8dfe2 - 2026-02-17 10:26:46+01:00 - Kim Kleinert - Collaborative work, next to following formal task descriptions involves many informal tasks: The preparation and sharing of files, discussion of changes made and planning of time for meetings or deadlines are all part of collaborative work, yet, they often remain unrecognized.)

To describe these informal tasks and their relation to rationalized concepts of work, sociologist Anselm Strauss coined the concept of ‘articulation work’ (1985).
(fda1472 - 2026-02-17 11:07:01+01:00 - Kim Kleinert - Articulation work involves the managing of unanticipated contingencies in real time or in other words the work that enables other work to happen. This type of work appears across different professions. Secretaries, doctors or pedagogues tasks can to a large extend be considered articulation work, while an accountant will be less occupied with it.)

Articulation work often fails to be included in formal definitions of work

(5f74940 - 2026-04-04 14:55:02+02:00 - Kim Kleinert - because it escapes the parameters which we use to classify what counts as work and therefore remains illegible to rationalized models)

Arguing for legible articulation work as crucial to collaborative processes, it is important to remember, that legibility is not an absolute condition, neither being inherently positive, nor universal.

(29099c0 - 2026-02-17 16:25:10+01:00 - Kim Kleinert - Legibility of articulation work can increase its legitimacy and therefore be compensated or supported more adequately.)

On the other hand, 'for every grain in granularity of description, there may be increased risk of surveillance' (Strauss and Star, 2016, 367).

(582a969 - 2026-03-01 20:05:31+01:00 - Kim Kleinert - Think of the in-text questions or signs of agreement you leave for a coworker in a shared document, the text messages you exchange to arrange an upcoming meeting or the discussion involved in combining different drafts.)

Most of these processes today are not explicit steps anymore but automated, enabled by metadata,

(0479b06 - 2026-04-04 14:53:45+02:00 - Kim Kleinert - their representation in software has always been challenging: a balance act between prescription and support, division, and conflation)

When offices became paperless in the 1980s, not only document storage and retrieval but also collaborative work were increasingly mediated by computers.

(65ad931 - 2026-04-05 11:16:56+02:00 - Kim Kleinert - Since then the study and development of collaborative software (also known as computer supported cooperative work or CSCW) has tried to resolve the unpredictability and situatedness of articulation work into simple user interface and predefined settings.)

Today, detached production methods based on metadata bring a new angle to what CSCW falsely treats as an engineering problem.

(f32522e - 2026-02-17 21:02:35+01:00 - Kim Kleinert - In a

more critical understanding, software is a model which tries to solve human communication and interaction through computer engineering. A model is a partial, situated representation, that should not be mistaken for reality. The reality of collaboration, which is messy, incoherent and unpredictable.)
(ac9ea5d - 2026-02-17 21:03:59+01:00 - Kim Kleinert - Collaborative work requires discourse and the ability to contain multitudes instead of fixed uniform identifiers. How can software support this reality, instead of trying to treat it as a problem that has to be solved through computational schemes?)

The earlier example of CRDTs shows, that articulation work, such as the task of merging different versions of a text to create a singular body of work, is not subject to the user anymore. Instead, articulation becomes implicit: being highly technically integrated turns it into a metadata based calculation that can be resolved by algorithms.

In two workshops [8] I conducted for this research I noticed how this compromises collaborators ability for explicit coordination of their processes, by lack of practicing it. Consequently, collaborators don't seem to understand the importance of this type of work and respond with confusion when given the opportunity for it.
(b788255 - 2026-02-18 20:25:59+01:00 - Kim Kleinert - A prominent example are git commit messages.)

[8] In which we explored how tools shape collaboration by looking through the actions that they prescribe. Not using any writing software, we collaboratively versioned, merged and edited text into a zine.

In contrast, the understanding of markup languages as a writing practice enables us to recognize how it inherently supports articulation work, through firstly,
(22b9d28 - 2026-04-04 15:00:17+02:00 - Kim Kleinert - the writing of source code comments, indicating process to collaborators and readers)
and secondly,
(0608c37 - 2026-04-04 15:00:59+02:00 - Kim Kleinert - the use of semantic HTML, as the meaningful tagging of text is an act of instructing reading, co-writing or rendering)

In this section I introduced the concept of articulation work, describing processes of collaboration that often remain unformalized. Under metadata based production, articulation work becomes not only legi-

ble to machines, but is ‘solved’ by algorithms. Taking this tendency and the previous discussion of legibility into account, I argue for human legible articulation work as crucial to collaborative processes. How can this be put into practice?

Articulating Context

This chapter describes the effects of implicit articulation work on shared context. Against this backdrop, the Git software opens up alternative methods. I will conclude this chapter with introducing GitPub and its design principles: linking together questions discussed throughout this thesis demonstrates how they can build a base for practice.

Detaching, Dissolving, Dismissing

Detached production based on metadata brings forth a dissolution of shared context, which I observe along two main axes. Firstly, in an abstracting gesture, text is stripped of humanly meaningful context in favor of machine legibility. By being treated as mere engineering problem, articulation work is rendered implicit and solved by software's algorithms. Secondly, and somewhat transversal to the first axis, lies the loss of access to the shared context. Access to means of collaborative production and the coordination of it becomes increasingly proprietary: centrally controlled by corporations.

The dissolution of shared context leads to an unlearning of articulation work and the dismissing of its importance in collaborative processes.

To what extend can collaborative software support mutable shared contexts, instead of solving problems through abstracted models?

Git: An Articulated Software

```
(69cdd75 - 2026-04-05 11:38:25+02:00 - Kim Kleinert - The fact that you can read this shows how git metadata serves both, its own operations and its users. But How?)
```

Asynchronous
Git provides great support for asynchronous collaboration across time and work environments. The software allows a body of work to diverge with methods such as branching (`git branch`).

Sharing Intentionally
The Git software's focus on asynchronous processes allows collaborators to share changes intentionally. This is enabled by a distinction be-

tween so called ‘local’ and ‘remote’ repositories, where ‘local’ indicates the files on your personal computer and ‘remote’ the shared files that are for instance hosted on a server and can be accessed by all collaborators.

Explicit Collaboration

Compared to collaborative writing or publishing tools examined earlier, Git does not take on much of its users work. Yet it supports coordination of collaborative work by explicitly asking users to take specific actions. This can be seen on the example of merging different versions of a file into one (`git merge`). While collaborative real time editors, such as Etherpad, solve this issue with the more or less seamless implementation of algorithms, Git does not provide a solution for it but will create a ‘merge conflict’. To solve a merge conflict, the collaborators have to manually select which changes they want to keep and which they want to discard in order to arrive at a singular version. A merge conflict creates space for discussion, it takes time off ‘work time’ acknowledging the articulation work necessarily involved in collaborative processes.

Another way in which Git supports explicit coordination of collaborative processes are commit messages (`git commit -m`). As notes that indicate changes made in a version, commit messages support not only collaborative, but also individual processes. They render modifications legible, without the need to parse through the entire document and find differences between versions. Thus, commit messages are useful for project maintenance, documentation, and communication.

Version Control

Versioning, as one of the most prominent features of git, is not only crucial in software development but inherent to many creative practices in fields such as arts, music, and literature. And so it has become an underlying feature that can be found in different collaborative software projects such as Etherpad or Google Docs. Yet, version control is here named ‘history’, which already indicates the strictly linear motion it enables. Allowing, basically, to only go back and forth implies that work evolves progressively towards the latest version. Git, in comparison, enables more variation through for example the rewriting of history (`git rebase`) or reverting to specific changes (`git revert`) while leaving others intact. Breaking with assumptions of lin-

ear processes, Git versions do not merely exist side by side, but interrelate in ways that more accurately reflect the nature of collaborative work.

