Journal of the Institute of Conservation, 2019
Vol. 42, No. 2, 94–113, https://doi.org/10.1080/19455224.2019.1604398

Dušan Barok, Julie Boschat Thorez, Annet Dekker, David Gauthier and Claudia Roeck

Archiving complex digital artworks

Abstract
The transmission of the documentation of changes made in each presentation of an artwork and the motivation behind each display are of importance to the continued preservation, re-exhibition and future understanding of artworks. However, it is generally acknowledged that existing digital archiving and documentation systems used by many museums are not suitable for complex digital artworks. Looking for an approach that can easily be adjusted, shared and adopted by others, this article focusses on open-source alternatives that also enable collaborative working to facilitate the sharing and changing of information. As an interdisciplinary team of conservators, researchers, artists and programmers, the authors set out to explore and compare the functionalities of two systems featuring version control: *MediaWiki* and *Git*. We reflect on their technical details, virtues and shortcomings for archiving complex digital artworks, while looking at the potential they offer for collaborative workflows.

Keywords
art documentation; archiving; preservation; media conservation; complex digital art; version control

Introduction

In 2017 UBERMORGEN, a Swiss-Austrian-American artist duo, submitted a selection of their works to be taken into the collection of LIMA, a platform for media art in Amsterdam. UBERMORGEN's main body of work consists of internet art, installation, video art, photography, software art and performance, and uses the convergence of digital media to produce and publish online and offline. Most of their early works were media hacking projects using low-tech tools to reach large audiences. As part of LIMA's event series *Cultural Matter*—about the preservation, presentation and distribution of digital art—researcher and artist Julie Boschat Thorez was asked to select one of the artworks as a starting point to discuss and contextualise the art historical and technical importance of their works. She selected *Chinese Gold* (2006–ongoing), a project on the phenomenon of industrial-scale gold mining in the online video game *World of Warcraft* and operated from China. The project seemed to be the best candidate because some time had passed since it was initiated and it had been exhibited several times in multiple ways. It also represents the type of work UBERMORGEN is known for, involving a lot of research from which different works develop. Hence, a careful consideration of the contexts and history was needed to understand the meaning of the work and any of its subsequent preservation measures.

Some of the first questions focussed on the different elements of the work: what was the difference between the work and the documentation? What should be considered as research or contextual material and what should be seen as the actual work? Would it be necessary to make such distinctions? How to preserve and present a work that does not have a determined form? To provide some answers to those questions, many discussions took place with the artists, who also gave access to their extensive

(Received 16 January 2019; Accepted 3 April 2019)

archive of the project, which complemented the information already gathered through online research.

The first step was to list all the work's elements and the multiple presentations, both online and offline, and with this information in place, and together with the help of the artists, an extensive description of the project could be made as well as a choice about which elements to keep. *Chinese Gold* can be described as a complex digital artwork[1]—a heterogeneous assemblage, from which the various elements can be combined, composed and compiled in different ways, at different times and locations (online and offline) and by different people.[2] The next step involved the current research team thinking about all these types of documents, documentation and other materials that were collected and how to archive them in a way that would do justice to the ever-changing nature of the work, in such a way that it would also be useful for future preservation projects.

The transmission of an artist's research, the documentation of any changes in the presentation of the work, and the motivation behind each display are of importance to the continued preservation, re-exhibition and future understanding of the work. However, it is generally acknowledged that existing digital archiving and documentation systems used by many museums, such as *The Museum System* or *Adlib*, are not suitable for these particular kinds of artworks due to their rigidity. For example, one cannot easily represent changes in the evolution of the work, nor show the relations between its different elements.[3] Even though standard schema can be adjusted to specific needs, most applications are developed by commercial companies and this kind of flexibility comes at a price. Moreover, proprietary solutions usually have high licensing costs and lack a more open model of governance. To move away from these systems and, more importantly, looking for an approach that can easily be adjusted, shared and adopted by others, the research team focussed on open-source alternatives that would also enable collaborative working to facilitate the (future) sharing and changing of information. With this choice, the team also hoped to build alliances with existing communities of practice, testing and using open-source alternative documentation systems.

While the research presented here is not focussed on preservation of the different elements of an artwork, it does provide a means to discuss alternative ways of documenting the changes and different versions of an artwork that take place in its biography and exhibition history, which are important to consider for both future redisplays and preservation of the work.[4] In this sense the research builds on discussions around the value of allographic provenance and versioning in relation to complex artworks.[5] As an interdisciplinary team of conservators, researchers, artists and programmers, we are determined to explore and compare the functionalities of two systems featuring version control and web interface: *MediaWiki* and *Git*, and its associated repository manager *GitLab*.[6]

Another reason for taking this direction relates to several early steps that were done in archival and conservation practices to test the usefulness of wiki-based platforms and version control systems for documenting artworks.[7] This research could be useful to further the discussion and expand the working methods and possibilities.

The study then focussed on how the version control elements of these systems encourages collaboration between conservators, curators and/or artists in archiving complex digital artworks by reflecting on the technical details of the different systems, their virtues and shortcomings.

Chinese Gold: the data

UBERMORGEN were founded in 1995 by Lizvlx and Hans Bernhard. Together they developed a series of landmark projects in digital art,

1 In contemporary art conservation, complex artworks have been considered to be installations and other types of work with one or more of the following elements: variable form (e.g. involving non-dedicated, replaceable components), conceptual or otherwise immaterial features crucial for re-exhibition, being process-based and being open-ended—see, among others, Pip Laurenson, 'The Conservation and Documentation of Video Art', in *Modern Art: Who Cares?* (Amsterdam: Foundation for the Conservation of Modern Art, 1999).

2 A similar way of working, and thus set of challenges, is inherent in many digital artworks, including *mouchette.org* by Martine Neddam, the practice of Young-Hae Chang Heavy Industries or Lynn Hershman Leeson. For more information see, for example, Annet Dekker, Gabriella Giannachi, and Vivian van Saaze, 'Expanding Documentation, and Making the Most of the "Cracks in the Wall"', in *Documenting Performance. The Context and Processes of Digital Curation and Archiving*, ed. Toni Sant (London/New York: Bloomsbury, 2017), 61–78; and Annet Dekker, *Collecting and Conserving Net Art: Moving beyond Conventional Methods* (Oxon: Routledge, 2018).

3 Cf. Annet Dekker and Patricia Falcão, 'Interdisciplinary Discussions about the Conservation of Software-Based Art. Community of Practice on Software-Based Art', *PERICLES*, March 2017, http://www.tate.org.uk/download/file/fid/108032 (accessed 23 September 2018); Deena Engel and Glenn Wharton, 'Managing Contemporary Art Documentation in Museums and Special Collections', *Art Documentation: Journal of the Art Libraries Society of North America* 36, no. 2 (2017): 293–311.

4 On an artwork's biography see, for example, Renée van de Vall et al., 'Reflections on a Biographical Approach to Contemporary Art Conservation', in *ICOM-CC 16th Triennial Conference—Lisbon 2011*, ed. J. Bridgland (Almada: Critério, 2011), 1–8.

5 For more information see, for instance, Renée van de Vall, 'Documenting Dilemmas. On the Relevance of Ethically Ambiguous Cases', *Revista de*

including *Vote-Auction* (2000), a media performance involving a false site where Americans could supposedly put their vote up for auction, and *Google Will Eat Itself (GWEI)* (2005, in collaboration with Alessandro Ludovico and Paolo Cirio), a project that proposed using Google's own advertising revenue to buy up every single share in the company. UBERMORGEN use their projects to create alternative narratives so as to critically reflect on networked culture, and to reveal something of the inside and downside of a 'post-truth' society.

Chinese Gold (2006–ongoing) reflects on the process of gold mining within the multiplayer online role-playing game *World of Warcraft*, which consists of generating an excess of in-game money to be then sold via online trading platforms. The artwork revolves around a partially fictive research project into the phenomenon to underline how virtual currency trading recreates familiar geometries of industrial good production in a globalised economy. Spanning over a decade, *Chinese Gold* consists of a mix of research, original computer files and documentation, and is presented in multiple ways in which methods such as appropriation, storytelling and remixing are used in parallel. As such, the project is constantly evolving, growing and in flux. Consequently, the documents and documentation about the work are scattered and at first sight there seems to be little logic to the structure of the digital storage of the project as supplied by the artists. As such any current re-construction of *Chinese Gold* would not be a straightforward procedure. Since the process of creation and re-creation of the work is heterogeneous and involves a certain level of improvisation that continually re-negotiates its structure, its documentation archive should ideally reflect this approach, while also enabling the re-visioning of past iterations and information.

