INSTITUTIONEN FÖR KULTURVÅRD

SOFTWARE CONSERVATION IN RHIZOME'S EAAS INTERFACE

A Conservator's Take on its Terminology and Aptitude

Erik Mattias Nilsson

Uppsats för avläggande av filosofie kandidatexamen med huvudområdet konservering med inriktning mot kulturvårdsobjekt

År 2026, 15 hp

Grundnivå

MJUKVARUKONSERVERING I RHIZOMES EAAS-GRÄNSSNITT

En konservators betänkande kring dess terminologi och duglighet

Erik Mattias Nilsson

Handledare: Omid Oudbashi

Examensarbete, 15 hp

Konservatorsprogrammet, 180 hp

GÖTEBORGS UNIVERSITET ISSN 1101-3303

Institutionen för kulturvård

UNIVERSITY OF GOTHENBURG
Department of Conservation
P.O. Box 130
SE-405 30 Gothenburg, Sweden

https://www.gu.se/kulturvard
Fax +46 31 7864703
Tel +46 31 786 00 00

Degree of Bachelor of Science with a major in Conservation with Specialization in Cultural Heritage Objects
Graduating thesis, BA/Sc, 2026

By: Erik Mattias Nilsson
Mentor: Omid Oudbashi

Software Conservation in Rhizome's EaaS Interface: A Conservator's Take on its Terminology and Aptitude

ABSTRACT

Software creations depend on computing environments that become obsolete. Emulation can recreate those environments, but the tools built for this work speak the language of computer science, not conservation. This leaves little room for the conservator's reasoning — the judgments behind a preservation decision, and not just its technical settings. This thesis examines the Demo-UI, the interface of the Emulation-as-a-Service (EaaS) framework as deployed at Rhizome, asking how its vocabulary relates to conservation practice and where a conservator can document preservation decisions within it.

The study combines a systematic terminology inventory of the Demo-UI — documenting every field, button, and setting visible in the interface — with a walkthrough of three case studies preserved at Rhizome: *Tod dem Fernsehen* (R.A.M.S., 1993–1994), *Bomb Iraq* (Cory Arcangel, 2005), and *Every Icon* (John F. Simon Jr., 1997). A comparative study of Rhizome's ArtBase catalog and the Stabilize interface shows how the same concepts are described differently in related systems. The study produces two contributions: first, it identifies four existing free-text fields in the Demo-UI where conservation information — rationale, condition, risk, authorship — can be entered today without changes to the software; second, it identifies six interface elements that need clarification through expanded abbreviations, tooltips, or bug fixes. The discussion draws on Deleuze's concepts of the rhizome and difference in repetition, Apprich's media genealogy of net cultures, and the conservation values tradition of Ruskin, Riegl, and Viollet-le-Duc. The thesis argues that the Demo-UI already has places where conservation documentation can go, and that making the interface's own vocabulary more transparent is a necessary step toward using it as a conservation tool — one in which the conservator can record not only how an environment is configured, but why.

Title in original language: Software Conservation in Rhizome's EaaS Interface: A Conservator's Take on its Terminology and Aptitude
Language of text: English
Number of pages: 60
Keywords: digital preservation, software-based art, emulation-as-a-service, conservation terminology, born-digital cultural heritage, EaaS Demo-UI, Rhizome ArtBase, media genealogy

ISSN 1101-3303

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Big thanks to Omid Oudbashi for supervising this thesis, and to Helena Berg for her valuable input. I am grateful to Nika Maltar, Rafael de Vries, Almut Schilling, and Dragan Espenschied, whose generosity and insight were essential to this work. Finally, my thanks to Viviane Kübler and Martina Griesser-Stermscheg at die Angewandte, who first brought me to Vienna. Any errors that remain are my own.

LIST OF ABBREVIATIONS

Abbreviation	Full Term
API	Application Programming Interface
bwFLA	Baden-Württemberg Functional Long-Term Archiving and Access
CD-ROM	Compact Disc Read-Only Memory
CIDR	Classless Inter-Domain Routing (network address notation)
CiTAR	Citing and Archiving Research
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
EaaS	Emulation-as-a-Service
EaaSI	Emulation-as-a-Service Infrastructure
EMiL	Emulation of Multimedia Objects in Libraries
EU	European Union
GRATE	Global Remote Access to Emulation services
ICOM-CC	International Council of Museums — Committee for Conservation
IMLS	Institute of Museum and Library Services
METS	Metadata Encoding and Transmission Standard
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OAIS	Open Archival Information System (ISO 14721)
OPF	Open Preservation Foundation
OS	Operating System
PNG	Portable Network Graphics (image format)
PREMIS	Preservation Metadata: Implementation Strategies
PRONOM	Public Record Office + Nôm (Brown, 2019)
PUID	PRONOM Unique Identifier (file format identification)
SPN	Software Preservation Network
TMW	Technisches Museum Wien (Vienna Technical Museum)
UUID	Universally Unique Identifier
UVI	Universal Virtual Interactor (Cochrane et al., 2019)
VHS	Video Home System
VRML	Virtual Reality Modeling Language
QEMU	Quick Emulator

1 Introduction

1.1 Context

The digital transformation is reshaping every profession — medicine, law, engineering, education. Conservation is no exception. As cultural production increasingly takes digital form, the conservator's field of responsibility extends into preservation metadata, digital storage media, and software, where both media and software are inherently vulnerable to obsolescence. Museums and archives already hold collections of born-digital oeuvres, and those collections are growing. The professionals who will care for this material need to understand both the cultural heritage values that guide conservation and the technical infrastructure that makes digital preservation possible.

Several strategies exist for preserving digital material. Migration converts files to newer formats, but risks altering the experience of the work. Hardware preservation maintains original machines, but depends on components that fail and cannot be replaced indefinitely (Rechert et al., 2016, p. 19). Documentation records the work without keeping it running. Web archiving captures online content as snapshots. For software objects that depend on specific computing environments — a particular operating system, a particular browser, a particular runtime — emulation has emerged as a central approach: rather than preserving the original machine, it recreates the machine's behavior in software. It is this strategy, and the tools built around it, that this thesis examines.

At the same time, the tools built for digital preservation were not built by conservators. Emulation as a preservation strategy was proposed by a computer scientist (Rothenberg, 1999), and the EaaS framework was developed from 2011 at the University of Freiburg (von Suchodoletz, 2012). These tools were designed to make emulation work, not to support conservation practice. The result is a gap: the tools function, but they do not speak the language of conservation. They describe how the technology works, not what a conservator needs to decide — which sensory qualities to prioritize, what to document about the creator's intentions, or how the cultural significance of the work should guide its preservation. The conservator who uses them must translate between the world of cultural heritage values and the world of technical infrastructure — without support from the interface itself.

Emulation-as-a-Service (EaaS) — a term first coined by van der Hoeven in 2009 (von Suchodoletz et al., 2013, p. 133) — recreates obsolete computing environments inside modern web browsers, allowing software objects to be experienced long after their original platforms have disappeared. Rhizome, a digital arts organization founded in 1996, uses EaaS to preserve its ArtBase archive of born-digital artworks (Rossenova, 2021, p. 14). The framework ships with the Demo-UI, a general-purpose administration interface whose vocabulary — environment, object, disk image, software — reflects its computer science origins. The interface shapes how a conservator preserves, configures, and documents the work.

1.2 Problem

The Demo-UI's terminology is oriented toward making the emulated system function. It does not support the conservator in documenting why and how preservation choices

were made — the reasoning behind decisions about examination and treatment that defines conservation as a discipline — even though a conservator turns to the interface precisely in order to understand a work's condition and how it has been conserved. This gap is less a deliberate design decision than a consequence of how the interface evolved through a succession of largely technical research projects (see Section 2.4), rather than from needs articulated by conservators and curators.

The problem is therefore not only one of documentation. The vocabulary a tool makes available shapes what its user attends to and what they treat as a choice worth thinking through. Without conservation terms, the interface affects how conservation is practiced, not merely how it is recorded: the work risks becoming purely technical — what Molander (1993, p. 19) calls “applied” science, in which the practitioner applies configurations without a framework for recording the values, judgments, and knowledge that guided those choices. At present the interface provides no explicit vocabulary linking the technical infrastructure to the conservation discipline, yet both forms of knowledge are needed.

Rossenova (2021) examined the ArtBase data model in depth, but the Demo-UI itself has not yet been systematically analyzed from a conservation perspective in the published literature known to the author. That is the contribution of this thesis.

1.3 Purpose and Aim

The aim of this thesis is to examine how the terminology in the EaaS Demo-UI supports — or falls short of — the conservator's role in preserving software objects. Through a comparative analysis with the ArtBase and Stabilize, and three case studies, the goals are:

1. To examine the conservator's role in preserving software objects, and to identify where the Demo-UI's vocabulary — built from a computer science perspective — falls short for conservation purposes.
2. To map and explain the Demo-UI's terminology from a conservation perspective: what the concepts mean, where they create ambiguity, and what is missing.
3. To show how the existing terminology can be used more deliberately by a conservator, without requiring the interface to be redesigned.