Decentralization
The extend of gits decentralization and dependence on infrastructures such as internet connection are situationally adaptable. Today, Git is predominantly being used through platforms such as GitHub. But the Git software does not need to be mediated by a platform, nor does it require a centralized server. Instead, the remote can be hosted on a collaborators computer or even a USB.

Throughout the above examination we could understand how metadata elements such as commit messages and nonlinear process coordinate collaborative work across time and distance. I also emphasized that Git intentionally leaves space for users to make context driven decisions and adjust their degree of infrastructural autonomy, instead of taking over the work entirely. This key argument has also been discussed by computation and cognitive scientist Eevi E. Beck. In her 1994 study, following the collaborative writing process of two scholars across distance, she notes that collaboration is too complex to be represented in a technical model entirely. 'Instead, technological support could be provided for parts of the process in a way which allows its users to define and develop their own methods of coordination and control' (Beck, 1994). At the time Beck's work was published, Git did not yet exist. And although her examination stays strictly within a scholarly writing context, the Git software adheres to the necessities for variation and agency she derives from her study. In the Git software these principles can come at the cost of technical complexity, which constitutes an entry barrier for practitioners with less technical backgrounds. My goal in the following is to show how Git can be used outside of development centered practices and on different levels of complexity.
The intersection of software engineering and publishing practice, and thus the possibilities of using Git as alternative protocol for collaborative work has been explored by numerous projects. In the following I will discuss 'Giterature' and 'Visual Culture', two approaches that are of particular interest for this research.

Giterature poses an alternative model to editorial processes in litera-

ture. The concept and practice is coined by digital humanities scholars Servanne Monjour and Nicolas Sauret as part of the Aprübt publishing collective. In ‘For a Giterature’ (translated form the french original ‘Pour une gittérature’) Monjour and Sauret propose the Git protocol as an editorial protocol that transforms literature politically and poetically through the appropriation of, and formal experimentation with literary texts. Giterature understands ‘Git as a writing environment [that is] inherently processual, performative and conversational’ (Monjour and Sauret, 2021).

Visual Culture, is an application that renders metadata of a Git based website visible on the page itself. It is developed and used by Open Source Publishing (OSP), a collective who does graphic design, research, and pedagogy with and around F/LOSS. OSP describes their application and motivation behind it as follows: ‘when we started sharing our source code through the internet, we found all interfaces to Git were geared to sharing text files. We want to create an interface for sharing our work’ (2016). By acknowledging the discrepancy between sharing files and sharing work, Visual Culture closely connects to GitPub, which I aim to introduce in the following. Both tools understand Git metadata as space for articulation work and highlight the necessity of including this type of work visibly in its final form.

Coordinating and Capturing Collaborative Work with GitPub [9]

[9] project documentation and scripts are published in this Git repository: gitlab.com/km_kt/gitpub-scripts

The research for this thesis is closely interrelated with the development of GitPub.

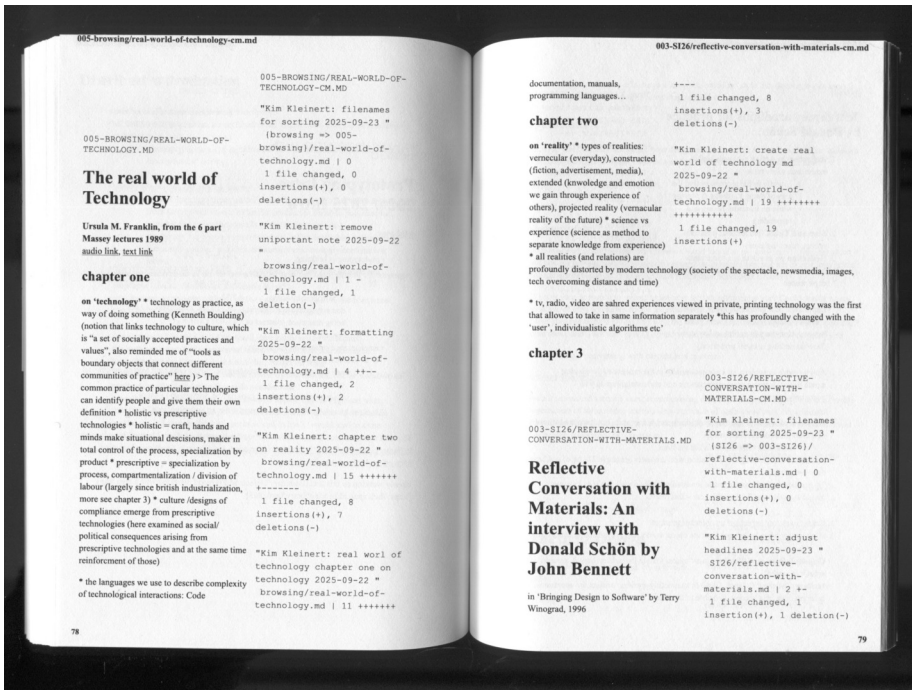
```
(930b2e6 - 2026-03-25 20:38:56+01:00 - Kim Kleinert - In this
moment GitPub is still work in progress.)
```

Built upon the Git software, GitPub is a tool that turns files inside a Git repository and their metadata into a printed or web based publication. GitPub proposes an understanding of Git metadata as both, coordinating, and capturing collaborative processes. It’s inquiries therefore unfold on two levels: The first is rooted in the moment of collaboratively writing a text or producing a publication. Here GitPub asks how the Git software shapes coordination of collaborative processes through the use of human readable and explicit metadata.
On the second level, GitPub looks at the act of publishing a record of collaborative process as part of the main body of work by directing the focus from metadata as means to an end of production to including it

in the publications final form. What does this act reveal to readers and collaborators? Can it open up the collaborative process to discussion and change?

Additionally, GitPub attempts to shift how the regular Git user conceives the software itself. Working with Git in day to day operations, the familiarity of its gestures and rhythms tends to conceal the software's own ways of writing and its cultural situatedness [10]. Sociologist Susan Leigh Star describes this in her concept of naturalization in which naturalized objects become invisible to members of community of practice through omnipresence and use, causing decreasing criticality towards the object (Star, 2016). GitPub uses Git otherwise, therefore reflecting attention to the software and its making. Here de-naturalization could enhance a critical understanding of the Git software's intricate culture and break open its engineering context, bridging between communities of practice.

[10] For more information I recommend the comics about Git by xkcd: xkcd.com/1296 and xkcd.com/1597 which are further discussed on this wiki: explainxkcd.com/wiki. Part of my inquiry into Git culture was also Aymeric Mansoux' critical research on free and open source culture, published under the title 'Fork Workers' (2013).



A GitPub prototype: metadata is presented alongside the main text.

Throughout this text I have addressed matters of dependencies, centralization and articulation and how they impact collaborative production of publications under the prevalence of metadata. I will conclude this chapter with sharing some design considerations of the GitPub tool. The tool itself and the following principles are not intended as

solutions but rather make space for experimentation, discussion, and reflection.

GitPub Design Principles

1. Articulation through Markup Language
GitPub's input files are written in Markdown or HTML. Firstly, the use of markup language ensures a separation between source and print file. This renders the input document independent from the software that processes it, and enables single source publishing [11].
Additionally, software independent and single source based workflows simplify sharing, maintenance, and longevity of documents. As I have outlined earlier, markup languages also ensure humanly legible documents. This prevents abstraction and provides inherent modes of articulation through the use of semantic markup tags or source code comments.