One of the places where some information about the work is presented is the dedicated web page made by the artists for the project which indexes the work's constitutive elements: the research, literature on the project, and information about past exhibitions and partners.[8] It is specified that the work consists of an ensemble of four image series—'Belgrade Series' with eight images, 'Blue Series' with seven images, 'MTV Series 3' with six images and 'MTV Series 4' with six images—and a 14:12min-long video made by repurposing online game graphic engines to create a cinematic-like production in an art genre known as 'Machinima'. Between 2006 and 2017 the work was shown by several art galleries and organisations with each iteration being different in response to both the exhibition space and the *particular curatorial theme* (Fig. 1). In conversation with each

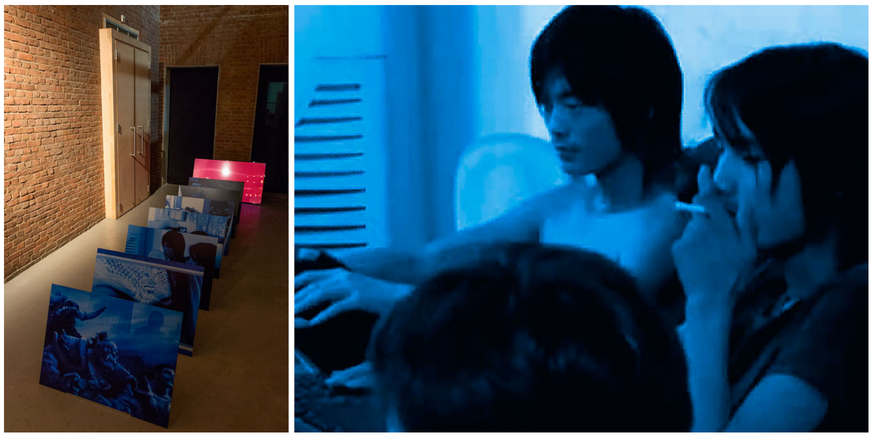


Fig. 1 Left: Installation view of *Chinese Gold*, part of Cultural Matter, LIMA, Amsterdam, 2017. Right: detail from *Chinese Gold, Untitled 1-7 (Blue Series)*, 2006. Courtesy of artists

História da Arte (2015): 7–17; and Dekker, *Collecting and Conserving Net Art*, 127–39.

6 This research began with the 'Versioning the Networked Archive' workshop initiated by Annet Dekker and Miglena Minkova at the *Re:learn* summer school at Poortgebouw, Rotterdam, 29 August–2 September 2017 and was continued during the 'Collaborative Archiving of Digital Art' workshop at Digital Methods Winter School at the University of Amsterdam, 8–12 January 2018.

7 In the case of media arts but also more widely in contemporary arts, with, for example, wiki-based documentation initiatives at San Francisco Museum of Modern Art (SFMOMA), ZKM | Center for Art and Media Karlsruhe and New York University (NYU).

8 http://www.ubermorgen.com/2007/projects/Chinese_Gold/ (accessed 17 March 2019). The last update dates back to 2011, and reflects the state of the project at the time.

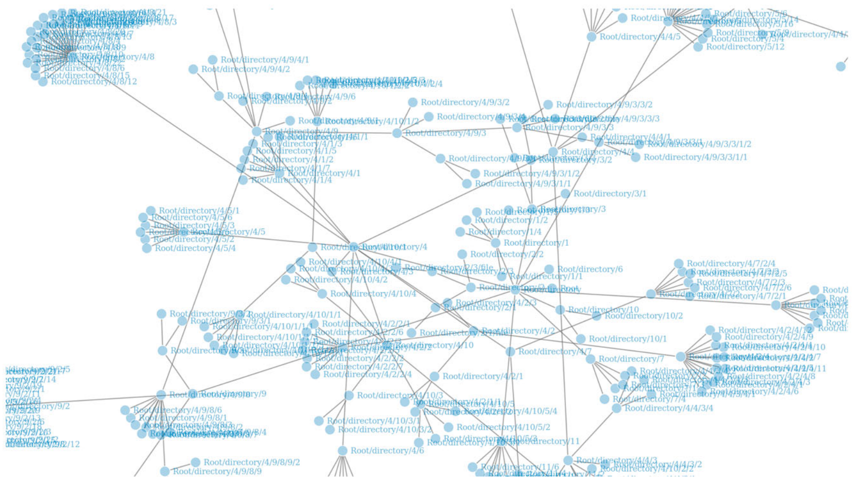


Fig. 2 A diagram of the folder structure of artists' archive. Courtesy of Judith Hartstein.

curator the artists would select and (re)produce specific items to represent the work. In the absence of detailed guidelines the work has thus been materialised in many different ways that have been influenced further by factors such as technical possibilities, type of audience, curatorial decisions and so forth.

As a test case, UBERMORGEN gave the team access to their archive of *Chinese Gold*, which amounted to more than 10GB of data (see Fig. 2). To simplify the testing of different systems we decided to focus on the oldest folder in the *Chinese Gold* archive: 'CHINESE_GOLD_2006'. This choice was also motivated by the diversity of its file formats, which was representative of the overall archive. 'CHINESE_GOLD_2006' consisted of 337 files—41 of them hidden files—in 66 directories, and most of the files were images (see Table 1). The images vary in format, colour, size and cropping, revealing that the series was created in a long process of filtering images to fit the project's narrative.

Alongside press articles, e-mails and other research items, the folders also contained the work's image files in multiple versions. The team had

Table 1 The distribution of file extensions by type.

Count	Extension
117	jpg
82	png
67	none (folder)
57	tif
41	DS_Store (hidden Mac OS files)
12	zip
6	html
5	doc
4	mov
3	gif
2	rtf
1	dv
1	iMovieProj
1	iMovieProject
1	mp4
1	odt
1	pdf
1	xls

learnt from the artists that they select versions of these files for different publications and exhibition contexts, which is reflected in the naming of the folders, for example: 2014_CONTEMPORARY_ISTANBUL_HIGH_RES, IMAGES_FOR_CULTURAS_2008_CATALOGUE, NIMk_exhibition and so on. The image files exist in multiple copies and in various sizes and resolutions. Besides this, the folders contain the artists' research on the topic including downloaded images and news articles, e-mail correspondence and website snapshots. Although these items have yet to be used by the artists as part of any presentation, they are potentially useful in informing researchers about their methods and the project's development. The team also collected additional information to fill some of the gaps in the archive thought necessary for future research including, for example, wider information about the contexts of the exhibitions in which the work was presented.

Version control: an introduction

Finding a coherent and structured way to organise and control revisions has always been at the core of archival practices. In the era of computing, these fundamentals became even more urgent and complex, and stimulated the development of 'version control systems' (VCS). Version control systems typically check the differences between versions of code or text as well as of file structures. By archiving through an agency of timestamping and the author's name, and by making ongoing versions of a project available, a VCS allows multiple people to work on elements of a project without overwriting each others' modifications and variation changes. Changes that are made can easily be compared, restored or, in some cases, merged. A VCS is also a necessary archival allowance for computer-based art due to the frequency of memory-intensive files and variations on those files within the evolution of those artworks.

1 *Wiki*

Wikipedia is perhaps the most well-known example of an online platform using version control in its 'page history'. Its difference engine is based on character-by-character analysis and allows users to check the differences between new and previous versions of pages. However, developing version control began in the late 1960s,[9] when it was primarily used for isolating something that didn't function as expected and the code needed to be revised; by tracing the history one could find the bug that caused the problem.[10] In 1995 Ward Cunningham developed *WikiWikiWeb* as the first user-editable website with the main idea that it would facilitate quick communication between software developers.[11] Another popular wiki was *MediaWiki* from 2002, developed by a Wikipedia volunteer Magnus Manske, and based on the wiki software 'UseMod Wiki', created by Clifford Adams for *Wikipedia*.[12] The wiki software was 'originally developed for project documentation and collaboration around Agile software development'.[13] One of the main changes in *MediaWiki* was that anyone could review the changes that were made, allowing easy detection and undoing of any perceived mistakes.

2 *Git* and *GitLab*

In contrast to these early wikis, Linus Torvalds, creator and principal developer of the Linux kernel, designed *Git* not to document software projects nor facilitate communication between developers per se, but as a platform on which to directly write and develop source code. Before creating *Git*, Torvalds relied on a proprietary product called *Bitkeeper* for the source code management (SCM) of the Linux kernel. Following the breakdown of relations between the Linux developer community and the company

9 See Aymeric Mansoux, 'Sandbox Culture' (PhD thesis, Goldsmiths University, London, 2017), 343.

10 See Marc J. Rochkind, 'The Source Code Control System', *IEEE Transactions on Software Engineering* 1, no. 4 (1975): 364–70.

11 Hence the name 'wiki', which is Hawaiian for quick.

12 See Jennifer Joline Anderson, *Wikipedia: The Company and Its Founders* (Edina, MN: ABDO Publishing Company, 2011).

13 Matthew Fuller et al., 'Big Diff, Granularity, Incoherence, and Production', in *Memory in Motion: Archives, Technology, and the Social*, ed. I. Blom, T. Lundemo and E. Røssaak (Amsterdam: Amsterdam University Press, 2016), 90.

owning *Bitkeeper*, Torvalds created *Git* in 2005.[14] With *Git*, he intended to improve the performance of existing SCM systems, especially in relation to speed when applying patches and updating versions; to simplify SCM design and make it fully distributable; and to support non-linear development such as parallel branching.[15] *Git* makes it possible to write code in a decentralised and distributed way by encouraging 'branching', that is, a mechanism that allows various people to work on multiple versions of a work at the same time. As well, and of particular interest to our research, branching facilitates the tracking and auditing of changes.

14 See Scott Chacon and Ben Straub, *Pro Git. Everything You Need to Know About Git*, 2nd edn (New York, NY: Apress, 2014), 13.

15 Chacon and Straub, *Pro Git*, 11.

In 2007 GitHub Inc. started hosting *Git* repositories (or repos) online at github.com. Interestingly, as such *GitHub* quickly evolved into a site of so-called 'social coding',[16] and rather quickly the collaborative coding repositories became used for widely diverse needs: from software development to writing licence agreements, sharing Gregorian chants and announcing a wedding—anything, as it turns out, that needs quick information sharing.