1.4 Research Questions

Main research question:

How does the terminology of the EaaS Demo-UI relate to conservation practice, and how can it be used to document the preservation of software objects more deliberately?

Supporting questions:

1. How do the Demo-UI's concepts relate to conservation practice, and where does its vocabulary fall short of supporting the conservator?
2. How can a conservator use the existing terminology more deliberately to document preservation decisions, without redesigning the interface?
3. What would a conservation-oriented vocabulary for EaaS look like, and what alternatives already exist?

The questions are addressed through a terminology inventory of the Demo-UI, a comparative analysis with the ArtBase and Stabilize, and three case studies — *Tod dem Fernsehen* (R.A.M.S., 1993–1994), *Bomb Iraq* (Cory Arcangel, 2005), and *Every Icon* (John F. Simon Jr., 1997).

1.5 Theoretical Framework

This study draws on three theoretical perspectives, each chosen for a distinct task rather than as part of a single school.

Cather's (2006) trans-technological conservation methodology establishes that conservation principles apply regardless of an object's material or technology. It provides the warrant for treating software, and its emulation, as a legitimate object of conservation, and for carrying the discipline's questions — about condition, value, and treatment — into a digital interface.

Apprich's (2017) media genealogy holds that digital tools carry the vocabulary of the communities that built them. This supplies the analytical lens: it licenses reading the Demo-UI's terminology not as neutral labelling but as a trace of its computer science origins, and so makes the vocabulary itself an object of study.

Deleuze's difference in repetition (Deleuze, 1994) and the discourse around the rhizome he developed with Guattari (Deleuze & Guattari, 1987) reframe what emulation produces: each emulation is a new actualization rather than a copy. This locates the conservation problem precisely — what has to be documented is not a stable original but a field of difference — and shows why a vocabulary oriented toward making a system run struggles to capture it.

Together, these perspectives set the terms for analyzing the Demo-UI's terminology: where conservation can be addressed within the interface and where it must be taken up outside it, why the tool holds so little conservation thinking in the first place — a question its origins answer — and how an emulated object, re-actualized rather than copied, has to be reconciled with the concerns of conservation.

1.6 Delimitations

Several boundaries define the scope of this thesis. It centers on the conservator's role in software preservation and on how one specific interface — the EaaS Demo-UI as deployed at Rhizome — supports that role through its vocabulary and functionality. Related systems, tools, and questions are treated only insofar as they illuminate that focus, as set out in the delimitations below:

- The study examines the EaaS Demo-UI, with the ArtBase and Stabilize as comparative references. These two systems were chosen because they directly engage with emulation-based preservation of software and share an institutional lineage with the Demo-UI (see Section 3.4.2 for the full rationale). Other preservation approaches lie outside the scope.
- The EaaSI platform, which builds on the EaaS framework for institutional deployment in the United States, is not analyzed. EaaSI is primarily oriented toward library and archival use cases (Rossenova, 2021), and this thesis focuses on the Demo-UI as it is used at Rhizome for software conservation, not on EaaSI's interface or infrastructure.

- The focus is on the vocabulary visible in the user-facing interface — not the underlying databases or APIs. Digital preservation tools such as Siegfried (file format identification using PRONOM signatures) and BagIt (a packaging standard for verifiable transfer of digital content) are part of the broader preservation workflow around the Demo-UI, but their vocabulary and output are not visible in the interface and are therefore not analyzed in detail. They are discussed in Section 5.5 as examples of information that exists in the preservation workflow but is not exposed in the interface.
- The goal is to explain and analyze the existing terminology, not to propose a redesign, though minor clarifications are suggested where the interface is unclear.
- The case studies ground the terminology analysis in practice, not as comprehensive preservation histories.
- The legal challenges of displaying software through emulation — including copyright, licensing, and intellectual property — are considerable and currently being addressed at the European Union level. Artworks tend to be the most straightforward case; video games and commercial software raise additional complexities (A. Schilling, personal communication, April 15, 2026). These legal questions fall outside the scope of this thesis.
- This is a conservation thesis written at Gothenburg University. It does not represent the views of Rhizome, OpenSLX, or the University of Freiburg.

2 Background

2.1 The Digital Shift in Cultural Heritage

Since the 1960s, a growing part of cultural heritage has taken the form of software: programs that run, respond, and change. Such works first emerged in artistic practice, but the same condition now reaches across art and technical collections alike — including the Technical Museum Vienna (TMW), which preserves software it does not classify as art. These works depend on machines to exist, and when those machines disappear, they disappear with them — unless someone intervenes.

Already in the 1960s, the Swedish artist duo Beck & Jung — Holger Bäckström (visual artist) and Bo Ljungberg (mathematician and engineer) — used early 1960s computers at Lund University's data center to generate art from random binary data. Their Bildalfabetet (1966) was a graphic alphabet derived from computer-generated ones and zeros, and their interactive work Felixsnurran (1968) — 35 reversible tiles offering over 34 billion combinations — made every displayed arrangement potentially unique (Reeder, 2024). The conservation questions their work raises — is an interactive, combinatorial artwork the same object each time it is displayed? what is its “original” state? — anticipate the questions a conservator working with emulated software grapples with today.

Making such work no longer requires access to a university data center. Personal computers brought software-based practice into studios and galleries; CD-ROMs made interactive works distributable; and the web, from the mid-1990s, opened net art — a scene in which the browser itself became the medium.

Each new layer of software, however, brought new dependencies — and new ways for works to stop working. The Macintosh operating system that ran in 1994 does not run natively on any computer sold today. The Netscape browser that displayed web art in 1997 no longer runs on modern systems. The Java runtime that powered applets in the late 1990s has been deprecated by its manufacturer. Cultural heritage institutions worldwide — museums, archives, libraries — now hold collections of software that cannot be accessed without specialized intervention.

2.2 Net Cultures and Software Objects

One of this thesis's case studies comes from R.A.M.S., a Viennese media art collective formed by Alf Altendorf, Margarete Jahrmann, and Max Moswitzer (Altendorf, 1994). *Tod dem Fernsehen* (1993–1994) emerged directly from the early net cultures of the 1990s, a scene shaped by an amateur sensibility, improvisational production, and a deliberately rough aesthetic (Apprich, 2017, p. 38). Across Europe and the United States, non-commercial internet providers — Public Netbase in Vienna, Ljudmila in Ljubljana, The Thing in New York — offered both access and a platform for the self-determined use of new media technologies (Apprich, 2017, pp. 35–37), blurring the boundaries between art practice and technical infrastructure. *Bomb Iraq* (Cory Arcangel, 2005) belongs to the same political tradition: a used Macintosh with its previous owner's traces intact became the artwork (Rhizome, 2014). *Every Icon* (John F. Simon Jr., 1997) belongs to a different strand of software objects — a Java applet

that systematically cycles through every possible 32×32 black-and-white grid (Rhizome ArtBase, n.d.-b).

These works cannot be separated from the platforms they ran on. *Tod dem Fernsehen* was built for a Macintosh operating system of the early 1990s; *Every Icon* as a Java applet for Netscape Navigator; *Bomb Iraq* in HyperCard, which Apple discontinued in 2004. These platforms were not neutral infrastructure but technical environments with their own user cultures and aesthetics. Preserving the works for posterity is how a specific moment of net cultures and software objects has the chance to be studied as a historical object or to survive as a living thing.

2.2.1 Defining the Software Object: A Working Definition

This thesis uses the term *software object* rather than “software-based art.” The choice is deliberate: not all institutions that preserve software-dependent works operate within an art context. A technical museum such as the TMW preserves software for its historical and cultural significance without framing it as art. The term *software object* is broader — it refers to any cultural object whose existence depends on executable software and the computing environment that runs it. Throughout this thesis, the three case studies — *Tod dem Fernsehen*, *The Web Stalker*, and *Every Icon* — are referred to as software objects in this sense.

2.3 Emulation as Preservation Strategy

This section introduces emulation as a preservation strategy for software objects, alongside its main alternative — hardware preservation. Emulation recreates obsolete computing environments inside modern systems. Rothenberg (1999, pp. 17–25) proposed emulation as a viable strategy for the long-term preservation of digital objects, arguing that rather than converting old files to new formats (migration) or maintaining original hardware (which eventually fails), a software replica of the original machine could run inside a new one. The work's original files run on this virtual machine as if nothing has changed — though of course, everything has changed underneath. Emulation is not a one-time solution. As von Suchodoletz (2012) notes, emulators themselves are software that must be regularly adapted to current hardware and operating systems — the preservation tool requires its own preservation.