[11] 'Single source' or 'hybrid' publishing allows for the output of one source file into various formats, opening up to different (production) environments and distribution systems

2. Coordination through Metadata
GitPub is critical of its metadata but does not refuse it entirely, nor instrumentalizes it for convenience or efficiency. Introducing the Git software above, we could see that it makes extensive use of metadata, not only for storing and rendering versions but also to enable humanly legible coordination between collaborators in asynchronous workflows. GitPub leverages the explicit coordinative capabilities of Git metadata which it understands to articulate process.

3. Autonomy over Means of Production and Circulation
Open licensing and relying on protocols instead of platforms encourage autonomy over means of production, access, and circulation.
Firstly, GitPub is currently licensed under Collective Conditions for Reuse (CC4R) which allows users to analyse, modify and redistribute source code with the exemption of contributions to oppressive power arrangements and the requirement to credit the collective origins of the work (Constant, 2025).
Secondly, GitPub is based on the Git protocol and therefore independent from platforms like GitHub or GitLab. Thus, it might be used entirely locally or in other ways which I described in the above section on Git and decentralization.

4. Modularity and Partial Support
GitPub adheres to the notion that collaborative work can't be fully grasped by software and instead provides partial support which leaves enough openness for users to adapt configurations to their own environments and processes. Integrated technologies remain tangible and therefore interchangeable and can be linked to other existing tools. Coordinating process, GitPub is primarily intended to be part of the collaborative writing. In this case, it is used from the beginning on, and thereby inquires the reciprocal relation between Git metadata and collaborative process that creates it. However, GitPub can also be run on existing repositories, capturing process. Here the tool is separated from the files it is applied to and the publication is rendered in an external location, keeping the working repository clean. This renders the proximity of relationship between tool and process flexible to different circumstances.

5. Different Degrees of Legibility
GitPub supports different degrees of legibility. This thesis is written and rendered using GitPub. Facing constraints of an 8k word limit for this text lead to a thesis specific version of the tool. Unable to include this thesis' entire metadata alongside the 'main text', I developed a function to cite my documents metadata by expanding my existing reference framework using markdown, a BibTex file to store references and a CSL file for formatting the citations.
This function came to be more than a mere compromise. Selectively citing its file's metadata instead of including everything in the final publication treats publicness of metadata with more intention and maintains the notion that full legibility is neither absolute nor positive.

Conclusion

This thesis set out to understand how metadata and markup languages have become formalized through the introduction of digital technologies and how this impacts collaborative work in the field of publishing.

In their historical analog and later digital forms, markup and metadata create a shared context for collaborative work. This understanding counters the common assumption of marks and data as technical byproduct or mere interchange format and instead proposes them as writing practices that actively inform collaborative production through coordination and structuring.

We then considered how the transformation from static markup to increasingly diffuse platforms and frameworks based on metadata has had an impact on forms of collaboration, and specifically on where and how articulation work occurs.

Discussing the latest turn in the formalization of markup and metadata, I point out three intertwined tendencies. The shift towards metadata based production establishes higher technological dependencies, which, in combination with increasing granularity of metadata causes abstraction for human collaborators. This, in turn, increases legibility for machines. Under these conditions shared context dissolves: collaborators loose access to it through proprietary means of production and articulation work is rendered implicit and solved by software's algorithms.

I conclude this thesis with highlighting the importance of articulation in collaborative work. GitPub then is posed as an experiment in making that articulation work more explicit through activating and making visible metadata.

This text offers both a social, and technical understanding of collaborative work. Conceiving engineering and sociology not as distinct but continuous fields, allows us to see how they reciprocally inform another and together shape collaboration. I here argue, that software can and should always only be an approximation to articulation and coordination as part of collaborative work and not a full automation of it.

Because, as philosopher Ivan Illich reminds us, technology merely displaces work onto the machine or other workers, but never resolves it (1981). But what if instead of intending to resolve work, technology would create a context to share it?

References

Beck, Eevie E. 1994. “Changing Documents/Documenting Changes: Using Computers for Collaborative Writing over Distance.”*The Sociological Review*, no. 42: 53–68. https://doi.org/10.1111/j.1467-954X.1994.tb03409.x.

Bieniusa, Annette, Martin Kleppmann, and Marc Shapiro. 2026. “About CRDTs.”*Conflict-Free Replicated Data Types*. https://crdt.tech.

Burgess, Helen J. 2010. “<?php>: ’Invisible’ Code and the Mystique of Web Writing.” In *From A to <A>: Keywords of Markup*, edited by Bradley Dilger and Jeff Rice, 296. University Of Minnesota Press.

Carpenter, J. R. 2015. “A Handmade Web.” https://luckysoap.com/statements/handmadeweb.html.

Chimero, Frank. 2015. “The Web’s Grain.” Blog. https://frankchimero.com/blog/2015/the-webs-grain/.

Constant. 2025. “CC4r * COLLECTIVE CONDITIONS FOR RE-USE.”*Unbound Libraries Documentation*. https://constantvzw.org/wefts/cc4r.en.html.

Coombs, James H., Steven J. DeRose, and Allen H. Renear. 1987. “Markup Systems and the Future of Scholarly Text Processing.” Edited by Peter J. Denning. *Communications of the ACM* 30 (11). https://doi.org/10.1145/32206.3220.

Decker, Kris De. 2026. “About the Solar Powered Website.”*LOWTECH MAGAZINE*. https://solar.lowtechmagazine.com/about/the-solar-website/.

“F. 127v : Marque de Pecia : Finis Xxx Pet.” n.d. San Marino, Huntington Library, HM 26298.

Febvre, Lucien, and Henri-Jean Martin. 1976. *The Coming of the Book: The Impact of Printing 1450-1800*. Verso Classics 10. Verso.

Franklin, Ursula M. 1990. *The Real World of Technology*. CBC Massey Lecture Series. CBC Enterprises.

Gartner, Richard. 2016. *Metadata: Shaping Knowledge from Antiquity to the Semantic Web*. Springer.

Gentle, Joseph, and Martin Kleppmann. 2025. “Collaborative Text Editing with Eg-walker: Better, Faster, Smaller.” In *Proceedings of the Twentieth European Conference on Computer Systems*, 311–28. https://doi.org/10.1145/3689031.3696076.

Gross, Jenny. 2025. “Amazon Outage Forces Hundreds of Websites Offline for Hours.”*The New York Times*, October.

Harper, Douglas. 2026a. “Mark-up - Etymology, Origin & Meaning.”*Etymonline*. https://www.etymonline.com/word/mark-up.

———. 2026b. “Meta- - Etymology & Meaning of the Prefix.”*Etymonline*. https://www.etymonline.com/word/meta-.

Illich, Ivan. 1981. “Vernacular Values.” In *Shadow Work*. London: Marion Boyars.

ISO/IEC JTC 1, comittee. 2015. “ISO/IEC 2382:2015(en), Information Technology — Vocabulary.” Online Browsing Platform. *Iso*. https://www.iso.org/obp/ui/en/#iso:std:iso-iec:2382:ed-1:v2:en.

Jackson. n.d. “Words.” https://justinjackson.ca/words.html. Accessed April 2, 2026.

Kleinert, Kim. 2026a. “Gitpub Prototype.”

———. 2026b. “Marc Record Example.”

———. 2026c. “Markup Example.”

Lialina, Olia. 2005. “A Vernacular Web.” https://art.teleportacia.org/observation/vernacular/.