16 Cf. Fuller et al., 'Big Diff, Granularity, Incoherence, and Production'.

For the project, the team focussed on *GitLab* as it is an open-source alternative to *GitHub*. While *Git* is the core system for version control, online hosting platforms such as *GitHub* and *GitLab* provide web interfaces to view the content of a given repository, including the changes applied to it, and the collaborators that have made these changes over time. While there is a panoply of free hosting platforms we could have used instead of *GitLab*, such as *GitHub*, *Bitbucket* and *SourceForge*, the decision to host the repository on *GitLab* was based on a compromise between a sense of political integrity in the advocacy of free software and the practical lack of time for deploying and customising an alternative self-hosted *Git* platform on our own dedicated servers.[17] It is important to note though that while most hosting platforms currently available online offer free hosting plans, they are in most cases built on top of closed-source infrastructures and opaque business models. As Aymeric Mansoux remarked, in the case of *GitHub*, 'the employees and founders of the platform, whose core components are strategically closed-source, are the ones to decide what projects and behaviours are acceptable',[18] an observation that becomes profound considering the extremely closed business model of *GitHub*, now owned by Microsoft.[19] A paradox thus emerges from these platforms: they are built on top of an open-source infrastructure—*Git*—and typically demand that hosted projects be similarly open-source, but their own meta-infrastructure is unquestionably closed. This contradiction, which Mansoux brilliantly analyses, is one of the main reasons the team decided to use *GitLab*, which is open-source, instead of *GitHub*.[20] As briefly mentioned earlier, building alliances with open-source communities of practices, such as software development and maintenance, offers the means to tailor the software depending on one's needs, and in so doing, allows for fine grained and sustainable functional customisations as open-source software is freely available and allows system design and infrastructural decisions to be made and acted upon in diverse partnerships.

17 There are other self-hosted platforms such as *Gitea*, *stagit* and *Gogs*, yet the choice of using free *GitLab* seemed a more viable option given that our initial work was conducted as part of a workshop. Having said this, our findings would still apply to most of these self-hosted platforms as they present, more or less, the same functionality as *GitLab*.

18 Mansoux, 'Sandbox Culture', 353.

19 See, for example, Tom Preston-Werner, 'Open Source (Almost) Everything', 22 November 2011, http://tom.preston-werner.com/2011/11/22/open-source-everything.html (accessed 22 September 2018); Satya Nadella, 'Microsoft + *GitHub* = Empowering Developers', *The Official Microsoft Blog*, 4 June 2018, https://blogs.microsoft.com/blog/2018/06/04/microsoft-GitHub-empowering-developers/ (accessed 22 September 2018).

20 Mansoux, 'Sandbox Culture'.

Furthermore, a given repository can be configured to ensure that only specific known collaborators can view, read and write to it, and that since *Git* is a decentralised system decoupled from any online hosting platform, any *Git* repository can be migrated from a given platform to another at any time.

Git and *MediaWiki*: a system-level comparison

Before detailing the experiments made with UBERMORGEN's *Chinese Gold* archive, it is worth comparing the basic system-level principles underpinning both *Git*/*GitLab* and *MediaWiki*, starting with their versioning models and synchronisation techniques to highlight the salient differences between both systems.

1 Versioning models

MediaWiki is a web-based publishing system that is hosted on a server. Programmed in PHP (Hypertext Preprocessor programming language) it supports different SQL (Structured Query Language) databases, of which the research used the variant MySQL for the project. However, *MediaWiki* is not merely a database, but is a document-oriented VCS based on the notion of linked pages. *MediaWiki* pages are dynamically transformed into html pages as they are authored online and *MediaWiki* places a strong emphasis on the collaborative creation and maintenance of page content, with a range of user rights applicable to a given user account by the *MediaWiki* administrator. The administrator provides different degrees of access and control to the system, such as the capacity to create, edit and delete pages, or amend the broader structure of the online interface through, for example, the provision of information presented on the sidebar. Each page contains its own discussion tab, as well as the complete edit history of the page, consisting of all time-stamped edits performed over time, and which are comparable in a manner similar to Unix' *diff*.[21] Beyond this, there is no branching as in having multiple concurrent versions or splitting capabilities for a page—the latest canonical version of a page is presented by default to the user, using a custom mark-up syntax called 'wikitext'. Moreover, the open-source nature of *MediaWiki* has allowed for an ecosystem to develop where a panoply of extensions are available, providing a range of functionalities beyond those of the base package.

21 The Open Group, 'diff', in *The Open Group Base Specifications Issue 7, 2018 edition* (IEEE, 2018), http://pubs.opengroup.org/onlinepubs/9699919799/utilities/diff.html (accessed 22 September 2018).

It is worth noting that *MediaWiki* versions its pages differently than files that are imported to it such as images and videos. Pages are entries stored in its SQL database as wikitext and digital files stored in a structured directory of uploaded files. An imported file cannot be changed within *MediaWiki*. If the file has to be changed, it has to be done outside of *MediaWiki* and then re-imported. Consequently, *MediaWiki* replaces the old file with the new file and the old file is kept as the old version and can still be accessed after it has been updated.

Git, on the other hand, is a VCS that works directly with the files and directories of a given computer's operating system. *Git* is a collection of Unix-type programs written in the C programming language. These programs need to be installed locally on a given computer and are usually evoked using the computer's command-line interface (CLI). There exist graphical user interfaces (GUI) for *Git*, though the team used the CLI for the experiments. A directory of a computer can be put under version control at any time, meaning that all the files and folders the directory contains will be tracked over time by *Git* from its initial state when the directory is first put under version control. *Git* keeps track of the content of files and whenever a file is changed, the system makes a version of it and records its state —the version—in a local repository that resides in the same directory as the one under version control (in a special '.*Git*' subdirectory). In this way, *Git* works directly on a given archive's files and their local file system, unlike *MediaWiki* which constructs a database of the archive's page content on a centralised server.

Matthew Fuller et al. suggested that *Git* tracks changes through a 'Big Diff', that is, accounting for versions of files by only tracking differences between them in a similar way to the *diff* and *patch* Unix commands.[22] As mentioned above, *MediaWiki*'s internal versioning system is indeed based on a *diff* and thus stores differences, or deltas, in its database.[23] Yet, contrary to what Fuller et al. suggest, *Git* takes an initial 'snapshot' and not a *diff* of the state of the archive's directory recursively—that is, recording all the content of files and subdirectories it may contain—which subsequent states of the directory will be compared to. This difference between each system is made explicit in *Git*'s official manual:

22 Cf. Fuller et al., 'Big Diff, Granularity, Incoherence, and Production'.

23 'Manual: MediaWiki Architecture', *MediaWiki*, https://www.mediawiki.org/wiki/Manual:MediaWiki architecture (accessed 23 September 2018).

> 'The major difference between *Git* and any other VCS (*Subversion* and friends included [i.e. *MediaWiki*'s internal versioning model for pages]) is the way *Git* thinks about its data. … These other systems … think of the information they store as a set of files [or pages] and the changes made to each file over time (this is commonly described as delta-based version control). *Git* doesn't think of or store its data this way. Instead, *Git* thinks of its data more like a series of snapshots of a miniature filesystem.'[24]

24 Chacon and Straub, *Pro Git*, 13–4.

But what does this 'snapshot' method mean? Simply put, *Git* records the content of files as versions where the whole content of the file is stored, rather than solely recording the parts (deltas) that have changed between earlier and later versions. However, *Git* does use a custom 'delta compression' when compressing or packaging its repository, but only when the repository size exceeds a certain pre-configured threshold or, alternatively, when peer-to-peer repositories need to be synchronised over a network. By default, *Git* does not delta compress its objects—it works with 'loose objects' in *Git*'s jargon—unlike *MediaWiki*, which does so necessarily each time a page is changed. Also it is worth noting that *Git* works with the content of files rather than files as such. It indexes the content of files using cryptographic checksums, or hashes, and abstracts away from its own internal repository representation the peculiarities of the underlying file system the files are embedded in. In other words, *Git*'s internal repository is a miniature file system that is indexed by content-based cryptographic ciphers. *Git* can thus be framed as a content-addressable repository rather than a file-oriented VCS.

2 Synchronisation

Another important aspect of versioning is synchronisation. Important differences exist here between *Git* and *MediaWiki*. *Git* is a peer-to-peer system whereas *MediaWiki* is modelled as a client–server one. The main difference between these models lies in the fact that a client–server architecture is a centralised model that keeps a master copy of a repository or archive on a single server. All clients are synchronised through this server, meaning that each client/collaborator needs to log into the centralised server using their respective credentials in order to contribute changes to that master. The model does not allow for 'disconnected operations' as clients need to remain online at all times in order to be synchronised with the master copy of the centralised server.

Instead of having clients commit changes to a central remote repository, in *Git* changes are committed to a self-contained local repository. These changes are termed 'commits' in *Git*'s nomenclature. There is no hard distinction between clients and servers involved: servers are clients and clients are servers, meaning that any collaborator having a copy of a repository can share it locally with anybody else of interest. There is no need to be connected to the internet and no *a priori* credentials are required to share repositories. Moreover, when a repository is first copied from a peer in an operation also known as 'cloning' a repository, the entire version history of the repository is cloned as well, making every cloned instance of a repository a coherent one and an actual back-up of the archive as a whole. When changes are made to a peer's local repository in isolation, these can be committed to the peer's local repository first and then 'pushed' to another peer's repository over a shared local network. This operation is called a 'push request'. Alternatively, a peer can 'pull' committed changes from another peer's local repository in what is called a 'pull request'. This peer-to-peer push/pull logic is how *Git* synchronises peers' archives with one another in a distributed manner. It is important to note that this logic is a combinatorial one, that is, the potential of distributed repositories to produce new states, and thus new versions, grows exponentially as the number of peers increases.

MediaWiki provides the possibility for exports of the whole database, yet it lacks a synchronisation mechanism between various instances of a single database.

One last difference to highlight is how 'commits' are handled in both systems. With *Git*, a user can decide which changes and how many changes he/she commits at once. In other words, *Git* changes can be 'cherry-picked' within files themselves—changes at the word level, for instance—and also changes within different files can be grouped as a single commit. With each commit, the user is forced to add a commit message in order to explain the reason for the commit. With *MediaWiki*, changes can only be saved per page or per page section and changes on a page are saved as a group, when the 'save changes' button is clicked on the *MediaWiki* interface. These group changes are visible as a version under the 'view history' tab of the page.