The alternative to emulation is hardware preservation — and one striking example is the Japanese media artist Toshio Iwai, who owns over a hundred computers that he purchased when he first created his Time Stratum series (1985–1990). Iwai is a rare case of an artist who consciously planned for the future conservation of his own work by stockpiling the original machines (Civic Creative Base Tokyo, 2023). When the series was re-exhibited at CCBT Tokyo in 2023 — over twenty-five years after its last showing — the works could be presented in nearly their original state. But this strategy depends on a single artist's foresight and resources. For institutions managing large collections, hardware preservation is not a viable long-term strategy. Emulation takes a different approach: rather than preserving the machines themselves, it recreates their behavior in software. This is the basis of EaaS. What distinguishes Emulation-as-a-Service from running emulators locally is that it delivers emulation through web browsers — no installation, no local configuration, no specialist hardware. An institution can provide access to hundreds of emulated environments from a single server, and a

conservator or researcher can open a 1990s Macintosh in a browser tab without managing the emulator themselves.

This thesis examines one specific EaaS deployment: the one maintained by Rhizome, an independent arts organization based in New York and a partner of the New Museum since 2003. Rhizome runs the ArtBase — a catalog of over 2,300 born-digital artworks (Rhizome ArtBase, n.d.-a) — and preserves a subset of them through EaaS, managed via the Demo-UI. The Demo-UI is a general-purpose administration interface through which environments are configured, run, and documented. It is this interface, and its vocabulary, that the thesis analyzes. The analysis extends from the terminology itself to the role the interface leaves for the conservator, and to how EaaS can be used to document the preservation of software objects more deliberately.

Furthermore, an underlying argument in the thesis is whether an emulated work remains the same work once its original machine is gone and only its behavior is recreated. This is not unique to software: conservators are asked to handle objects whose material must be renewed, replaced, or consumed for the work to remain present, such as living or organic works that must be regrown, or perishable and performative works that are re-enacted rather than physically kept. In these cases authenticity rests less in the original substance than in the work's concept and the experience it produces. The thesis returns to this in Section 2.5 and Section 5.2, where, following Deleuze (1994), each emulation can be understood as a new actualization rather than a copy.

2.4 The History of Emulation-as-a-Service

Understanding the Demo-UI's vocabulary requires knowing where it came from. This section traces the institutional history of EaaS (summarized in Table 1), from its origins in research projects in the mid-2000s. The EaaS framework can run many different emulators (such as QEMU, SheepShaver, DOSBox and Basilisk II) — each replicating a different type of obsolete computer (von Suchodoletz et al., 2013, p. 141). It makes these available through APIs: a way for one piece of software to ask another piece of software to do something. The Demo-UI is the web interface that sends these requests on behalf of the user, so that a conservator can select an emulator, configure an environment, and document changes through a familiar browser window.

Table 1: The Development of Emulation-as-a-Service

Period	Initiative	Actors	Contribution
2005–2010	PLANETS (EU project)	University of Freiburg (von Suchodoletz, Rechert)	Emulation established as a viable preservation strategy within a sixteen-institution EU consortium
2006–present	OpenSLX GmbH	Freiburg spin-off (Rechert, later Stobbe)	Commercial development and maintenance of the EaaS software, including the Demo-UI
2011–2014	bwFLA (state-funded project)	University of Freiburg, Baden-Württemberg	First working EaaS framework: emulation delivered as a web service. OPF Award for Research and Innovation 2014 (Digital Preservation Coalition, 2014)

2013–2014	Yale pilot	Yale University Library (Cochrane)	First American institutional deployment of bwFLA
2014–2016	EMiL (DFG project)	Deutsche Nationalbibliothek, Bayerische Staatsbibliothek, HfG Karlsruhe, University of Freiburg (Rechert)	Took the bwFLA EaaS framework into libraries and museums: an emulation-based access framework for multimedia objects (encyclopedias, educational software, late-1990s digital art), released as open source and piloted at the German National Library
2014–2026	Software Preservation Network	Meyerson, Vowell, Educopia Institute	Professional community, governance structures, IMLS-funded training programs
2018–present	EaaSI	Yale (Cochrane), OpenSLX, SPN, Educopia	National infrastructure program. Mellon/Sloan funding (~$3.5M). Nodes at 10+ institutions

2.4.1 Origins at the University of Freiburg

Dirk von Suchodoletz, a researcher at the Albert-Ludwigs-Universität Freiburg, received his PhD in 2008 on the requirements for emulation as a long-term preservation strategy (von Suchodoletz, 2009). His work argued that emulation could provide authentic long-term access to dynamic digital objects without altering them, and set out the conditions this required — among them formalized “view paths” linking an object to the software and hardware environment needed to render it, and a dedicated software archive to preserve those secondary components (von Suchodoletz, 2009).

In 2005 the Freiburg group joined the German nestor competence network for digital preservation, which led to participation in PLANETS — a major EU integration project running from 2006 to 2010. Within PLANETS, the Freiburg team — von Suchodoletz together with Klaus Rechert — worked in the emulation working group, producing the first sustained multi-institutional research on emulation as a preservation method (Rechert et al., 2010; von Suchodoletz, 2009). Rechert would go on to lead the technical development of the EaaS framework that this thesis examines. One concrete result of this period was GRATE (Global Remote Access to Emulation services), released as open-source software in August 2008: a web service that let a user open an obsolete environment — for example, to read a WordPerfect 5.1 document — inside an ordinary browser, with no emulation software installed locally, uploading the digital object and saving it back afterwards (Welte, 2008, pp. 3, 5, 13). GRATE was an early demonstration of the network-delivered model that bwFLA would later formalize as Emulation-as-a-Service.

2.4.2 bwFLA and OpenSLX

After PLANETS, the team secured state funding from Baden-Württemberg for Baden-Württemberg Functional Long-Term Archiving and Access (bwFLA). This project produced the Emulation-as-a-Service framework: a web-based platform that delivered emulation through web services with browser access via HTML5 already working by

2012 (Rechert et al., 2012). Early adopters included the British Library, the Deutsche Nationalbibliothek, and Rhizome (von Suchodoletz et al., 2013).

The Deutsche Nationalbibliothek's Emulation of Multimedia Objects in Libraries (EMiL) project further developed the framework, contributing to the Demo-UI and adding the ability to use objects — such as CD-ROM images — directly from external archival and storage systems (Liebetraut et al., 2016). The Citing and Archiving Research (CiTAR) project extended EaaS in another direction, adding the ability to import container images (Docker/OCI and Singularity/Apptainer) and to preserve web servers (de Vries et al., 2019; de Vries et al., 2021).

OpenSLX GmbH, founded in 2006 as a spin-off from the University of Freiburg, became the commercial entity responsible for developing and maintaining the EaaS software (OpenSLX, n.d.). The Demo-UI — the administration interface examined in this thesis — was first committed to the codebase in October 2015 (Russler, 2015). Its vocabulary — environment, object, disk image — reflects its origin in computer science and systems engineering at Freiburg.

2.4.3 Yale, SPN, and the Formation of EaaSI

In late 2013, Euan Cochrane started as Digital Preservation Manager at Yale University Library. Yale piloted bwFLA's EaaS framework in early 2014 — one of the first American institutional deployments (Yale University Library, 2014). In parallel, Jessica Meyerson and Zach Vowell[1] were building the Software Preservation Network (SPN), receiving IMLS grants in 2015 and 2017 to establish a professional community for software preservation (Meyerson et al., 2017).

These threads converged in 2018 with the launch of EaaSI, pronounced “easy” — unlike EaaS, which is spoken letter by letter, “E-A-A-S” (R. de Vries, personal communication, 2026). Led by Yale with Cochrane as principal investigator, the program brought together OpenSLX as lead technology developer and SPN as community governance partner. The Andrew W. Mellon Foundation and Alfred P. Sloan Foundation provided approximately $3.5 million in three phases (2018, 2020, 2022) to build a national software preservation infrastructure (Yale University Library, 2020; Mellon Foundation, 2017; 2020; 2022). EaaSI nodes were deployed at over ten partner institutions and expanded internationally through the EaaSI Research Alliance. The Australian AusEaaSI initiative (auseaasi.org) operates in connection with the Play It Again projects for preserving Australian video game history, extending the EaaS ecosystem into the cultural heritage of interactive media. EaaSI has recently adopted the all-caps spelling “EAASI” (EAASI Research Alliance, 2026).

SPN announced a twelve-month sunsetting process in February 2026, with EaaSI software development and governance transitioning to Yale Library (Software Preservation Network, 2026).

---

[1] Jessica Meyerson and Zach Vowell are digital archivists who co-founded the Software Preservation Network (SPN). At the time, Meyerson was a digital archivist at the Briscoe Center for American History, University of Texas at Austin, and Vowell a digital archivist at the Robert E. Kennedy Library, California Polytechnic State University. They proposed the consortial model at the Society of American Archivists meeting in 2014; in 2015 their institutions received an IMLS National Forum Grant to establish the organization (Meyerson et al., 2017).