Library of Congress. 2009. “What Is a MARC Record and Why Is It Important?”*Understanding MARC*

Bibliographic: Machine-Readable Cataloging. https://www.loc.gov/marc/umb/um01to06.html.

———. 2014. “Dublin Core in RDF-XML Example.”

Manovich, Lev. 2001. *The Language of New Media*.

Mansoux, Aymeric. 2013. “Fork Workers.”*Are You Being Served?* https://areyoubeingserved.constantvzw.org/Fork_Workers.xhtml.

McKenzie, Donald Francis. 2002. “Printers of the Mind: Some Notes on Bibliographical Theories and Printing- House Practices.” In *Making Meaning*. University of Massachusetts Press.

———. 2009. *Bibliography and the Sociology of Texts*. Cambridge University Press. https://doi.org/10.1017/CBO9780511483226.

MDN Contributors. 2025. “Introduction to Client-Side Frameworks - Learn Web Development MDN.”*MDN Web Docs*. https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Introduction.

Meyer, Aljoscha. 2026. “Notes on CRDTs.” Blog. *Worm-Blossom*. https://worm-blossom.org/notes/crdts_cl.

Monjour, Servanne, and Nicolas Sauret. 2021. “Pour une gittérature: L’autorité à l’épreuve du hack,” 16 pages, pages 237–252. https://doi.org/10.48611/ISBN.978-2-406-12363-7.P.0237.

Münchmeyer, Rosa. 2026. “‘Going Underground’: Legibility, Illegibility and the Limits of State Vision.” {{BA}}, Berlin: Technische Universität Berlin.

Nix, Larry T. 2009. “Evolution of the Card Catalog.”*The Library History Buff*. https://www.libraryhistorybuff.org/cardcatalog-evolution.htm.

O’Reilly, Tim. 2005. “What Is Web 2.0.”*O’Reilly Media*. https://web.archive.org/web/20130501095209/http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1.

Open Source Publishing. 2016. “Visual Culture.”

Österreichische Nationalbibliothek. 2022. “1780: The Oldest Card Catalogue - Österreichische Nationalbibliothek.”*Österreichische Nationalbibliothek*. https://web.archive.org/web/20220302063835/https://www.onb.ac.at/en/about-us/650th-anniversary/timeline/1780-the-oldest-card-catalogue.

Phillips, Heather. 2010. “The Great Library of Alexandria?”*Library Philosophy and Practice*.

Pomerantz, Jeffrey. 2015. *Metadata*. MIT Press Essential Knowledge Series. MIT Press.

Ray, Alison Joan. 2015. “The Pecia System and Its Use.” PhD thesis, UCL.

Ray, Siladitya. 2025. “Cloudflare Says Outage ‘Resolved’ After Knocking X, ChatGPT And Other Websites Offline.”*Forbes*. https://www.forbes.com/sites/siladityaray/2025/11/18/cloudflare-outage-knocks-x-and-chatgpt-offline/; Forbes.

Schwulst, Laurel. 2018. “My Website Is a Shifting House Next to a River of Knowledge. What Could Yours Be?”*The Creative Independent*. https://thecreativeindependent.com/essays/laurel-schwulst-my-website-is-a-shifting-house-next-to-a-river-of-knowledge-what-could-yours-be/.

Star, Susan Leigh. 2016. “Misplaced Concretism and Concrete Situations: Feminism, Method and Information Technology.” In *Boundary Objects and Beyond: Working with Leigh Star*, edited by Geoffrey C. Bowker, Stefan Timmermans, Adele E. Clarke, and Ellen Balka. Cambridge, Massachusetts: MIT Press.

Strauss, Anselm. 1985. “Work and the Division of Labour.”*The Sociological Quarterly* 26 (1). https://doi.org/10.1111/j.1533-8525.1985.tb00212.x.

Strauss, Anselm, and Susan Leigh Star. 2016. “Layers of Silence, Arenas of Voice: The Ecology of Visible and Invisible Work.” In *Boundary Objects and Beyond: Working with Leigh Star*, edited by Ellen Balka, Geoffrey C. Bowker, Stefan Timmermans, and Adele E. Clarke. MIT Press.

Wiktionary Contributors. 2026. “Contextus.”*Wiktionary, the Free Dictionary*.

xkcd. 2013. “Git Commit.”

———. 2015. “Git.”

Appendix

0_intro.md

id	date	message	diff	branch
69564000	2025-12-12 10:35:06+01:00	divide introduction into subchapters	-0 +10	• short-form • main • notebook • citemeta
171b3c8	2025-12-12 11:08:34+01:00	map wordcount	-1 +9	• short-form • main • notebook • citemeta
5.28e795	2026-02-01 17:53:39+01:00	state introduction structure and paste formerly thesis outline part - hopefully this can be merged into the planned structure	-6 +27	• short-form • main • notebook
c98ddce	2026-02-01 21:31:13+01:00	minor corrections + css	-3 +12	• short-form • main • notebook
c72a50f	2026-02-01 22:01:02+01:00	minor fixes and css	-9 +18	• short-form • main • notebook
a9dfb18	2026-02-01 22:02:19+01:00	manual sorting by filename	-0 +0	• short-form • main • notebook
f80d31a	2026-02-18 20:19:02+01:00	vague but crucial question	-0 +1	• short-form • main
acce9e0	2026-02-21 18:22:38+01:00	research question and text structure, quick sentences, needs further care	-16 +18	• short-form • main
18e6aa1	2026-02-27 11:32:53+01:00	Specify pecia as operational marks and add fidelty of text	-4 +5	• short-form • main
2255372	2026-02-27 15:39:14+01:00	modify marloes suggestions into intro	-5 +8	• short-form • main
f3f739f	2026-03-05 17:00:08+01:00	introduction intro	-2 +18	• short-form • main
a4752a8	2026-03-07 20:18:23+01:00	transcribed annotations from paper copy	-30 +23	• short-form • main
6b00197	2026-03-08 15:59:12+01:00	spellcheck	-120 +427	• short-form • main
bed8a68	2026-03-08 18:53:56+01:00	dear michael, in the manner of addressing my reader in commit messages, here are a few comments on this second-reader version: text in italic are mainly notes thet are not fully formulated yet. the headlines will change but for now i need them to		

		structure my document while writing. beginning and end of the text are still pending, my thoughts on this are not clear yet, this is also why there is no conclusion.	-6 +2	• short-form • main
0ac1a62	2026-03-08 18:56:53+01:00	note for michael	-0 +2	• short-form • main
b560cb0	2026-03-27 17:36:55+01:00	intro intro and definition of production	-4 +9	• short-form • main
93a6235	2026-03-28 21:17:39+01:00	add changes from printed version. Include reference for mckenzie's work, I want to write more about it but dont have any words left, in fact, at this point i am beyond limits already.	-31 +21	• short-form • main
3bc2715	2026-04-02 08:58:05+02:00	You are reading this text within a different time, place and knowledge than I initially wrote it. 'Context' reminds us, that reading is not just the static updating of your ideas, but an interweaving of your thoughts, surroundings and mine.	-14 +24	• short-form • main
bd6aae8	2026-04-02 09:00:10+02:00	3 first sentences. will keep all for now, some descisions have to happen later.	-4 +2	• short-form • main
01a9458	2026-04-02 21:17:39+02:00	cutting minimal	-6 +5	• short-form • main
438f9e1	2026-04-04 15:14:29+02:00	spelling	-46 +42	• short-form • main
f85e2cc	2026-04-05 11:43:52+02:00	adding commits	-15 +6	• short-form • main
48cf029	2026-04-05 12:05:19+02:00	fix image rendering with captions	-10 +6	• short-form • main
ca8979c	2026-04-05 18:48:56+02:00	add img mediaval scribes	-0 +4	• main
1f3e348	2026-04-05 21:18:29+02:00	change img mediaval scribes	-4 +5	• main
e5d86d8	2026-04-10 17:50:45+02:00	intro big cuts, smaller changes around after last feedback	-7787 +28	• main
cefdba9	2026-04-11 17:25:21+02:00	puh intro edits, sentences feel long and complex, return later, maybe	-8 +5	• main
5542603	2026-04-14 18:06:39+02:00	citation	-13 +11	• main
d542e5f	2026-04-15 15:46:11+02:00	small edits getting there	-7 +7	• main
2092957	2026-04-15 16:00:08+02:00	spelllinng	-12 +12	• main
9508515	2026-04-15 16:48:15+02:00	punctuation	-1 +1	• main
800e27f	2026-04-16 08:17:03+02:00	final reading	-46 +47	• main

```
1d37474  2026-04-16 16:23:40+02:00  quotes page numbers  -5 +6  • main
```