In fact, a thorough comparative analysis of what 'versioning' means to both systems, and more importantly, how it is effected operationally, would require an entire article of its own. Yet, as an attempt to push this research forward, we describe preliminary research into the selected two VCSs in what follows, and discuss possible implications for archival practices that centre on the documentation of networked and processual artworks.

Experiments and findings

As mentioned, to focus the team's experiments on the 10GB of data in the UBERMORGEN archive, we selected the oldest directory of the archive 'CHINESE_GOLD_2006'. The aim of the experiment was to evaluate the affordances of *Git*/*GitLab* against *MediaWiki* for the preservation and presentation of what is a complex digital artwork. Four aspects were compared.[25]

1) *File and storage management*: does the platform support the archiving and access to digital artwork? How convenient and adaptable is the data/file ingestion, operations and retrieval of files? Can it handle all files with respect to format and size? Can it display encoded files such as images and videos? How portable is the system?
2) *Metadata and provenance*: how does the system handle provenance and metadata? Does it preserve the original file name, creation date and path? Does it display metadata produced by recording devices, such as EXIF for cameras and audio recorders? Does it record information about who manipulated the file, when and why? Does it keep versions?
3) *Context, presentation, curation*: how does the platform support the exhibition and display of the artwork itself? How does it prepare archives for this? Does it provide the means for relating and assembling items and their associated documentation?
4) *Collaboration, usability*: how does the platform accommodate users? What user-roles does it offer? How does it support interaction and collaboration between users? Is the system accessible in terms of economic and technical engagement? Is it hard to maintain?

Other aspects that need further research are beyond the scope here but include issues around file fixity, data integrity and interoperability, and integration with other systems.

25 The aspects selected all allude to the features described in the model for an Open Archival Information System (OAIS). The package consists of 'Content Information' (data object), further described by 'features that determine information integrity and deserve attention for archival purposes', namely reference, provenance, context and fixity. See Christopher A. Lee, 'Open Archival Information System (OAIS) Reference Model', in *Encyclopedia of Library and Information Sciences, Third Edition*, ed. Marcia J. Bates and Mary Niles Maack (Boca Raton, FL: CRC Press, 2009), 4026–7.

1 File and storage management

1a MediaWiki

While it is primarily a text-oriented platform, *MediaWiki* is capable of storing files ingested into the system through the process of uploading. Files are stored in a dedicated directory structure where they can be accessed under their assigned URLs that contain their original file names.

This functionality is geared towards archiving individual files with the purpose of embedding them in content pages. However, dealing with UBERMORGEN's archive of hundreds of files, the first challenge was to optimise the ingest procedure. There is no means in the base package to upload multiple files at once so a partial resolution to this was by installing the 'UploadWizard' extension that allows for batch uploading of files as well as batch editing of their metadata by changing the pre-sets in *MediaWiki*'s 'LocalSettings.php' configuration file. The extension speeds up the process of ingestion, but does not support uploading batched folders—each folder has to be uploaded separately. Problems occur when files that have the same name need to be uploaded from different folders because the system is only able to register files with distinct names.

MediaWiki also refused to accept certain file types as for security reasons the system keeps two blacklists to prevent users from uploading executable files (i.e. as an anti-spam precaution). To include certain file types as accepted formats, the blacklists have to be overridden manually in the configuration file, and to allow for the upload of otherwise unsupported file types, the range of accepted file formats also has to be reconfigured.[26]

26 See https://www.mediawiki.org/wiki/Manual:Configuring_file_uploads#Configuring_file_types (accessed 23 September 2018). For blacklists and allowing for the upload of otherwise excluded executables, see https://www.mediawiki.org/wiki/Manual:$wgFileBlacklist and https://www.mediawiki.org/wiki/Manual:$wgMimeTypeBlacklist (both accessed 22 September 2018).

The version history of an imported file is available in the 'File history' section and similarly to the text-based VCS in *Git*, the comparison of encoded files such as images and binary formats cannot be directly visualised. The file history merely shows that one image has been replaced with another and provides access to the former versions.

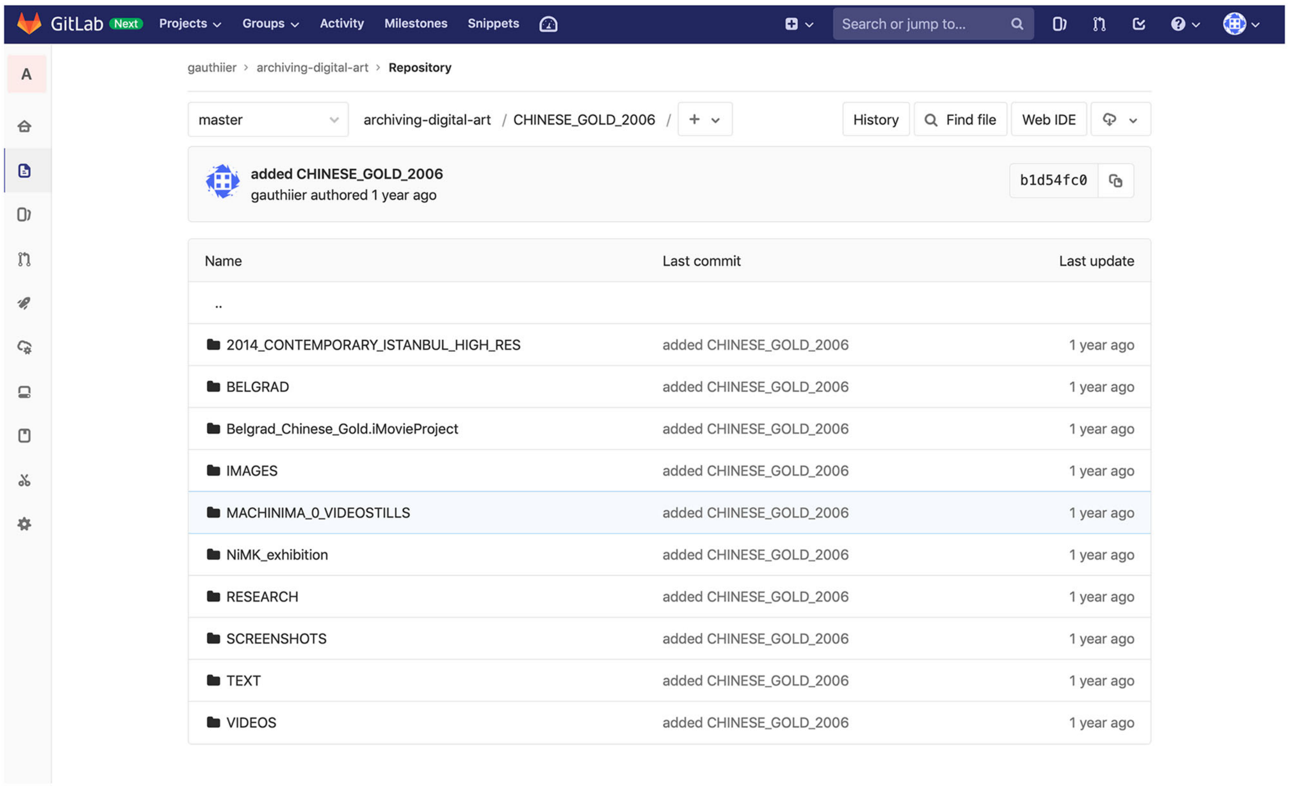
MediaWiki allows for the automatic previewing of images and videos on the respective 'File' pages for images and on request through file-embeds within the wiki pages. While JPEG images were rendered successfully, TIFF images were not at the first attempt as another setting in the configuration file needs to be adapted in order to render TIFFs.[27] TIFF images are important as TIFF is an archival format in contrast to JPEG. In order to enable the embedding of videos, the extension 'EmbedVideo' was installed. It is important to mention that these embeds have to be arranged manually.

27 TIFF images can be rendered after setting $wgTiffThumbnailType variable in the config file (*LocalSettings.php*)—see https://www.mediawiki.org/wiki/Manual:$wgTiffThumbnailType (accessed 22 September 2018).

Besides these shortcomings, the storage of data on a centralised server is suboptimal in preservation terms, particularly when compared to the distributed nature of *Git*. When following the digital preservation principle of LOCKSS (Lots Of Copies Keep Stuff Safe), the *MediaWiki* server would need to be cloned at different places. At the time of writing, an extension that would allow the cloning of parts of a *MediaWiki* server to a client computer appeared unavailable, but the team felt this would eventually change and as such an extension will be made.