2.4.4 Why This History Matters

The Demo-UI was developed from 2015 within bwFLA's EaaS framework and subsequently adopted by Rhizome for net art and software preservation, as well as by Yale University Library's EaaSI program (Russler, 2015; Yale University Library, 2014). Projects like EMiL, CiTAR, and AusEaaSI show that the EaaS framework has been extended in multiple directions — multimedia objects, container images, web servers, video games — but the Demo-UI's core vocabulary has remained largely unchanged since 2015. Efforts to bridge the gap between preservation practice and the technical interface have been made — Rhizome's work with the ArtBase and the Open Preservation Foundation, and Rossenova's (2021) doctoral research on the ArtBase data model and user agency in born-digital archives. The Demo-UI's vocabulary, however, remains that of computer science. The ArtBase, Rhizome's born-digital art collection built on Wikibase, describes the same software objects using conservation-oriented categories. Stabilize, a commercial platform built on the same EaaS technology for institutional software preservation, reorganizes the interface around projects and assets rather than environments and objects. These differences are examined in Section 4.2.

2.5 Who Is a Conservator?

There is a Persian expression — "I'm just the Mordeshur (مرده‌شور)" — the person who washes the body for burial without concerning themselves with whether the soul goes to heaven or hell. A conservator cannot be a Mordeshur. The conservator must care — not just about the material form, but about the idea of the work.

At its core, the conservator's role is one of decision-making: determining what has value, what is at risk, what should be preserved and how, and documenting those decisions so that future generations can understand, revise, or build on them. The ICOM-CC definition of the profession makes this explicit: only the conservator-restorer can "correctly interpret the results of such examinations and foresee the consequences of the decisions made," and the recommendation of whether to intervene "can be made only by a well trained, well educated, experienced and highly sensitive conservator-restorer" — not by a craftsman or technician alone (ICOM-CC, 1984). The distinction matters because it locates the profession not in manual skill but in judgement.

This judgement is exercised across several overlapping roles. The conservator examines an object and assesses its condition and the risks it faces; weighs its significance, drawing on the different and sometimes competing values an object can hold (Riegl, 1903); decides whether and how to intervene, balancing the competing demands of revealing, investigating, and preserving the object (Caple, 2000); carries out or supervises treatment; and records the reasoning behind each choice. Contemporary theory has widened this picture: Muñoz Viñas (2005) reframes conservation as the negotiation of the meanings and values that different stakeholders attach to an object, so that decisions are no longer measured against a single material "truth" but against what a community agrees the object is. Appelbaum (2007) makes a parallel methodological point, arguing that treatment follows only after the conservator has characterized the object and established the state in which it should be held. In each account, documentation is not an afterthought but the medium through which judgement becomes transmissible.

These roles intensify rather than dissolve when the object is software. A software-based work has no single stable state; preserving it means deciding which of its behaviors and dependencies define the work and which may change — a question that can only be settled through the kind of judgement described above (Laurenson, 2006, Conclusion section; Rinehart & Ippolito, 2014). It is the conservator understood in this way — as the one who examines, values, decides, and documents — whose perspective the present study brings to the EaaS interface.

2.6 Conservation Thinking in a Digital Context

Conservation thinking rests on a long tradition of value theory. John Ruskin (1849) argued that buildings should be left to decay rather than restored, because restoration destroys the authenticity that time has given them. Eugène Viollet-le-Duc (1854–1868) argued the opposite: restoration should bring the object to a state of completeness it may never have had, realizing its ideal form. Alois Riegl (1903) offered a more nuanced framework, distinguishing between *age value* (the patina of time), *historical value* (the object as evidence of a specific moment), *intentional commemorative value*, and *use value* — each making different demands on the conservator.

These are not historical curiosities. They are live questions in digital preservation. When a conservator emulates a 1993 CD-ROM artwork, Riegl's competing values resurface: should the emulated experience preserve the traces of time — the glitches, the slow loading, the degraded media (age value)? Should it faithfully document the work's original state (historical value)? Or should it be restored to an ideal, complete form that runs smoothly in a modern browser (newness value)? These values “never exist in purity; they always appear in conjunction and thereby in constant competition” (Lehne, 2010, p. 79, as cited in Harrer, 2017, p. 31).

The vocabulary a preservation tool uses shapes which of these values become visible — and which remain unspoken. The challenge is understanding whether the Demo-UI's technical vocabulary can support this kind of conservation reasoning.

2.7 Digital Material and the Need for Technical Literacy

It is tempting to think that digital preservation is fundamentally different from physical conservation. But as Cather (2006, pp. 89–93) argues, conservation methodology is trans-technological: the same principles apply regardless of the material. The conservator must understand materials (software dependencies, file formats, hardware requirements), understand history (how the work was created, exhibited, and previously preserved), and proceed through iterative intervention — successive approximations rather than a single definitive solution.

Materiality has always been central to conservation: to preserve an object, the conservator must first understand what it is made of and how that substance behaves over time. Software-based works are no exception. Their material is layered — the physical media on which ones and zeros are stored, the processors that execute them, and the screens that display them, together with the software layers that make the work run — and every one of these is subject to obsolescence and decay (Rothenberg, 1999). What differs from traditional conservation is not the principles but the knowledge required to apply them: a conservator working with software-based art has to understand both conservation methodology and computing infrastructure.

It is this second body of knowledge that the Demo-UI takes for granted. When a conservator trained in material science and art history opens it for the first time, they encounter a vocabulary that assumes a different education — environment, object, disk image, frameskip, CIDR notation (see Table 3, Chapter 4). Nothing in the interface explains what these terms mean for conservation, or how they bear on the decisions a conservator has to make. The result is a barrier — not because the conservator lacks competence, but because the tool was not built to speak to that competence.

The barrier matters because it determines who gets to do the work. If only those with computer science training can operate the Demo-UI, then the conservation decisions embedded in every emulated environment — which emulator, which resolution, which compromises — are made by engineers rather than conservators. This inverts the conservator–engineer relationship, since decisions about intervening in the software are precisely the judgements that the profession reserves for the conservator (ICOM-CC, 1984; see Section 2.5).

This cuts two ways. Part of the answer lies in the vocabulary itself — in how a tool like the Demo-UI names things. This thesis suggests ways a conservator might read and use its existing terms more deliberately, and where the interface could be improved in the future. But much of the toolchain a software conservator relies on consists of general-purpose technical tools that were never built for conservation and are not going to be: disk images are made with utilities such as dd, file formats identified with tools such as Siegfried, and obsolete environments run on emulators written for other contexts entirely. Technical literacy is therefore a permanent part of software conservation, not something better tools will remove. By acquiring enough of this literacy to read these terms — to understand the computing vocabulary rather than defer to it — a conservator can hold on to the decisions an emulated work requires even where the tool offers no help.

2.8 Metadata Standards in Digital Preservation

Yet standards for structuring the kind of information necessary for digital conservation already exist. Over the past two decades, the digital preservation community has developed a set of international standards to document what the Demo-UI ought to record: what an object is, what has been done to preserve it, who did it, and why.

These standards form a layered system. At the top is OAIS (Open Archival Information System, ISO 14721), a reference model that describes what any long-term archive should do — ingest, store, manage, plan, and provide access. OAIS does not tell an institution *how* to build its archive; it defines what *functions* the archive needs. Below OAIS sit implementation-level standards that carry out those functions in practice.

Metadata Encoding and Transmission Standard (METS), maintained by the Library of Congress, is a container format. It bundles together all the metadata about a digital object — description, preservation history, file inventories, structural relationships — into a single package. METS does not define its own vocabulary for describing objects or preservation actions. Instead, it wraps other standards inside it.

For description — what is this object? — the most widely used standard is Dublin Core (ISO 15836). It defines fifteen simple elements: Title, Creator, Date, Description,

Format, and so on. Dublin Core is deliberately generic: it can describe a painting, a dataset, or a disk image. It covers identification, not preservation.

For preservation — what has been done to keep this object accessible? — the standard is Preservation Metadata: Implementation Strategies (PREMIS), also maintained by the Library of Congress (current version 3.0, 2015). PREMIS defines five core entities: **Objects** (the digital files being preserved), **Events** (actions performed on those files — a format migration, a virus scan, an emulation session), **Agents** (who or what performed the action — a person, an organization, or a piece of software), **Rights** (what the repository is permitted to do), and **Intellectual Entities** (the conceptual work that the files represent). The Events and Agents entities are directly relevant to conservation: PREMIS provides a structured way to record that a specific preservation action was taken, by a specific person, at a specific time, for a specific reason, with a specific outcome (Library of Congress, 2015).

Together, these standards address many of the gaps identified in this thesis (Table 2).

Table 2: *Digital Preservation Metadata Standards and Conservation Needs*

Conservation need	Standard	How it is addressed
What is this artwork?	Dublin Core	Title, Creator, Date, Description
What has been done to preserve it?	PREMIS Events	Action type, date, outcome, rationale
Who made this preservation decision?	PREMIS Agents	Person, organization, or software identified
How do the files relate to each other?	METS	Structural map linking disk images, configurations, documentation
What is the overall preservation strategy?	OAIS	Conceptual framework for ingest, storage, access

As introduced in Section 2.4.3, the EaaSI program built on the EaaS framework to create a national infrastructure for emulation-based preservation across American institutions. Its current development focuses on user management and how institutions share and control access to emulated environments (EaaSI, n.d.). Conservation metadata — who made a preservation decision, why, and on what basis — does not appear as a development priority. With the structured sundown of the Software Preservation Network (2026), whose Metadata Working Group developed crosswalks mapping standards like PREMIS and Dublin Core to software description requirements (Software Preservation Network, n.d.), it remains unclear whether these metadata efforts will be integrated into the EaaSI platform.