chapter_1.md

id	date	message	diff	branch
b11bd06	2025-12-10 16:32:29+01:00	draft subchapter headlines	-0 +8	• short-form • main • notebook • citemeta
e8c385d	2025-12-10 16:56:21+01:00	add bullet points to subchapter headlines (very tentatively, I feel all I have for now are assumptions)	-1 +7	• short-form • main • notebook • citemeta
0aadf26	2025-12-10 17:08:33+01:00	list of possible examples, this requires further research and decision where to set focus	-1 +9	• short-form • main • notebook • citemeta
5be7915	2025-12-11 12:56:38+01:00	suchapter 2 emphasizing today because i state the rest in the intro	-1 +1	• short-form • main • notebook • citemeta
b853c79	2025-12-11 17:50:52+01:00	mv example list to local notes requires more thought	-9 +1	• short-form • main • notebook • citemeta
3d604be	2025-12-12 11:19:49+01:00	map wordcount	-1 +7	• short-form • main • notebook • citemeta
f89d2ee	2025-12-31 13:14:21+01:00	first draft contextualizing markup language subchapter 2	-0 +11	• short-form • main • notebook • citemeta
47f41fa	2025-12-31 13:41:18+01:00	metadata definition missing contextualization eventually add a proper definition like metadata is data about data or a statement about a potentially informative object	-0 +4	• short-form • main • notebook • citemeta
687e49a	2026-01-02 09:42:07+01:00	previous writing revs	-4 +8	• short-form • main • notebook • citemeta
2afa0f2	2026-01-03 10:36:45+01:00	metadata definition and context - use historical marc records markup to bridge back to markup language e.g. summarize section?	-4 +5	• short-form • main • notebook • citemeta
67f5cbf	2026-01-03 22:29:47+01:00	reminder to continue writing tomorrow metadata history and context of use		

		today	-1 +3	• short-form • main • notebook • citemeta
3871f11	2026-01-04 14:48:37+01:00	metadata context of use today in 2 main tendencies	-3 +10	• short-form • main • notebook • citemeta
da17524	2026-01-04 16:37:33+01:00	metadata and markup relation towards another, distinct boundaries become more blurred but where do i go with this observation	-4 +3	• short-form • main • notebook • citemeta
5abc10e	2026-01-04 16:51:20+01:00	formatting	-5 +8	• short-form • main • notebook • citemeta
6656842	2026-01-04 21:24:28+01:00	current lost with introducing the idea of metadata based rendering rn, where? current order of chapter 1 subchapters does not make sense	-1 +19	• short-form • main • notebook • citemeta
b9e35bf	2026-01-05 10:37:56+01:00	expanding on shifts historical context and attempt to introduce idea of metadata besed rendering, needs furhter thought and connection to the rest of the chapter	-4 +9	• short-form • main • notebook • citemeta
4295be9	2026-01-06 21:02:18+01:00	citation trial run	-1 +26	• short-form • main • notebook • citemeta
c9129df	2026-01-06 21:21:32+01:00	citation test run 2 working when pandoc conversion: pandoc –citeproc chapter_1.md -o chapter_1.html	-1 +6	• short-form • main • notebook • citemeta
6a1cf5e	2026-01-09 19:39:05+01:00	reconsidering Helen Burgess shift from two dimensional to distributed rendering. this got too deep here and needs to be moved elsewhere	-16 +37	• short-form • main • notebook • citemeta
3bdb8b0	2026-01-09 19:46:42+01:00	moved analysis and implications of metadata based rendering to chapter 2	-22 +26	• short-form • main • notebook • citemeta
1c84621	2026-01-10 09:05:07+01:00	introduce metadata based rendering and text prodcution: in git, which I am writing this text in right now, versions of a file are not stored as a whole. instead the changes made in a file are indexed. To retrive a specific version, it will be reassembled from the indexed metadata using hash driven reference		

		clusters.	-2 +8	• short-form • main • notebook • citemeta
70f8d7e	2026-01-10 15:01:52+01:00	situate shifts historically. watch word count	-5 +18	• short-form • main • notebook • citemeta
cfd3760	2026-01-17 10:09:49+01:00	insert edit comments to continue writing	-19 +9	• short-form • main • notebook • citemeta
0a450e1	2026-01-17 15:48:14+01:00	I am writing this text in markdown! restructuring some other paragraphs on the side	-12 +22	• short-form • main • notebook • citemeta
903260	2026-01-17 16:16:58+01:00	markup languages still enable coordination between parties of production but human and computer. markdown renders this *italic*.	-1 +1	• short-form • main • notebook • citemeta
d201a74	2026-01-17 18:07:55+01:00	While writing this section about metadata, I am also manually adding and programmatically generating metadata. How so? This text is written in a git repository. In case this hasn't come up yet, git is a software that helps you keep track of changes made in a file and lets you collaborate with others on the same files. The tracking of changes is also called version control. Git adds metadata, such as unique identifiers, to each version to store and retrieve it. Git metadata is not only crucial to version control but also for collaboration: next to unique identifiers, changes are provided with author name, timestamp and a so called commit message in which the author making the change describes what was changed. You are reading a commit message right now.	-7 +7	• short-form • main • notebook • citemeta
cb4e498	2026-01-17 22:22:01+01:00	metadata tendencies today, introduce examples for granularity and component structuring	-2 +23	• short-form • main • notebook • citemeta
5b38a14	2026-01-18 16:10:10+01:00	social and politacl implications of technological shifts: add more specific examples here	-17 +14	• short-form • main • notebook • citemeta
69b7a3b	2026-01-18 17:06:49+01:00	formatting	-13 +15	• short-form • main • notebook • citemeta