1b Git

The first part of experimenting with *Git* was to put the entire content of the archive in a repository (Fig. 3). *Git* was originally created for code management, that is, essentially text files, which tend to be smaller than image and video files. When images and videos are managed with *Git*, each time a file is changed, a new version of the file is stored. In this way, the data volume quickly expands. As the UBERMORGEN archive contained rather large files (*.zip, *.mov, iMovie source projects, etc.) the team had to use a special *Git*



gauthiier › archiving-digital-art › Repository

master archiving-digital-art / CHINESE_GOLD_2006 /

added CHINESE_GOLD_2006
gauthiier authored 1 year ago

b1d54fc0

Name	Last commit	Last update
..		
2014_CONTEMPORARY_ISTANBUL_HIGH_RES	added CHINESE_GOLD_2006	1 year ago
BELGRAD	added CHINESE_GOLD_2006	1 year ago
Belgrad_Chinese_Gold.iMovieProject	added CHINESE_GOLD_2006	1 year ago
IMAGES	added CHINESE_GOLD_2006	1 year ago
MACHINIMA_0_VIDEOSTILLS	added CHINESE_GOLD_2006	1 year ago
NiMK_exhibition	added CHINESE_GOLD_2006	1 year ago
RESEARCH	added CHINESE_GOLD_2006	1 year ago
SCREENSHOTS	added CHINESE_GOLD_2006	1 year ago
TEXT	added CHINESE_GOLD_2006	1 year ago
VIDEOS	added CHINESE_GOLD_2006	1 year ago

Fig. 3 Artists' file archive on *GitLab*.

subsystem called Large File System (LFS).[28] This allows large files to be uploaded to a specific remote LFS repository and modifies the *Git* repository so that references to the file instances will be replaced by pointers to the remote LFS server. In this way, when the repository is cloned or when changes are fetched from it, *Git* treats LFS pointers in a distinct way and will not, for instance, fetch the whole content of these files from the LFS repository but only copy their pointers, thus speeding up the cloning process. For the purpose of the project, this made sense since there were no changes to be made to the content of the files as such. LFS works with file extensions, and after an analysis of the files present in the archive, the LFS was instructed to track the following extensions: *.zip, *.mov, *.tif, *.png, *.jpg, *.jpeg, *.JPG, *.mp4, *.iMovieProj and *.psd. These were written to a special '.gitattributes' configuration file on the local repository while it was being created. When the repository was finally configured with LFS on one of our machines, *Git* pushed it to the *GitLab* hosting platform so that it could be shared amongst the team. Once in *GitLab* it was tagged with version 'v0' designating the unmodified state or version of the archive.

In short, to summarise the process of creating the *Git*/*GitLab* repository:

(1) the entire 'CHINESE_GOLD_2006' directory was put under version control using several *Git* and LFS commands;
(2) this initial repository created on a local computer was pushed to a central repository on *GitLab*'s servers;
(3) this initial version of the repository on *GitLab* was then cloned by others on their own computers.

Several observations should be noted here. The team used *GitLab* as a centralised server in order to synchronise our versions of the *Git* repository.

28 See 'Git Large File Storage', https://git-lfs.github.com/ (accessed 24 April 2018).

At first sight, this might seem to imply that *Git* and *MediaWiki* have the same centralised storage model. This is not the case as we could have synchronised our repository in an ad-hoc fashion using only *Git*'s push/pull functionality between our respective machines. We decided to use *GitLab*'s centralised services for convenience during the first 2017 workshop. Moreover, as explained above, every cloned copy of the initial version 'v0' could be ultimately shared on other hosting platforms or even exchanged using external storage devices, such as usb drives, while still being able to function with our respective versions, since every cloned *Git* repository contains locally the entire version history of the repository, regardless of the hosting service one might use.

In terms of interface for the viewing content online, it was clear when we pushed the initial repository to the *GitLab* platform that it would not allow direct viewing of all audio/visual documents of 'CHINESE_GOLD_2006'. Unlike *MediaWiki*, *GitLab* restricts the viewing of large images and videos to a maximum of 50MB at the time of writing and does not provide functionality for viewing file types that are not supported by HTML standards (such as TIFF or iMovie projects). Instead, *GitLab* offers the downloading of such files for viewing locally. This is typical of *Git* hosting platforms as they are designed for viewing text-based files—usually source code—and not to interpret a file's content such as instruct a browser to display the content of a *.jpeg.

2 Metadata and provenance

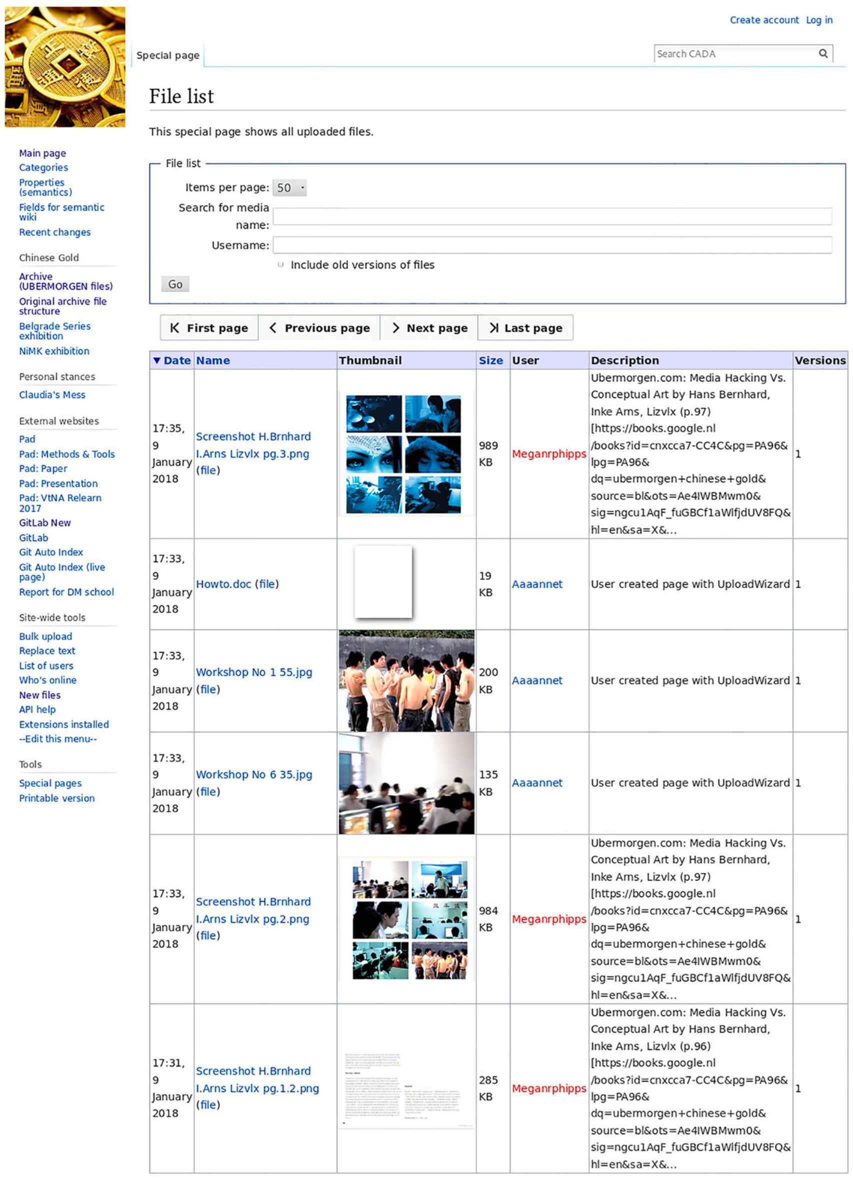
2a MediaWiki

Aside from URLs of raw files, at the time of the upload, *MediaWiki* automatically creates a dedicated page for each file under another assigned URL. This page features file preview, text description, file history, a list of pages linking to the file, EXIF data (for image and audio files), and a subpage for discussion. To distinguish file pages from content pages, the system maintains a dedicated namespace, 'File'—it can be renamed in the configuration file—under which they are accessed.

The time stamp *MediaWiki* produces for files represents the time when a file was uploaded to the server. However, digital archiving requires the preservation of the creation and modification date. In addition, the file path of individual files is lost with the upload as well, as although *MediaWiki* does provide a general list of all uploaded files, its essential structure is flat (i.e. non-hierarchised) so that archived files are treated as if on the same level (Fig. 4). In other words, *MediaWiki* provides no straightforward way to maintain the folder structure of the materials to be archived. Relations between files—and pages—are dependent on the placement of information and links within their description text. The team considered preserving the original file paths by entering them into the description for each file, however this would be highly time consuming. A more optimal, albeit still not ideal, workaround was to include paths within file names ahead of their upload, for which we wrote a Python script.[29]

29 The script is available at: https://multiplace.org/cada/index.php/Filenamingsystem (accessed 3 March 2019).

If a user wishes to browse through a folder structure of an uploaded artist's archive on *MediaWiki*, with or without file previews, it has to be reproduced manually. A more convenient option is to employ the extension 'Semantic *MediaWiki*' that allows for greater categorisation and automatic querying of data. But even in this case, the relevant metadata need to be pre-entered on the respective pages of the files. 'Semantic *MediaWiki*' presents something of a double-edged sword: while the use of its properties and values enables the consolidation of data, this requires a strong organisation and mutual understanding on the part of its users, otherwise the cat-



File list

This special page shows all uploaded files.