3 Method

This chapter describes the methodological approach of the thesis: how the study was designed, which methods were used, what material was analyzed, and why these choices are appropriate for the research questions. A conservator's work with software objects requires examination, assessment, and documentation — the same as with any other object. The chapter also introduces the Demo-UI and the three software objects that form the basis of the analysis in Chapter 4.

3.1 Research Design

The study follows a qualitative, interpretive research design: it works from observable material — the interface, its vocabulary, and three software objects preserved through it — and asks what the terms mean, how a conservator would read them, and what consequences they carry for conservation practice. Two methods are combined: a comparative terminology analysis across three related systems, and a case-study walkthrough of three software objects. The first examines the Demo-UI's vocabulary as a whole and against alternatives; the second tests that vocabulary against the specifics of individual works, so that the analysis moves continually between the general and the particular.

The analysis proceeds in three stages, from which two contributions follow. First, the vocabulary of the Demo-UI is inventoried: the terms, fields, and labels a conservator encounters in the interface are identified and recorded (Section 4.1, Table 3). Second, this vocabulary is compared with that of two related systems — Rhizome's ArtBase, a Wikibase catalog of born-digital art that links to the Demo-UI, and Stabilize, an alternative EaaS interface oriented toward institutional software preservation — to see how the same preservation work can be organized through different vocabularies (Section 4.2). Third, the vocabulary is tested in practice through a walkthrough of three software objects — *Tod dem Fernsehen, Bomb Iraq*, and *Every Icon* — recording field by field how the interface behaves with real material and surfacing issues that become visible only in use (Section 4.3). From these stages the study derives its two contributions: a mapping of which existing free-text fields can already hold conservation information (Section 4.4), and a set of suggested clarifications for interface elements that are unclear without technical knowledge (Section 4.5).

The analysis is introspective: it documents how the interface is read, not how a sample of users would read it. This gives the study an insider vantage point that supports close, accurate description of systems that are otherwise difficult to access, while carrying assumptions an outsider might question. That positionality, and the way it is handled, is discussed in Section 3.4.

3.2 Methods

The study uses three methods:

Terminology inventory. Every field, button, tab, and setting visible in the Demo-UI were systematically documented. The inventory was carried out on the Rhizome instance (eu.r4.rhizome.computer, build v2021.10-393-gd54e3a4, UI-Build 6FC8C59C35) and recorded in mockup reconstructions (presented in Section 4.1).

The purpose was to establish what vocabulary the interface uses, how it is organized, and what it asks the user to provide.

Comparative terminology analysis. The Demo-UI's vocabulary was compared with two related systems: the ArtBase catalog (Rhizome's Wikibase-based collection database) and Stabilize (a commercial EaaS platform for institutional software preservation). The comparison identifies where the same preservation concepts are named differently, where one system makes distinctions the other does not, and where conservation-relevant information is present in one system but absent from another (see Section 4.2).

Case study walkthrough. Three software objects — *Tod dem Fernsehen* (R.A.M.S., 1993–1994), *Bomb Iraq* (Cory Arcangel, 2005), and *Every Icon* (John F. Simon Jr., 1997) — were examined through the Demo-UI. Each work was opened, and every field value was documented across the Details, Advanced Settings, and Revisions tabs (Table 6 to Table 10). The case studies test the vocabulary in practice: they show where the interface holds information that the conservator can use, where it falls short, and where existing fields can carry conservation information that they were not designed for.

3.3 Material

The primary material consists of three (3) digital systems and three (3) software objects:

Systems:

1. **The EaaS Demo-UI** (Rhizome instance). The web-based administration interface through which emulated environments are configured, run, and documented. Accessed at eu.r4.rhizome.computer.
2. **The ArtBase catalog.** Rhizome's structured catalog for born-digital artworks, built on Wikibase. Used as a comparative reference for how artworks are described outside the Demo-UI.
3. **Stabilize.** A commercial platform extending the EaaS framework, oriented toward institutional software preservation — including legacy hardware-dependent workflows such as microscopy software. Analyzed through wireframes by Rossenova (2020) and the author's professional knowledge of the platform.

Software Objects:

1. **Tod dem Fernsehen** (R.A.M.S., 1993–1994). VHS cassette and interactive CD-ROM with VRML elements. Preserved at TMW; no ArtBase entry.
2. **Bomb Iraq** (Cory Arcangel, 2005). HyperCard readymade found on a used Macintosh TV. ArtBase Q4893.
3. **Every Icon** (John F. Simon Jr., 1997). Java applet. ArtBase Q2428.

The three case studies are already preserved through EaaS at Rhizome and accessible to the author through professional engagement with the EaaS ecosystem. Together they span three different preservation situations: a multi-format work spanning VHS and CD-ROM with no catalog entry (*Tod dem Fernsehen*), a found-object system image where the computing environment itself is part of the work (*Bomb Iraq*), and a browser-hosted applet whose integrity depends on computational behavior rather than visual appearance (*Every Icon*). They also represent different relationships to the archive: *Tod dem Fernsehen* is held by TMW with no ArtBase entry, *Bomb Iraq* has full ArtBase metadata (Q4893), and *Every Icon* has an ArtBase entry (Q2428) but depends on the applet running correctly — at the right speed, from the right starting position — rather than on recreating a complex environment.

3.4 Justification of Methods

The research questions concern meaning and professional relevance rather than statistics or correlations, so a qualitative, interpretive approach is appropriate. The design pairs a comparative terminology analysis with a set of case studies because each answers a different need. The comparison — setting the Demo-UI beside the ArtBase and Stabilize (Section 3.4.2) — shows that its vocabulary is one design choice among several, shaped by its computer science origins, rather than the only way the work could be named. The case studies then ground that analysis in practice, surfacing findings that emerge only from working with the material, such as duplicate environments, mixed-language revision histories, and missing software listings.

The study's insider access — to the Rhizome instance and to the teams that develop and use it — allows a level of detail not available to an external researcher. That access also carries a risk: some observations may reflect assumptions shared within the EaaS community rather than views held more widely, and the analysis is written with that in mind.

3.4.1 Limitations

This study has two limitations that deserve explicit acknowledgment. First, the analysis is based on the Demo-UI as deployed by Rhizome (build v2021.10-393-gd54e3a4, UI-Build 6FC8C59C35), the institution's active preservation interface; other institutions running EaaS, such as Yale's EaaSI deployment, may have customized their interface, and the findings do not claim to describe all EaaS installations. Second, the terminology analysis is introspective: it records how the interface is read by a researcher trained in both conservation and computing, not how a conservator without that technical background would encounter the same terms.

3.4.2 Why ArtBase and Stabilize

ArtBase and Stabilize were chosen because each stands in a direct relationship to the Demo-UI while naming the work differently. ArtBase is Rhizome's Wikibase-based catalog for born-digital art, with a deployed vocabulary of over 140 properties (Section 4.2.1); it describes many of the same works the Demo-UI preserves and links to it through dedicated EaaS-bridging properties, though it is not itself built on the EaaS framework. Stabilize, by contrast, is built on the same EaaS technology as the Demo-UI — developed by OpenSLX at the University of Freiburg — but is oriented toward institutional software preservation, including commercial software such as microscopy

tools, with a proposed vocabulary visible in wireframes (Rossenova, 2020). A more general standard would not serve the same purpose: OAIS (Section 2.8), for instance, is used by conservators to guide preservation documentation across institutions — Almut Schilling, a freelance conservator working with several Viennese museums, uses it to structure the preservation information she writes for each institution (A. Schilling, personal communication, April 15, 2026) — but it operates at a different level, defining what functions an archive should fulfill rather than how a specific tool names its objects and fields. ArtBase and Stabilize work at exactly that concrete level: between them they offer two contrasting vocabularies for the same software objects — one from conservation practice, one from institutional software preservation.

3.5 Introducing the Case Study Objects

Before analyzing the Demo-UI in practice (Section 4.3), the three software objects used in the study need to be introduced. Each poses different preservation challenges and each is already preserved through Rhizome's EaaS interface (the Demo-UI). Together they span a range of situations: a multi-format multimedia CD-ROM, a found-object readymade on a dead machine, and a conceptually infinite Java applet.

3.5.1 Tod dem Fernsehen (R.A.M.S., 1993–1994)

- **Title:** Tod dem Fernsehen (Death to Television)
- **Artists:** R.A.M.S. — Margarete Jahrmann, Max Moswitzer, and Alf Altendorf
- **Date:** 1993–1994
- **Medium:** VHS cassette and interactive CD-ROM; multimedia work with VRML elements
- **Collection:** TMW Digital Collection (no ArtBase entry)

Tod dem Fernsehen is part video, part interactive CD-ROM (Altendorf, 1994). The work emerged from the Viennese media art scene around Public Netbase and critical net culture, at a moment when the CD-ROM was a serious artistic medium: it could hold video, 3D, sound, and software on a single disc.