7ddda5e	2026-01-18 17:39:22+01:00	to marloes: this commit marks the snapshot i will be sending to you. the chapter grew in size and i am currently a bit over my aspired word count but hope i can edit that out with time and paying more attention to wording / style, something i have tried to ignore so far in favour of content. I am getting a better idea for where this chapter is going but theres still work to be done, especially chaotic paragraphs i emphasized in italic, so you know the idea is there but its not really worked in yet.	-1 +1	• short-form • main • notebook • citemeta
f8cc87e	2026-01-21 20:40:50+01:00	implement external reference rendering based on user input citation true	-13 +8	• short-form • main • notebook • citemeta
1acd865	2026-01-24 17:27:07+01:00	working in feedback marloes from 22-01-26 - i am stuck	-35 +43	• short-form • main • notebook • citemeta
72fefef	2026-01-24 22:09:00+01:00	fundamentally restructured subchapters arrangement to help me writing and the reader reading, hope this helps	-46 +52	• short-form • main • notebook • citemeta
88cf708	2026-01-26 08:25:04+01:00	rebuilding bridges to the new structure	-9 +8	• short-form • main • notebook • citemeta
7.812e59	2026-01-27 14:36:46+01:00	metadata taxonomy change pomerantz to gartner	-1 +12	• short-form • main • notebook • citemeta
0ed2ae3	2026-01-28 18:24:28+01:00	small changes and tests csl nothing chnaged so far, rendering metadata references the correct id and outputs status content	-17 +17	• short-form • main • notebook • citemeta
ab41a13	2026-02-01 16:18:27+01:00	general edits	-23 +40	• short-form • main • notebook
6df0e3f	2026-02-01 20:28:57+01:00	git auto merged this chapterwith prev version on side branch, revert to latest commit on main	-1 +1	• short-form • main • notebook
7c8d895	2026-02-01 20:43:24+01:00	add inline metadata citation with at symbol + short hash of the commit i want to reference	-6 +4	• short-form • main • notebook
cc0f5a8	2026-02-01 20:47:32+01:00	This in-text citation, and all the other ones in this text, are enabled by citation style language (CSL). CSL is an XML based markup language that enables		

		computer readable formatting of scholarly citation styles such as Harvard or APA. I adapted the harvard referencing csl file in order to be able to cite this documents metadata.	-3 +1	• short-form • main • notebook
ad36998	2026-02-01 20:51:17+01:00	I think is mainly used by other software projects such as Zotero (a citation management tool), which would explain why there is little documentation about csl. Its constructed upon a fixed voabulary, accepting only few terms and types as values (compared to html or markdown). This is only developed for one specific purpose (scholarly citation) and this purposes mechanisms are deeply engrained in the language itself. they leave little to no space to break with biased structures scholarly citation systems are build upon.	-5 +1	• short-form • main • notebook
fc37e0c	2026-02-01 20:55:15+01:00	citing more metadata makes me wonder, can i argue for a final word count of the source file or does it have to be the print file?	-1 +1	• short-form • main • notebook
f82749f	2026-02-01 21:01:31+01:00	fix short hash of inline citation	-4 +4	• short-form • main • notebook
c98ddce	2026-02-01 21:31:13+01:00	minor corrections + css	-3 +12	• short-form • main • notebook
eebcaba	2026-02-20 21:46:14+01:00	chapter 1 reworked draft with shared context	-49 +62	• short-form • main
7675ef4	2026-02-21 18:50:07+01:00	clean up, re render references file did not update properly	-25 +13	• short-form • main
b2e7f8c	2026-02-27 12:00:03+01:00	add print marks paragraph	-1 +2	• short-form • main
621a0fc	2026-02-28 11:24:00+01:00	some of marloes comments worked in. need to change the order of chapter 1 now to make it fit better	-9 +12	• short-form • main
fedcb5f	2026-02-28 11:39:29+01:00	reordered chapter contents	-24 +31	• short-form • main
8800cfc	2026-02-28 11:42:07+01:00	You can already sense, that this is where things get a bit more complicated.	-1 +1	• short-form • main
8abe75f	2026-03-01 10:57:29+01:00	changes?	-7 +9	• short-form • main
9d4d409	2026-03-05 16:57:50+01:00	metadata today introduce database architecture with paradigm of absolute flexibility	-1 +1	• short-form • main
a19f944	2026-03-07 22:01:36+01:00	transcribed annotations from paper copy	-35 +34	• short-form • main
6b00197	2026-03-08 15:59:12+01:00	spellcheck	-120 +427	• short-form • main

687d64b	2026-03-24 21:00:12+01:00	clarify merge algorithms in collaborative real time editors example	-4 +4	• short-form • main
0d333bc	2026-03-24 21:13:30+01:00	This exceeds the scope so Ill mention it here and see if i add it later: google docs and etherpad for example rely on ot, an algorithm that, other than crdts needs a centralized server to exchange data over. crdts can be used in p2p applications but this and their higher amount of metadata can lead to increasing overhead compared to ot.	-1 +1	• short-form • main
4a334c8	2026-03-27 08:25:47+01:00	add michaels comment on xml as interchange format from engineering perspective	-16 +18	• short-form • main
61917b2	2026-03-28 21:29:53+01:00	Focusing on coordination and articulation exposes the perception of markup as mere interchange format and regards it as writing practice instead, which from an engineering perspective is not a given.	-14 +10	• short-form • main
28f87fa	2026-03-28 21:32:36+01:00	I am writing this text in markdown	-5 +2	• short-form • main
893d80a	2026-03-28 21:43:35+01:00	its late, i should stop, anyway, next chapter notes tomorrow.	-10 +6	• short-form • main
c9368be	2026-04-02 13:05:02+02:00	subchapter headlines, not fully there yet but it will do	-3 +3	• short-form • main
1374cea	2026-04-02 21:38:47+02:00	cutting minimal	-12 +8	• short-form • main
438f9e1	2026-04-04 15:14:29+02:00	spelling	-46 +42	• short-form • main
aa768e1	2026-04-05 10:50:24+02:00	Writing this thesis in a source file, I can focus on the texts content, while leaving its presentation open to different formats. Source files can also easily be shared for editing or publishing (Coombs, DeRose and Renear, 1987).	-0 +2	• short-form • main
f85e2cc	2026-04-05 11:43:52+02:00	adding commits	-15 +6	• short-form • main
48cf029	2026-04-05 12:05:19+02:00	fix image rendering with captions	-10 +6	• short-form • main
98fe9c3	2026-04-10 15:09:48+02:00	big changes after last marloes feedback. merging history into here, rm from intro	-17 +24	• main
6b4fc37	2026-04-11 15:09:44+02:00	paper notes and rosas comments	-7 +7	• main
20b647e	2026-04-12 11:45:37+02:00	edits from marloes feedback and rosa	-18 +16	• main
44354de	2026-04-12 15:59:55+02:00	summary	-2 +1	• main
74f83db	2026-04-12			

```
         18:08:59+02:00
                                  clean up repo with sub dirs and  -177
                                  new paths                        +22     • main

5542603  2026-04-14               citation                         -13     • main
         18:06:39+02:00                                            +11

d542e5f  2026-04-15               small edits getting there        -7 +7   • main
         15:46:11+02:00

2092957  2026-04-15               spelllinng                       -12     • main
         16:00:08+02:00                                            +12

800e27f  2026-04-16               final reading                    -46     • main
         08:17:03+02:00                                            +47

1d37474  2026-04-16               quotes page numbers              -5 +6   • main
         16:23:40+02:00
```