File list

Items per page: 50

Search for media name:

Username:

Include old versions of files

▼ Date	Name	Thumbnail	Size	User	Description	Versions
17:35, 9 January 2018	Screenshot H.Brnhard I.Arns Lizvlx pg.3.png (file)		989 KB	Meganrphipps	Ubermorgen.com: Media Hacking Vs. Conceptual Art by Hans Bernhard, Inke Arns, Lizvlx (p.97) [https://books.google.nl /books?id=cnxcca7-CC4C&pg=PA96& lpg=PA96& dq=ubermorgen+chinese+gold& source=bl&ots=Ae4IWBMwm0& sig=ngcu1AqF_fuGBCf1aWlfjdUV8FQ& hl=en&sa=X&...	1
17:33, 9 January 2018	Howto.doc (file)		19 KB	Aaaannet	User created page with UploadWizard	1
17:33, 9 January 2018	Workshop No 1 55.jpg (file)		200 KB	Aaaannet	User created page with UploadWizard	1
17:33, 9 January 2018	Workshop No 6 35.jpg (file)		135 KB	Aaaannet	User created page with UploadWizard	1
17:33, 9 January 2018	Screenshot H.Brnhard I.Arns Lizvlx pg.2.png (file)		984 KB	Meganrphipps	Ubermorgen.com: Media Hacking Vs. Conceptual Art by Hans Bernhard, Inke Arns, Lizvlx (p.97) [https://books.google.nl /books?id=cnxcca7-CC4C&pg=PA96& lpg=PA96& dq=ubermorgen+chinese+gold& source=bl&ots=Ae4IWBMwm0& sig=ngcu1AqF_fuGBCf1aWlfjdUV8FQ& hl=en&sa=X&...	1
17:31, 9 January 2018	Screenshot H.Brnhard I.Arns Lizvlx pg.1.2.png (file)		285 KB	Meganrphipps	Ubermorgen.com: Media Hacking Vs. Conceptual Art by Hans Bernhard, Inke Arns, Lizvlx (p.96) [https://books.google.nl /books?id=cnxcca7-CC4C&pg=PA96& lpg=PA96& dq=ubermorgen+chinese+gold& source=bl=Ae4IWBMwm0& sig=ngcu1AqF_fuGBCf1aWlfjdUV8FQ& hl=en&sa=X&...	1

Fig. 4 Artists' file archive on *MediaWiki*.

egorisation of data could become too chaotic and too meaningless for the user community.[30]

MediaWiki automatically displays the EXIF metadata as created by cameras and recorders of images and audio recordings. It also displays the name of the user who uploaded a file and it has a section on the licensing of an uploaded file.[31]

2b Git

Since *Git* archived the entirety of 'CHINESE_GOLD_2006', all file paths were retained during the creation of the repository, its subsequent uploading to *GitLab* and cloning to any peer computers. On *GitLab*, one can browse the archive in a similar manner as one would on a computer, the file path of the files and directories being the same. The URLs produced by *GitLab* are permanent and follow the path structure of the archive as well. In fact, like *MediaWiki*, *GitLab* produces two URL's for each file, one

30 Engel and Wharton came to similar conclusions after testing different open-source software platforms to develop information resources about artists in their project, *The Artist Archives Initiatives*. Engel and Wharton, 'Managing Contemporary Art Documentation'.

31 Metadata handling depends on the upload tool used and one of the difficulties heritage institutions encounter in using *MediaWiki* is the mapping of the metadata of the institution to the metadata of the upload tool. See Jonathan Morgan, et al., 'Research: Supporting Commons contribution by GLAM institutions', Wikimedia Meta-Wiki, 10 of which is the link for viewing the file reference through *GitLab*'s web interface and the other the address of the raw file on *GitLab*'s servers.

The *GitLab* interface shows metadata that was produced by *Git* rather than metadata about the files themselves. That is, rather than displaying each file's creation and modification dates, it shows when they were added to the *Git* repository (relative to the time of viewing) and as part of which commit. As explained earlier, when changes are made to a repository, these need to be committed to the repository in order to take effect and for *Git* to version the archive. Each commit metadata consists of the username of the user who commits changes, a commit message and an identification cypher for the commit itself. This is mainly what *GitLab* shows on its web interface.

December 2018, https://meta.wikimedia.org/wiki/Research:Supporting_Commons_contribution_by_GLAM_institutions (accessed 23 January 2019)

However, this information alone was not enough for the purpose of creating a digital archive. The team needed to extract as much metadata as possible for each file so as to display information about the digital object itself rather than just its provenance.

In order to do this *Git* can be configured to extract files' EXIF metadata using the program 'ExifTool'.[32] For each file containing one of the listed extensions, *Git* produces an additional text-based metadata file containing the EXIF information of the file. This metadata file is then added to the repository to be tracked by *Git* so that when a given file is changed and committed to the repository so does its accompanying EXIF metadata file.

32 'Exiftool' (version 11.10), 2018, https://www.sno.phy.queensu.ca/~phil/exiftool/ (accessed 23 September 2018). See also, Chacon and Straub, *Pro Git*, 347.

While this built-in approach of producing metadata seemed promising, it did not offer us the means to explore the raw, unsorted archive in a way we deemed optimal for our purpose. As explained, the team wanted to produce an index of all files in the archive along with their respective metadata in a presentable format that can be viewed via a web browser. We thus decided to write our own script (in Python) to excavate such information. The approach taken was to write the script so that it recursively crawls every directory and subdirectory of the archive, listing all the files they contain and producing a record of each file's metadata in an HTML format. We used the 'Hachoir' library to parse and read the files' metadata such that we were not only able to read standard metadata types (MIME, EXIF, etc.) but also proprietary ones (Microsoft, Apple, Adobe, etc.).[33] By crawling the archive with our script, we amassed a wealth of meta information on the dates, provenance, types and formats of each file; a type of forensic reading that enabled us to trace the editing lineage of artefacts involved in the production of the various *Chinese Gold* exhibits (Fig. 5).

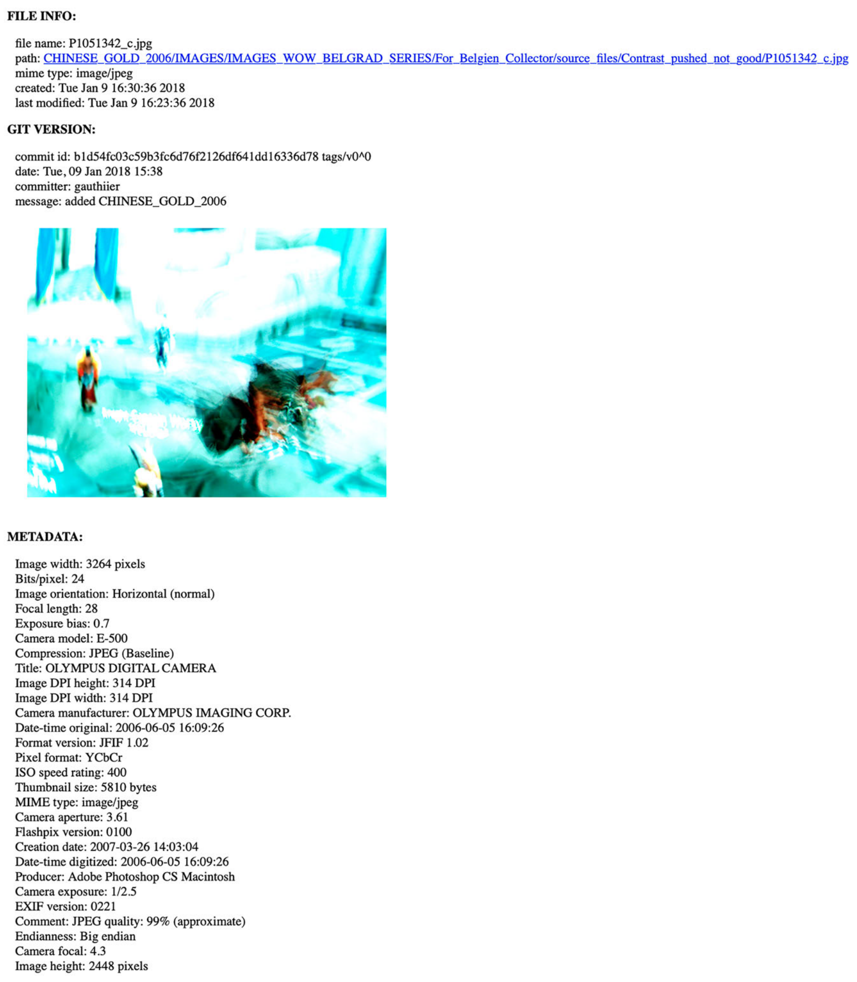
33 'Hachoir' (version 3.0a2), https://hachoir.readthedocs.io/en/latest/ (accessed 5 March 2019).

For example, the team rediscovered the original *World of Warcraft* snapshots/screenshots for the 'Belgrade Series'—taken in mid-2006 with an Olympus camera and further edited with Adobe Photoshop CS on an Apple Macintosh and transcoded from an original jpeg format to tiff. While we could have deciphered *Chinese Gold*'s digital artwork editing lineage directly from UBERMORGEN's 'CHINESE_GOLD_2006' organisation of the files—the directory names in the original archive being evocative enough to understand—these experimental forensic readings enabled us formulate a promising archive-agnostic method of extracting valuable archival information from 'unsorted' sets of digital files, meta information that would have otherwise remained mute in *Git*/*GitLab*.

3 Context, presentation, curation

3a MediaWiki

To explore the presentation possibilities of *MediaWiki* the team set out to compile the data and documentation of two distinct iterations of *Chinese Gold*—'Belgrade Series' (2007), a series of eight images exhibited as a slideshow, and *Chinese Gold* (2010), a single-channel video and two images



```
FILE INFO:

  file name: P1051342_c.jpg
  path: CHINESE_GOLD_2006/IMAGES/IMAGES_WOW_BELGRAD_SERIES/For_Belgien_Collector/source_files/Contrast_pushed_not_good/P1051342_c.jpg
  mime type: image/jpeg
  created: Tue Jan 9 16:30:36 2018
  last modified: Tue Jan 9 16:23:36 2018

GIT VERSION:

  commit id: b1d54fc03c59b3fc6d76f2126df641dd16336d78 tags/v0^0
  date: Tue, 09 Jan 2018 15:38
  committer: gauthiier
  message: added CHINESE_GOLD_2006

METADATA:

  Image width: 3264 pixels
  Bits/pixel: 24
  Image orientation: Horizontal (normal)
  Focal length: 28
  Exposure bias: 0.7
  Camera model: E-500
  Compression: JPEG (Baseline)
  Title: OLYMPUS DIGITAL CAMERA
  Image DPI height: 314 DPI
  Image DPI width: 314 DPI
  Camera manufacturer: OLYMPUS IMAGING CORP.
  Date-time original: 2006-06-05 16:09:26
  Format version: JFIF 1.02
  Pixel format: YCbCr
  ISO speed rating: 400
  Thumbnail size: 5810 bytes
  MIME type: image/jpeg
  Camera aperture: 3.61
  Flashpix version: 0100
  Creation date: 2007-03-26 14:03:04
  Date-time digitized: 2006-06-05 16:09:26
  Producer: Adobe Photoshop CS Macintosh
  Camera exposure: 1/2.5
  EXIF version: 0221
  Comment: JPEG quality: 99% (approximate)
  Endianness: Big endian
  Camera focal: 4.3
  Image height: 2448 pixels
```

Fig. 5 Forensic analysis of the files in *Git* repository. Courtesy of David Gauthier.

exhibited as an installation. Each iteration was given a content page that followed a distinct presentation method.