The CD-ROM opens onto a grid of thumbnail images — video stills, test patterns, distorted faces, color bars — that serve as an interactive navigation map. Clicking a thumbnail launches the corresponding content. Much of the experience takes place inside VRML 3D environments: the user moves through textured, rippling surfaces where photographic imagery of faces is wrapped around three-dimensional forms. A small color-bar element in the lower-left corner provides navigation controls. The aesthetic is deliberately raw — low-resolution video, raw sounds, early 3D rendering, saturated color — reflecting both the technical constraints and the experimental ambition of mid-1990s multimedia art (Figure 1).

Figure 1. Tod dem Fernsehen (R.A.M.S., 1993–94) running in the emulated BasiliskII environment. Top left: the CD-ROM navigation grid. Top right, bottom left, bottom right: three views from the VRML 3D environments.

For preservation, *Tod dem Fernsehen* poses a specific challenge: it exists across two physical formats (VHS and CD-ROM) and depends on software that is no longer maintained. The work belongs to the Technical Museum Vienna (TMW), not to Rhizome, so it has no entry in Rhizome's ArtBase catalog. Everything the conservator needs to know about this work — what it is, what condition it is in, why the emulator settings were chosen — must come from the Demo-UI or from memory.

3.5.2 Bomb Iraq (Cory Arcangel, 2005)

- **Title:** Bomb Iraq
- **Artist:** Cory Arcangel
- **Date:** 2005
- **Medium:** HyperCard readymade — found program on a used Macintosh TV
- **Collection:** Rhizome ArtBase (Q4893)

In 2005, Cory Arcangel bought a used Macintosh TV at a Salvation Army store in Buffalo and found a HyperCard program on the hard drive. HyperCard was an application shipped with every Macintosh in the late 1980s and 1990s that let users create interactive “stacks” of linked cards combining graphics, text, and simple programming. The program on this machine contained hand-drawn bombs dropping on a clipart map of Iraq, navigated by flipping through a virtual deck of cards (Figure 2;

Arcangel, 2005). The computer had belonged to a family whose teenage son used it for school assignments. Arcangel exhibited the found machine as a readymade at PaceWildenstein, New York. The original hardware died three days into the exhibition, but the hard disk contents were saved onto a CD-ROM.

Dragan Espenschied, Rhizome's Digital Preservation Director, later restored the work using EaaS (Rhizome, 2014). Critically, Espenschied chose to preserve not just the HyperCard program but the entire computing environment as the previous owner had left it — the modified system font, the desktop background, the arrangement of icons, the installed software. Rhizome calls this "system ambience": the traces of a person's life expressed through their computer's configuration. The teenager's choice of font, the school assignments sitting alongside the bomb program, the desktop wallpaper — together they form the context without which the artwork loses a dimension of its meaning. The work has since been exhibited at MoMA PS1 (*Theater of Operations*, 2019) and Firstsite (*BACK OFF*, 2019).

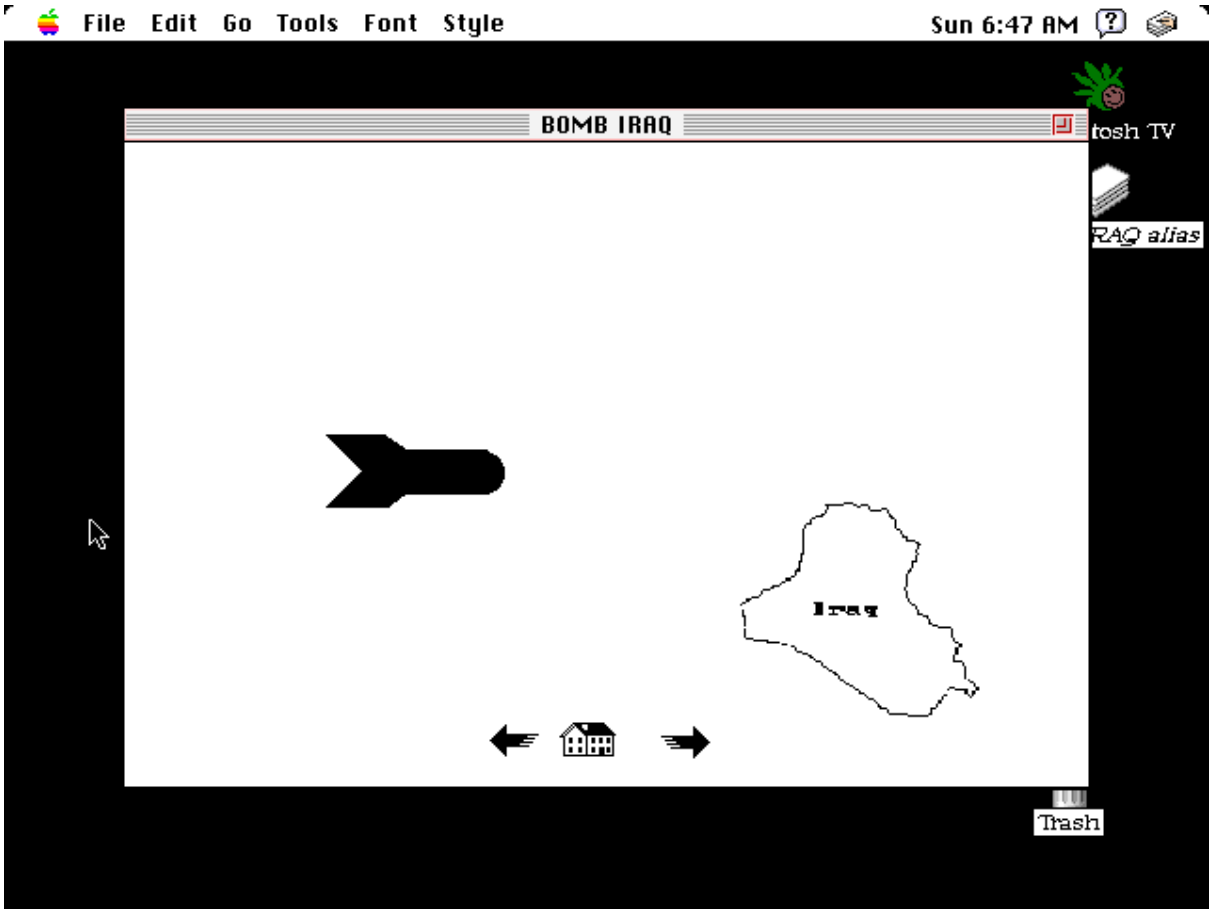
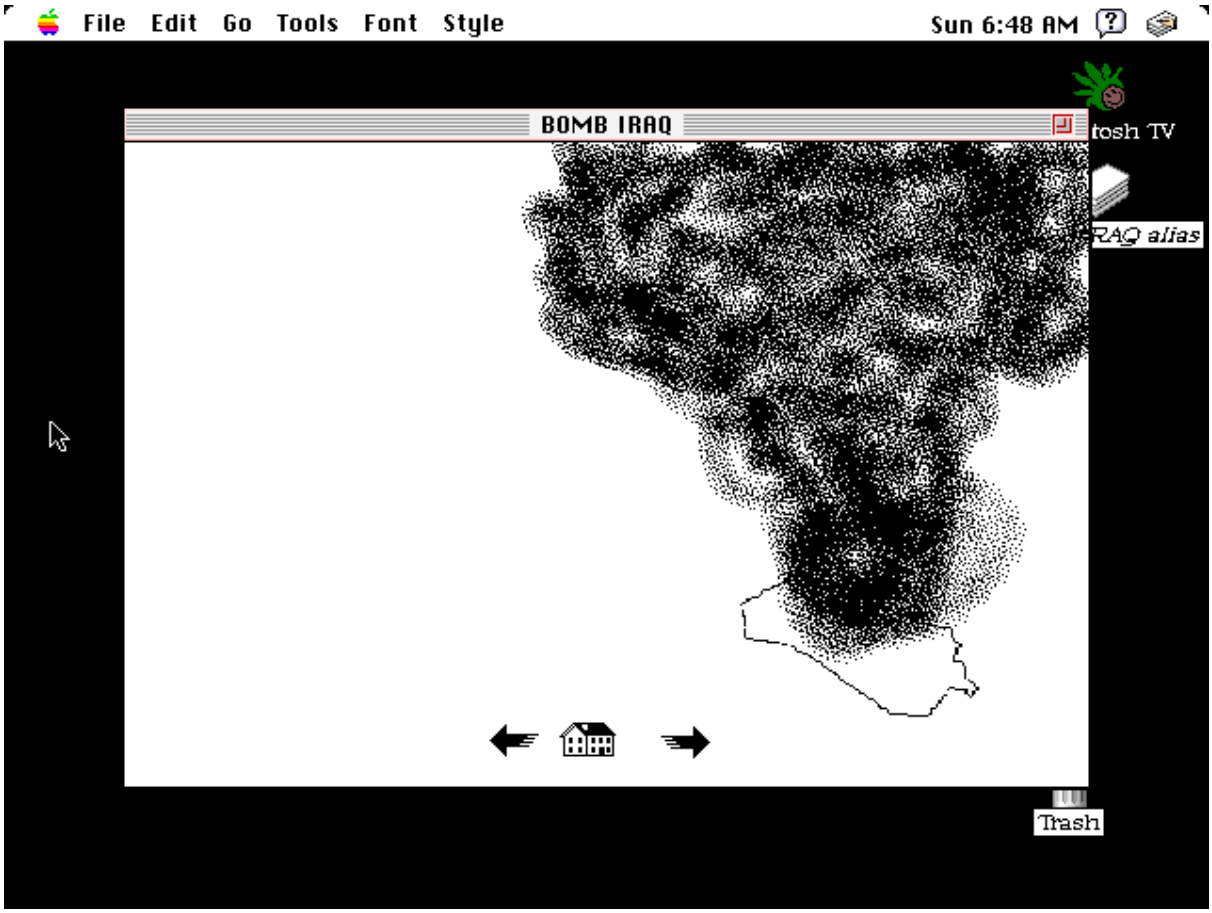