chapter_2.md

id	date	message	diff	branch
8966f26	2025-12-12 10:50:22+01:00	divide introduction into subchapters, tentatively, add more granularity in the process of writing chapter 1	-0 +6	• short-form • main • notebook • citemeta
9.97e508	2025-12-12 11:21:53+01:00	map wordcount	-1 +7	• short-form • main • notebook • citemeta
3bdb8b0	2026-01-09 19:46:42+01:00	moved analysis and implications of metadata based rendering to chapter 2	-22 +26	• short-form • main • notebook • citemeta
0a450e1	2026-01-17 15:48:14+01:00	I am writing this text in markdown! restructuring some other paragraphs on the side	-12 +22	• short-form • main • notebook • citemeta
6e4fbe9	2026-02-01 17:16:55+01:00	hello marloes and claudio, i cleared up this chapter a bit so it hopefully becomes understandable where I want to go here. to myself: moved section about syntax and semantics to local notes	-27 +23	• short-form • main • notebook
7c8d895	2026-02-01 20:43:24+01:00	add inline metadata citation with at symbol + short hash of the commit i want to reference	-6 +4	• short-form • main • notebook
f82749f	2026-02-01 21:01:31+01:00	fix short hash of inline citation	-4 +4	• short-form • main • notebook
cc9a90b	2026-02-13 17:33:44+01:00	start chapter 2: replace semantics with legibility	-6 +7	• short-form • main • notebook
f43736d	2026-02-14 21:10:08+01:00	big chunk of dependencies section: access, storage, longevity, abstraction	-5 +109	• short-form • main • notebook
da32709	2026-02-15 12:44:50+01:00	figuring out bridges between lines: abstraction is a matter of legibility. through systems of simplification, standardazation and obscuring it creates directed illegibility (for the user). To enable abstraction, more code, data and metadata are required. Does this in turn increase legibility (for computers)?	-14 +23	• short-form • main • notebook
0cc553f	2026-02-15 17:58:54+01:00	writing commit messages is articulation work. in a collaborative project they inform others of the changes i made. but also when working alone, commit		

		messages coordinate process, let me go back to specific versions or help me remember steps i have taken	-1 +36	• short-form • main • notebook
22c404f	2026-02-15 18:05:04+01:00	change tentative notes on last subchapter proposing gitpub and context	-2 +6	• short-form • main • notebook
aea4449	2026-02-16 12:51:31+01:00	manually merge handwritten into main file chapter 2 part 1	-17 +14	• short-form • main • notebook
5dc47bf	2026-02-16 13:22:15+01:00	manually merge notebook to chapter 2 finished section on dependencies continue legibility	-5 +5	• short-form • main • notebook
1.354e93	2026-02-16 18:57:35+01:00	collecting outtakes, specifications, bridges from abstraction to legibility. might move out of this document	-12 +5	• short-form • main • notebook
92248c3	2026-02-16 19:18:26+01:00	legibility section becomes more legible but needs further editing	-5 +6	• short-form • main • notebook
d5761d2	2026-02-16 22:01:48+01:00	notebook chapter 2 manual merge complete	-13 +9	• short-form • main • notebook
0f8dfe2	2026-02-17 10:26:46+01:00	Collaborative work, next to following formal task descriptions involves many informal tasks: The preparation and sharing of files, discussion of changes made and planning of time for meetings or deadlines are all part of collaborative work, yet, they often remain unrecognized.	-5 +4	• short-form • main
fda1472	2026-02-17 11:07:01+01:00	Articulation work involves the managing of unanticipated contingencies in real time or in other words the work that enables other work to happen. This type of work appears across different professions. Secretaries, doctors or pedagogues tasks can to a large extend be considered articulation work, while an accountant will be less occupied with it.	-2 +2	• short-form • main
a5a31bf	2026-02-17 12:46:21+01:00	how exactly do collaborative and articulation work relate to another? collaboration connects distributed tasks, while articulation work manages the consequences of collaborations distributedness. This renders their relationship recursive.	-1 +1	• short-form • main
8ee6317	2026-02-17 13:01:13+01:00	the extend of this contextuality becomes apparent when we ask: illegible, to whom? and might that in turn legibility for someone else? contextuality also implies that illegibility is neither inherently positive or negative.	-1 +7	• short-form • main
e327b65	2026-02-17			

	16:22:00+01:00	Understanding illegiblity as contextual here implies that firstly, it is not total. Illegibility to some might mean legibility to others. And secondly, illegibility cant be treated as inherently positive or negative.	-4 +4	• short-form • main
29099c0	2026-02-17 16:25:10+01:00	Legibility of articulation work can increase its legitimacy and therefore be compensated or supported more adequately.	-3 +2	• short-form • main
66b2593	2026-02-17 16:29:10+01:00	And, as we have noticed before, granularity is high within todays collaborative production based on metadata.	-1 +1	• short-form • main
23ec38d	2026-02-17 16:33:51+01:00	ironically, the articulation work of writing parts of this text in commit messages which I am referencing in the markdown document by their hash, entirely distorts the text. my own process and writing have become unreadable.	-1 +1	• short-form • main
8461878	2026-02-17 17:21:27+01:00	notes on engineering problems	-2 +10	• short-form • main
a31563f	2026-02-17 20:56:03+01:00	Disagreeing with this misplaced concretism	-0 +6	• short-form • main
f32522e	2026-02-17 21:02:35+01:00	I'll try to be more clear: software is a model which tries to solve human communication and interaction through computer engineering. A model is a partial, situated representation, that should not be mistaken for reality. The reality of collaboration, which is messy, incoherent and unpredictable.	-3 +2	• short-form • main
ac9ea5d	2026-02-17 21:03:59+01:00	Collaborative work requires discourse and the ability to contain multitudes instead of fixed uniform identifiers. How can software support this reality, instead of trying to treat it as a problem that has to be solved through computational schemes?	-11 +1	• short-form • main
e9d7974	2026-02-17 21:33:42+01:00	communication summary notes	-1 +13	• short-form • main
292ca5a	2026-02-18 09:53:56+01:00	confusion with illegiblity	-6 +3	• short-form • main
2b4b40f	2026-02-18 20:04:12+01:00	legibility of articulation work for whom? legible to formal definitions of work? legible to data recording systems? legible to a public? legible to co workers?	-11 +13	• short-form • main
b788255	2026-02-18 20:25:59+01:00	A prominent example are git commit messages.	-17 +7	• short-form • main
f5d9d8b	2026-02-18 20:41:16+01:00	I am using gitpub for writing and publishing this thesis	-3 +8	• short-form • main
efe8dcc	2026-02-18 20:47:53+01:00	I could not include all the metadata i wanted inside the text because the university prescribes a strict 8k word		

		character limit for ma theses. I still wanted to include some metadata, so i build a special thesis version of gitpub: I can now make an inline citation of a git commit by referencing its commit hash.	-1 +1	• short-form • main
68f4a0d	2026-02-19 11:45:55+01:00	notes on gitpub - might move to conclusion e.g. postscript	-4 +6	• short-form • main
1399220	2026-02-21 10:25:44+01:00	chapter 2 reworked draft with shared context	-69 +43	• short-form • main
b2c154c	2026-02-27 10:34:25+01:00	franklins culture of complinace and distributed production	-4 +4	• short-form • main
b6f76ba	2026-02-28 22:15:35+01:00	modify comments marloes	-9 +18	• short-form • main
582a969	2026-03-01 20:05:31+01:00	Think of the in-text questions or signs of agreement you leave for a coworker in a shared document, the text messages you exchange to arrange an upcoming meeting or the discussion involved in combining different drafts.	-7 +20	• short-form • main
2feea3b	2026-03-02 15:50:32+01:00	change prescriptive to detached production. also changes in articulation work section	-15 +17	• short-form • main
73f0983	2026-03-08 12:19:59+01:00	transcribed annotations from paper copy	-66 +41	• short-form • main
6b00197	2026-03-08 15:59:12+01:00	spellcheck	-120 +427	• short-form • main
1f1dbf0	2026-03-08 18:49:53+01:00	rm some notes	-2 +1	• short-form • main
f2fdcb9	2026-03-27 17:38:29+01:00	various small	-20 +13	• short-form • main
568d7a9	2026-03-30 07:38:15+02:00	changes from paper notes	-30 +23	• short-form • main
bae0b6b	2026-04-02 12:28:56+02:00	Much can be said about the webs materiality: its grain (Chimero, 2015), how it is handmade (Carpenter, 2015), vernacular (Lialina, 2005) or just words (Jackson), powered by the sun (Decker, 2024) or a house we live in (Schwulst, 2018). But most important right now is to remember, that HTML documents build the core of the web.	-8 +5	• short-form • main
94540bf	2026-04-02 12:53:57+02:00	citation inside commit message citation	-1 +1	• short-form • main
fd8fd49	2026-04-03 17:08:19+02:00	cut cut cut	-42 +22	• short-form • main
4803d43	2026-04-04 11:34:31+02:00	remove ursula franklin and designs for compliance to slim linear to distributed production section, this is not the main focus	-12 +7	• short-form • main