With the directory 'Belgrade Series' the team attempted to make a straightforward representation of its files, representing its sub-folder structure as different sections of the page. This meant structuring the web page so as to reflect the different versions of the same series of eight images: the listing of TIFF files used in the exhibition at the REX space in Belgrade; the listing of JPEG master files; and the presentation of the images by the artists on their public website (as of 2018).[34]

34 See https://multiplace.org/cada/index.php/Belgrade_Series and https://multiplace.org/cada/index.php/NiMK_exhibition (accessed 17 December 2018).

The team then used the data from the exhibition of *Chinese Gold* at NIMk, Amsterdam in 2010. We emphasised the collation of a wider range of information regarding the exhibition from a multiplicity of sources absent from the artists' archive. Rather than listing the artwork's media components on the page by hand, we used Semantic *MediaWiki* to link relevant media files to the exhibition page.

In this way, semantic relations can be created between wiki pages, whether they represent exhibitions, components, documents or other files. The relations can be subsequently queried and the results displayed along with other content on the wiki. Besides this, *MediaWiki* allowed the smooth embedding of images and videos within content pages. Metadata and further information about respective files were available on separate pages accessible through a simple click on a media file.

3b Git

Certain contemporary artworks adapt to their context and their curation becomes part of the work itself. With other contemporary works, the collection process is part of the work.[35] *Chinese Gold* is a bit of both. As a system, *Git* provides the basic infrastructure to collect and organise a project such as *Chinese Gold* and its curatorial context without imposing a rigid classification system. By facilitating the artwork's analysis and documentation, *Git* can, albeit indirectly, support the exhibition of the artwork. The two 'exhibit' branches of our *GitLab* repository were created with this in mind and put forth an exhibition-focussed order of the archive that we could collectively organise and assess in a decentralised manner.

35 See, for example, Martha Buskirk, *The Contingent Object of Contemporary Art* (Cambridge, MA: MIT Press, 2003), 162.

As the team were working on these branches, and in tandem with the work that was being done on *MediaWiki*, it became clear that a descriptive overview of *all* the files present in the archive would speed up the means by which we were able to locate and decide upon which documents to consider for a potential exhibition and which ones to discard. In our case, the *Chinese Gold* files were legibly sorted into directories labelled according to the various projects' exhibits, so we were able to manually locate the artwork's source material for each exhibit without much difficulty. It is important to understand that this may not always be the case. As we are interested in working with entire 'unordered' working copies of digital art archives, which may include source material of artworks—source code, Photoshop projects and so forth—as well as preliminary research materials, spreadsheets of budgets, various edited versions of artworks, etc., it may be necessary to devise algorithmic means to analyse the content of archives automatically. This could aid the descriptions of the types, assets and provenance of their constituting entities.

To see how this could work we developed the 'experimental' branch where the Python script described earlier was placed. This script recursively crawls every subdirectory of the root directory ('CHINESE_GOLD_2006') and produces a single html file where all the files of the archive are listed along their respective paths, their metadata and their content (if the browser allows its display). This html index gives an overview of the content of the repository as a flat list of entities with metadata descriptors. It thus makes it easy to glance at the repository's content to identify elements that might be of interest. As stated above, this scripted automatic reading of the archive is a promising avenue for the research of digital archive-agnostic methods to extract archival information from 'unsorted' sets of 'unknown' digital files.

By interpreting the evolution of the artwork and by understanding other stakeholders' contributions, as well as the documentation of previous exhibitions, curators can make well informed decisions about the next exhibition they are working on. The Guggenheim Museum's 'Iteration Report' for installation artworks fulfils a similar, but narrower, purpose: it documents the artwork's installation at each exhibition and adds to the documentation of the exhibition history.[36] Yet with digital artworks and *Git*, the artwork itself and its documentation can co-exist as part of the same multi-faceted archive, whose multiple variations can address the needs of curators, archivists and artists themselves.[37]

36 Guggenheim Museum, 'Conservation Department. Iteration Report', 2012, https://www.guggenheim.org/wp-content/uploads/2015/11/guggenheim-conservation-iteration-report-2012.pdf (accessed 13 December 2018).

4 Collaboration, usability

4a MediaWiki

A *MediaWiki* is made to be adjusted and populated entirely by users. Editors create menu structure, draft templates for articles and outline semantic relationships. The number of users is not restricted by design. To maintain the platform, dedicated editorial work is needed, albeit that compared to

37 In terms of curation of digital artworks, *Git*'s own ability to be programmed and scripted as a system to support various levels of automation cannot only be used in order to add context and functionality to the reposi-

tory but could also be a useful feature of algorithmic curation as such. These are avenues that were not directly explored as part of our current experiments but are nonetheless interesting trajectories the research points to.

38 See https://www.mediawiki.org/wiki/Help:Patrolling (accessed 23 September 2018).

39 See https://www.mediawiki.org/wiki/Help:Tracking_changes (accessed 23 September 2018).

other similar systems it can be more evenly distributed. Individual editors can self-appoint themselves to oversee creation of a new artwork entry and patrol respective changes.[38]

A built-in multi-layered system for tracking changes further aids collaboration. *MediaWiki* offers an overview of recent changes on the platform in general and a list of newly created pages.[39] The revision history of each page provides an overview of who has contributed, when and why. Visual diffs system allows for visual comparison between various versions of a page.

MediaWiki is web-based, made for remote access. The ease of access through a web browser means that users do not need any special software installed. People can use and contribute to a platform using any devices with internet connection, in offices, galleries and elsewhere. It is worth noting though that upgrading the system is moderately challenging in comparison to other systems of its scale.

4b Git

As mentioned earlier, the initial version of the archive was pushed to *GitLab* with the version 'v0' tag designating the unmodified state as provided to us by UBERMORGEN. After this version was cloned on various computers, the team created several branches in which the content of the repository and its file structure were modified in different ways. Operationally speaking, 'branching' denotes the creation of a 'sandbox' copy of a given parent branch of the repository, where changes can be applied to the new branch without propagating to the parent branch it initially came from. Typically, a *Git* repository contains many branches from which collaborators work. Changes to branches can be made in parallel without interfering with one another and branches can be merged together at any stage (if no insurmountable conflict between them arises). In *Git*, branches work as a sort of genealogical lineage binding all the actual—and potential—strands of a given archive; they produce a tree-like graph where each trajectory denotes a state of the archive. In our case, we produced three branches, namely two that were used to re-order the files according to *Chinese Gold*'s various exhibits (in a similar manner to *MediaWiki*'s ordering mentioned above), and another experimental branch where the Python scripts we devised were added to the archive to execute a custom digital forensics of the *Chinese Gold* files themselves. It is important to note that branching does not occur online using *GitLab* but locally using *Git* on a cloned version of the initial archive.

A preliminary conclusion

Common museum management systems do not handle complex digital artworks well. By using *Git* and *MediaWiki* as collaborative archival and curatorial systems for artworks instead of their originally intended support for software development and documentation, we explored these tools from a different point of view. The following aspects were considered for studying the systems: file and storage management, metadata and provenance, context, presentation, curation, collaboration and usability.

Both systems are capable of supporting the archiving and access to digital artwork, albeit with limitations. File upload is not straightforward for certain file types and for large files in the case of *MediaWiki*. Both *MediaWiki* and *Git* produce unique web addresses for each file but previews of certain files such as TIFF images, videos and large files are not supported by *Git*, while on *MediaWiki* they have to be manually configured. With *Git*, content remains available offline as the repository is usually cloned to one's own personal computer. *MediaWiki* is a client–server system and it is painstaking to replicate the content of any instance.

Since neither *MediaWiki* nor *Git* have been developed as archival systems, they lack some of the provisions for metadata and provenance. Unlike *MediaWiki*, *Git* preserves file paths of uploaded files, making it convenient to navigate file structures, but neither platform has the means to preserve provenance information such as the original file creation dates. While *Git* allows a rearrangement of files without having to change the original directory structure, *MediaWiki* lacks features for operations on file directories.

The systems are capable of indirectly supporting the exhibition of a complex digital artwork. The text-oriented *MediaWiki* not only allows for additional descriptions but also for content presentation and linking that can be particularly useful to understand the background and context. It offers a rich framework for creating composite pages that may bring together artwork documentation and source files in an unambiguous way. The file-oriented *Git*, on the other hand, allows one to branch the artwork and create new folders, so that the artwork components and its documentation can co-exist as part of the same multi-faceted archive, and whose multiple variations can address the needs of curators, archivists and artists themselves. A *Git* repository can even serve as the base for algorithmic curation.

From the outset, both *MediaWiki* and *Git* have been developed as systems for collaboration facilitated by the internet. However, their successful adoption requires resolution and patience. As shown above, the use of *MediaWiki*, with its rich syntax and many extensions, may be as demanding as the use of *Git* that needs a good faculty of abstraction to understand the collaboration processes.