Figure 2. Bomb Iraq (Cory Arcangel, 2005) running in the emulated BasiliskII environment. Above: the first HyperCard card. Below: the last card. The Macintosh TV desktop is visible around the HyperCard window.



(Figure 2, continued)

For preservation, *Bomb Iraq* is notable because the artwork includes not just the HyperCard program but the entire computing environment — the desktop, the fonts, the other files. The "system ambience" is part of what was preserved.

3.5.3 Every Icon (John F. Simon Jr., 1997)

- **Title:** Every Icon
- **Artist:** John F. Simon Jr.
- **Date:** 1997
- **Medium:** Java applet
- **Collection:** Rhizome ArtBase (Q2428)

Every Icon is a Java applet that systematically displays every possible arrangement of a 32x32 pixel grid, starting from the top-left corner and working through all combinations (Simon, 1997; Rhizome ArtBase, n.d.-b). It will not finish in any human timeframe — the number of possible grids exceeds the age of the universe. The artwork is not a picture. It is a process. What matters is not what it looks like at any given moment but that it runs correctly: starting from the right position, advancing at the right speed, following the right sequence (Figure 3).

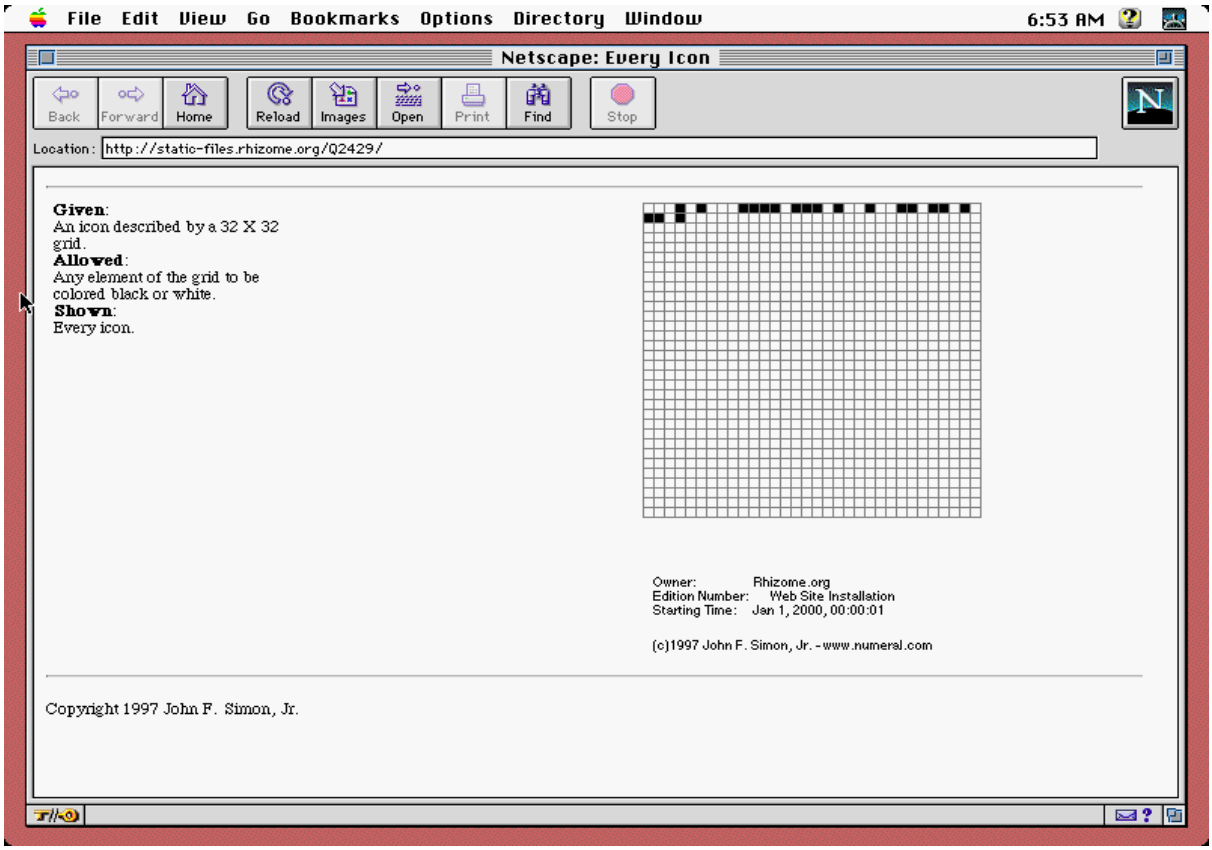


Figure 3. Every Icon (John F. Simon Jr., 1997) running in Netscape Navigator Gold 3.04 inside the emulated BasiliskII environment.

For preservation, *Every Icon* is technically simple — a single Java applet — but the preservation question is unlike the other two case studies. The challenge is not reconstructing a complex environment but ensuring that the emulated version runs correctly: same starting position, same speed, same sequence. The Java runtime version affects execution speed, which alters the temporal experience of the work. None of this is visible in the Demo-UI.

4 Results

As a contribution of this thesis, Section 4.1 identifies where existing free-text fields in the interface can hold conservation information, and Section 4.5 suggests concrete clarifications — expanded abbreviations, tooltips, and bug fixes — for interface elements that are unclear without prior technical knowledge.

4.1 Terminology Inventory of the Demo-UI

The EaaS framework ships with a web-based administration interface called the Demo-UI — short for “demonstration user interface.” Despite the name, it is not a prototype: the Demo-UI is the baseline interface through which institutions like Rhizome manage emulated environments and access preserved software. Other deployments build on it — EaaSI (see Section 2.4.3) adds institutional features — but the Demo-UI's vocabulary, established when the interface was first committed in 2015 (Russler, 2015), is the foundation they all started from. The following inventory documents every term visible in the interface, based on the Rhizome instance (eu.r4.rhizome.computer, build v2021.10-393-gd54e3a4, UI-Build 6FC8C59C35). Figure 4 is a mockup that reproduces the interface's layout, vocabulary, and data.