0479b06	2026-04-04 14:53:45+02:00	their representation in software has always been challenging: a balance act between prescription and support, division, and conflation	-1 +1	• short-form • main
5f74940	2026-04-04 14:55:02+02:00	because it escapes the parameters which we use to classify what counts as work and therefore remains illegible to rationalized models	-2 +2	• short-form • main
475955a	2026-04-04 14:57:36+02:00	How, in comparison, does markup support articulation work?	-1 +1	• short-form • main
22b9d28	2026-04-04 15:00:17+02:00	the writing of source code comments, indicating process to collaborators and readers	-1 +2	• short-form • main
0608c37	2026-04-04 15:00:59+02:00	the use of semantic HTML, as the meaningful tagging of text is an act of instructing reading, co-writing or rendering	-1 +1	• short-form • main
438f9e1	2026-04-04 15:14:29+02:00	spelling	-46 +42	• short-form • main
2902dbf	2026-04-04 15:44:15+02:00	reference rosa	-1 +1	• short-form • main
cfbd33d	2026-04-05 11:07:22+02:00	wait, but this is metadata and you can read it, right?	-4 +4	• short-form • main
65ad931	2026-04-05 11:16:56+02:00	Since then articulation work poses a key challenge in the study and development of collaborative software (also known as computer supported cooperative work or CSCW) because this type of work is unpredictable and situated. And this is difficult to represent in simple user interfaces and predefined settings.	-2 +5	• short-form • main
f85e2cc	2026-04-05 11:43:52+02:00	adding commits	-15 +6	• short-form • main
e5d86d8	2026-04-10 17:50:45+02:00	intro big cuts, smaller changes around after last feedback	-7787 +28	• main
20b647e	2026-04-12 11:45:37+02:00	edits from marloes feedback and rosa	-18 +16	• main
5542603	2026-04-14 18:06:39+02:00	citation	-13 +11	• main
d542e5f	2026-04-15 15:46:11+02:00	small edits getting there	-7 +7	• main
2092957	2026-04-15 16:00:08+02:00	spelllinng	-12 +12	• main
800e27f	2026-04-16 08:17:03+02:00	final reading	-46 +47	• main
1d37474	2026-04-16 16:23:40+02:00	quotes page numbers	-5 +6	• main

chapter_3.md

id	date	message	diff	branch
c1ce199	2026-02-21 18:24:26+01:00	chapter 3 init. first thoughts mainly	-0 +36	• short-form • main
aa872b8	2026-03-03 21:32:56+01:00	why am i using git despite criticism? I am trying to be clear but the subject matter is not. But maybe thats the point, working against normalization of the tools we use everyday, remaining aware and critical of their ingrained belief systems and actions they prescribe. while also seeing the possibilities they open up.	-16 +39	• short-form • main
5d496ba	2026-03-04 21:11:36+01:00	its the end of the day and everything is chaos	-17 +58	• short-form • main
dc9fb68	2026-03-07 15:16:52+01:00	transcribed annotations from paper copy	-57 +42	• short-form • main
684de9b	2026-03-08 12:19:40+01:00	chapter intro	-15 +13	• short-form • main
6b00197	2026-03-08 15:59:12+01:00	spellcheck	-120 +427	• short-form • main
1f1dbf0	2026-03-08 18:49:53+01:00	rm some notes	-2 +1	• short-form • main
930b2e6	2026-03-25 20:38:56+01:00	In this moment GitPub is still work in progress.	-5 +4	• short-form • main
651eab9	2026-03-26 20:05:10+01:00	Another distinction I want to mention at this point, is that git itself is not a text editor. It can either be used in the text editor of your choice such as VS Code or Obsidian or through platform interfaces provided by GitLab, GitHub or Codeberg.	-4 +4	• short-form • main
7241100000	2026-03-27 17:34:01+01:00	before re writing design principles	-30 +30	• short-form • main
f59cf22	2026-03-27 17:35:55+01:00	rewrote design principles	-10 +15	• short-form • main
658c8e5	2026-03-30 08:44:24+02:00	paper notes done	-26 +20	• short-form • main
21c5cca	2026-04-02 12:53:00+02:00	subchapter headlines	-9 +6	• short-form • main
ae7034e	2026-04-02 20:57:34+02:00	cutting git details	-14 +13	• short-form • main
dd67455	2026-04-03 20:34:44+02:00	removed entire section about git cultural situatedness. no way to put		

		this into concise words.	-13 +7	• short-form • main
a5e3c8c	2026-04-04 14:48:50+02:00	small cuts	-15 +14	• short-form • main
438f9e1	2026-04-04 15:14:29+02:00	spelling	-46 +42	• short-form • main
69cdd75	2026-04-05 11:38:25+02:00	The fact that you can read this shows how git metadata serves both, its own operations and its users.	-1 +3	• short-form • main
f85e2cc	2026-04-05 11:43:52+02:00	adding commits	-15 +6	• short-form • main
48cf029	2026-04-05 12:05:19+02:00	fix image rendering with captions	-10 +6	• short-form • main
e5d86d8	2026-04-10 17:50:45+02:00	intro big cuts, smaller changes around after last feedback	-7787 +28	• main
20b647e	2026-04-12 11:45:37+02:00	edits from marloes feedback and rosa	-18 +16	• main
74f83db	2026-04-12 18:08:59+02:00	clean up repo with sub dirs and new paths	-177 +22	• main
5542603	2026-04-14 18:06:39+02:00	citation	-13 +11	• main
bd300ed	2026-04-15 15:44:54+02:00	xkcd comic footnote	-7 +7	• main
2092957	2026-04-15 16:00:08+02:00	spelllinng	-12 +12	• main
800e27f	2026-04-16 08:17:03+02:00	final reading	-46 +47	• main
1d37474	2026-04-16 16:23:40+02:00	quotes page numbers	-5 +6	• main

conclusion.md

id	date	message	diff	branch
1b07414	2026-03-29 17:05:57+02:00	init end	-0 +29	• short-form • main
25948ef	2026-04-02 08:14:10+02:00	what is the last sentence? what is further research relevance? how do people end texts	-20 +10	• short-form • main
f0d6856	2026-04-02 11:15:04+02:00	last paragraph. maybe this is too disconnected from the rest. eventually needs a reference for last sentence	-3 +2	• short-form • main
caa86a0	2026-04-03 21:04:19+02:00	last sentence, quote illich	-3 +1	• short-form • main
438f9e1	2026-04-04 15:14:29+02:00	spelling	-46 +42	• short-form • main
e5d86d8	2026-04-10 17:50:45+02:00	intro big cuts, smaller changes around after last feedback	-7787 +28	• main
95d8321	2026-04-15 12:46:48+02:00	final sentence	-7 +8	• main
800e27f	2026-04-16 08:17:03+02:00	final reading	-46 +47	• main
e6e61c3	2026-04-16 10:18:07+02:00	last sentence (again)	-1 +1	• main