Since archiving and documenting often overlap, a system which would build upon both platforms would be an ideal version control system (VCS) for the collaborative archiving of complex digital artworks. Such a system should take the following into account:

1) *File and storage management*. The VCS must be capable of handling all files irrespective of their format and size. It should offer preview of files including images and videos. It is important that the system is easily portable by way of cloning. Cloning follows the idea of LOCKSS (Lots Of Copies Keep Stuff Safe) and also prompts users to experiment safely with sandboxed copies of archives. If technically feasible, the VCS might have a visual diff in place to allow comparing changes in the content of images or even multimedia files.
2) *Metadata and provenance*. The VCS must preserve the creation date, metadata and file path for each file at the moment of its ingest and afterwards. It should support operations on file structures such as replication and display. It has to record changing versions of a file and allow linking it to other elements.
3) *Context, presentation, curation*. The system should feature a framework for bringing source files and documentation together to present the background and context of the artwork.
4) *Collaboration, usability*. Rather than serving a general purpose, the design of the VCS must take into account the expertise of stakeholders in the process of archiving art and adapt its interface and functionality accordingly.

Of course, many questions remain that call for further exploration and research. How to make *MediaWiki* and *Git/GitLab* work together? Would it make sense to limit the use of *Git* to being a file repository and *MediaWiki* a documentation tool? Furthermore, the current work around *WikiBase* and *WikiData* might offer new insight and usage of version control that could be particularly beneficial for sustainable archiving of complex digital artworks.[40]

With regard to our case study, considering that a sizeable portion of the content used for *Chinese Gold* has been appropriated by UBERMORGEN from other sources, a collaborative and granular continuation of the project supported by a VCS would allow the files to remain in the same media-scattered form in which the artists found them online or offline. Archiving such artworks by way of a granular, open-source versioning system suggests a continuation of UBERMORGEN's artistic philosophy and intent well beyond their lifespan: a philosophy respecting the democratisation of cultural knowledge, challenging existing intellectual property regimes, with the prioritisation of context over concept, and eliminating the need for an established or finalised conceptualisation of an artwork.

40 Jonathan Morgan and Sandra Fauconnier, 'What Galleries, Libraries, Archives, and Museums can Teach Us about Multimedia Metadata on Wikimedia Commons', *Wikimedia Blog*, 29 January 2018, https://blog.wikimedia.org/2018/01/29/glam-multimedia-metadata-commons/ (accessed 23 September 2018).

ORCID

Dušan Barok http://orcid.org/0000-0002-0691-0748
Julie Boschat Thorez http://orcid.org/0000-0002-8979-987X
Claudia Roeck http://orcid.org/0000-0003-2612-5672

Acknowledgements
Dušan Barok's and Claudia Roeck's participation in the research was realised in the framework of the Marie Skłodowska-Curie Innovative Training Network 'New Approaches in the Conservation of Contemporary Art' (NACCA), funded by the European Union H2020 Programme (H2020-MSCA-ITN-2014) under Grant Agreement n° 642892. The authors would like to thank UBERMORGEN for generously providing us with their archive as well as Jim Wraith, Judith Hartstein, Larissa Tijsterman and Megan Phipps who all participated in the 'Collaborative Archiving of Digital Art' workshop in 2018—we thank them for their valuable input and contribution to the thinking about the issues and conclusions outlined in this article. We would also like to thank Lozana Rossenova who read and commented on earlier versions of the article.

Funding
Dušan Barok's and Claudia Roeck's participation in the research was realised in the framework of the Marie Skłodowska-Curie Innovative Training Network 'New Approaches in the Conservation of Contemporary Art' (NACCA), funded by the European Union H2020 Programme (H2020-MSCA-ITN-2014) under Grant Agreement n° 642892 (H2020 Marie Skłodowska-Curie Actions).

Résumé
«Archivage des œuvres numériques complexes»
La transmission de la documentation des modifications apportées à chaque présentation d'une œuvre d'art et des motivations pour chaque exposition est importante pour la préservation à long terme, pour la nouvelle exposition et la compréhension future de l'œuvre. Cependant, il est généralement reconnu que l'archivage numérique existants et les systèmes de documentation utilisés par de nombreux musées ne conviennent pas pour des œuvres numériques complexes. En recherchant une approche qui peut être facilement adaptée, partagée et adoptée par d'autres, cet article se concentre sur les alternatives en open source qui permettent également un travail collaboratif pour faciliter les échanges et la modification des informations. En tant qu'équipe interdisciplinaire de restaurateurs, chercheurs, artistes et programmeurs, les auteurs ont entrepris d'explorer et de comparer les fonctionnalités de deux systèmes dotés du contrôle de version: *MediaWiki* et *Git*. Nous réfléchissons sur leurs éléments techniques, leurs qualités et leurs inconvénients pour archiver des œuvres numériques complexes, tout en examinant le potentiel qu'elles offrent pour les flux de travail collaboratifs.

Zusammenfassung
„Die Archivierung komplexer digitaler Kunst"
Die Übermittlung der Dokumentation der Veränderungen, die bei jeder Präsentation eines Kunstwerkes durchgeführt werden und die Motivation hinter jeder Vorführung sind wichtig für die Erhaltung, erneute Präsentation und das zukünftige Verständnis des Werkes. Allerdings ist es generell so, dass die existierende digitale Archivierung und die Dokumentationssysteme, die von vielen Museen eingesetzt werden, nicht für komplexe digitale Kunst nutzbar sind. Auf der Suche nach einem leicht änderbaren, gemeinsam nutzbaren und von anderen übernehmbaren Konzept, konzentriert sich dieser Artikel auf open-source Alternativen, die auch kollaboratives Arbeiten mit dem Teilen und Ändern von Information ermöglichen. Als ein interdisziplinäres Team von Restauratoren, Forschern, Künstlern und Programmierern wollen die Autoren die Funktionalität zweier Systeme mit Versionskontrolle untersuchen und vergleichen, *MediaWiki* und *Git*. Die technischen Details, Vor- und Nachteile bei der Archivierung komplexer digitaler Kunst, sowie das Potential für kollaborative Workflows werden untersucht und reflektiert.

Resumen
"Archivando obras de arte digital complejas"
La transmisión de documentación de los cambios en cada exposición de una obra de arte y la motivación detrás de cada muestra son importantes para la preservación continua de la obra, su re exposición y su comprensión futura. Sin embargo, en general, se reconoce que los sistemas existentes de archivo y documentación digital utilizados por muchos museos no son adecuados para obras de arte digital complejas. Buscando un enfoque que otros puedan ajustar, compartir y adoptar fácilmente, este artículo se centra en alternativas de código abierto que también permitan el trabajo colaborativo y que faciliten el intercambio y la divulgación de información. Los autores, un equipo interdisciplinario de conservadores, investigadores, artistas y programadores, se dispusieron a explorar y comparar las características de dos sistemas de control de versiones: MediaWiki y Git. Reflexionamos sobre sus detalles técnicos, ventajas e inconvenientes para archivar obras de arte digitales complejas, mientras observamos el potencial que ofrecen para los procesos de trabajos colaborativos.

摘要
"复杂的数字艺术品的存档"
对于艺术品的持久保存、再展览和后续理解十分重要的是其每次展示时变更部分的记录存档和展示动机。然而，人们普遍认为许多博物馆现有的数字存档和文件系统并不适合存储复杂的数字艺术品。本文以能满足多方信息共享与变更的替代方案为重点，旨在寻找一种易于调整、共享和采纳的方法。本文作者是由保存修复人员、研究人员、艺术家和程序员组成的跨学科团队，他们打算探索和比较两个以版本控制为特色的系统（MediaWiki 和 Git）的功能。他们将反思两个系统在存档复杂的数字艺术品时的技术细节、优点和缺点，同时着眼于它们在协同工作流上带来的潜力。

Biographies

Dušan Barok is a PhD Fellow of the Marie Skłodowska-Curie Innovative Training Network 'New Approaches in the Conservation of Contemporary Art' (NACCA) at the University of Amsterdam. He graduated in Information Technologies from the University of Economics, Bratislava, and Networked Media Design from the Piet Zwart Institute, Rotterdam.

Julie Boschat Thorez is an artist and researcher with an interest in information structuring and the influence of digital networks on human agency. She was trained in Fine Arts at the ERG in Brussels and Media Design at the Piet Zwart Institute in Rotterdam.

Annet Dekker is Assistant Professor of Media Studies: Archival and Information Studies at the University of Amsterdam and Visiting Professor and co-director of the Centre for the Study of the Networked Image at London South Bank University.

David Gauthier is a PhD Fellow of the Netherlands Institute of Cultural Analysis (NICA) based at the Amsterdam School of Cultural Analysis (ASCA) of the University of Amsterdam. He holds a degree in Media Arts & Sciences from the Massachusetts Institute of Technology and a degree in Mathematics from the Université du Québec à Montréal.

Claudia Roeck is a time-based media conservator, fascinated and confronted with fast technological change and the processual character of many contemporary artworks. Currently, she is researching preservation strategies for software-based artworks as a PhD candidate at the University of Amsterdam for the NACCA.eu project.

Contact address

Dušan Barok
Department of Media Studies/Amsterdam School for Heritage, Memory and Material Culture
University of Amsterdam
The Netherlands
Email: D.Barok@uva.nl, db@monoskop.org

Julie Boschat Thorez
Independent Artist and Researcher
The Netherlands
Email: jboschatthorez@gmail.com

Annet Dekker
Department of Media Studies/Amsterdam School for Cultural Analysis
University of Amsterdam
The Netherlands
Email: annet@aaaan.net

David Gauthier
Department of Media Studies/Amsterdam School for Cultural Analysis
University of Amsterdam
The Netherlands
Email: gauthier@uva.nl

Claudia Roeck
Department of Media Studies/Amsterdam School for Heritage, Memory and Material Culture
University of Amsterdam
The Netherlands
Email: claudia.roeck@crockodile.ch