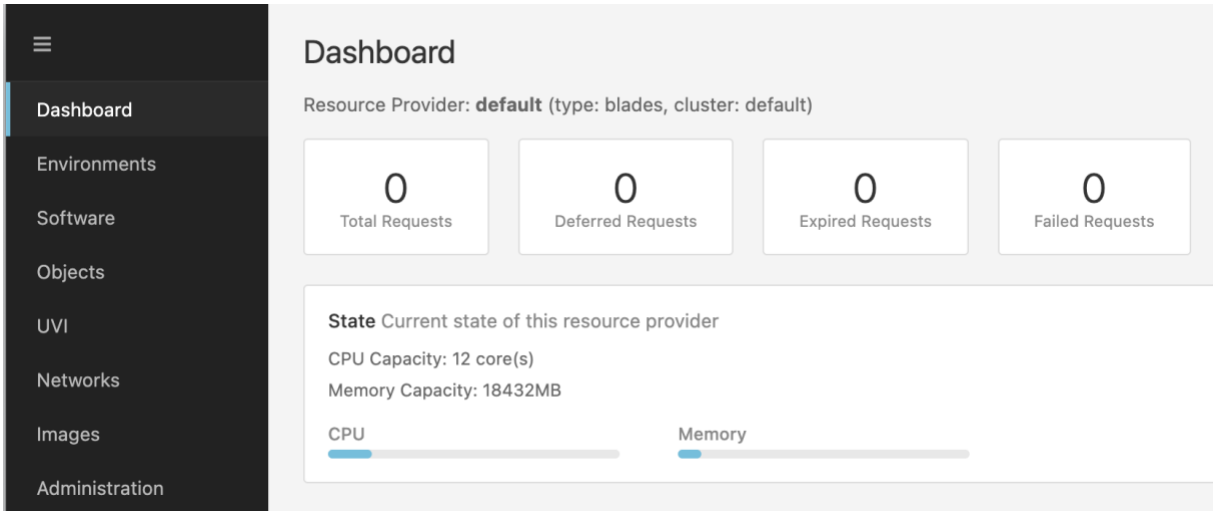
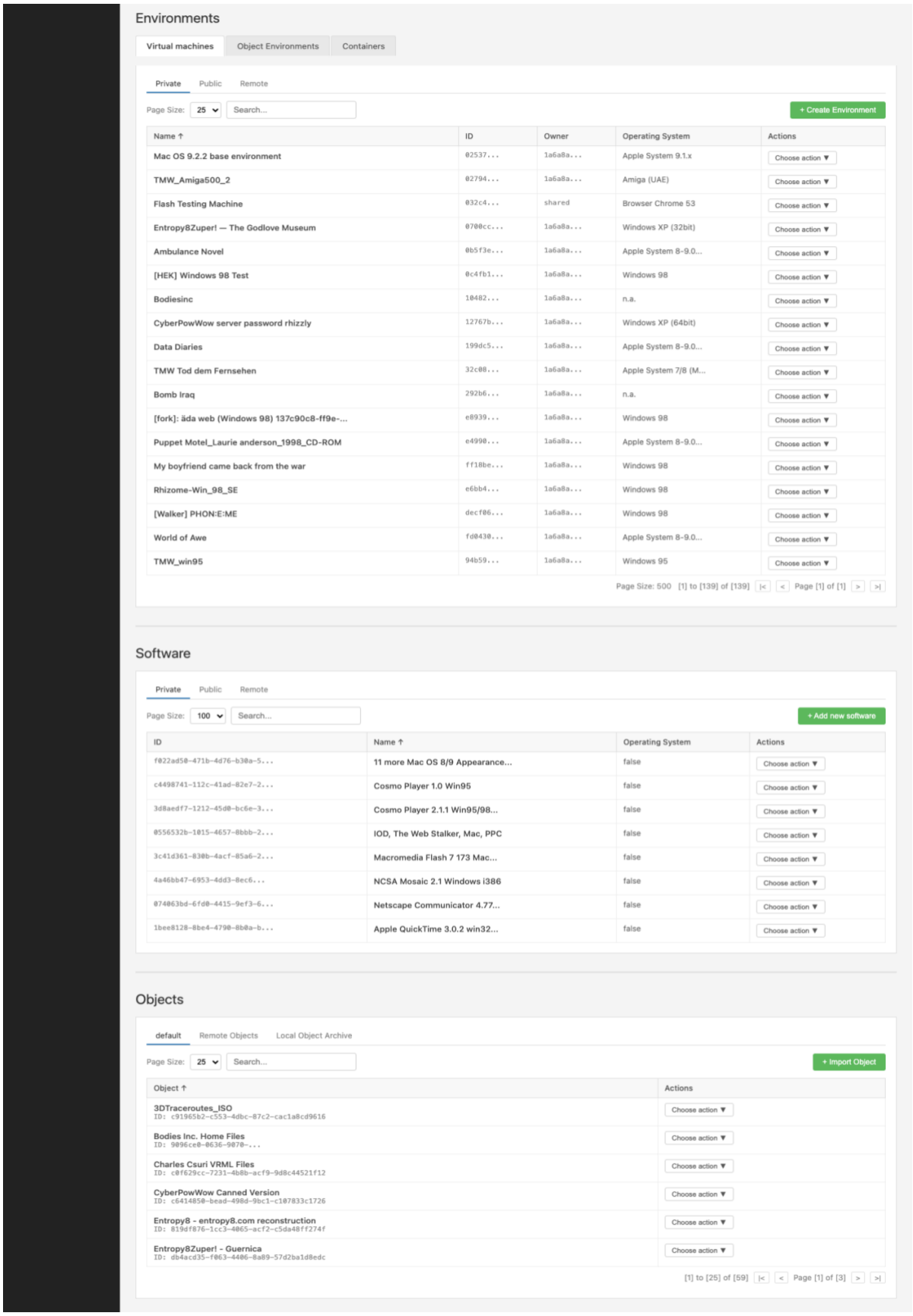
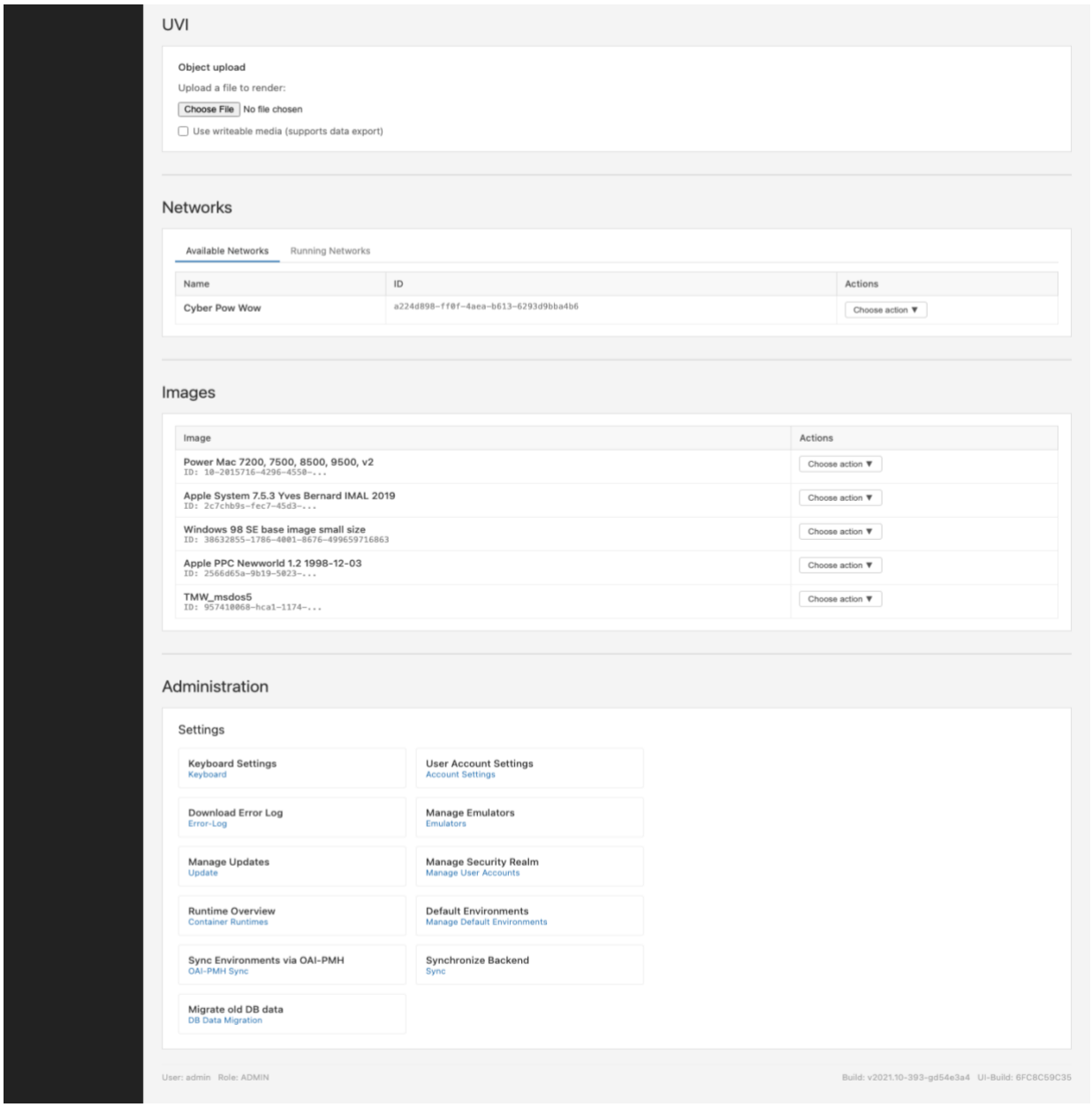


Figure 4. Mockup of the EaaS Demo-UI, reproducing the layout, vocabulary, and data from the Rhizome instance. All eight navigation tabs shown.



(Figure 4, continued)



(Figure 4, continued)

The interface is organized around eight navigation tabs, each introducing its own set of terms. Table 3 summarizes what each tab contains.

Table 3: *Demo-UI Navigation Tabs and Key Terms*

Tab	Key Terms	Description
Dashboard	Resource Provider, Requests, CPU/Memory Capacity	Server health metrics. No information about software objects/artworks or preservation.
Environments	Virtual machines, Object Environments, Containers; Private/Public/Remote; Name, ID, Owner, OS, Actions	The primary working area. Lists all emulation configurations in one flat list, with no distinction between their different purposes.

Tab	Key Terms	Description
Software	ID, Name, Operating System (“true”/”false”), Actions	Installable software packages. The “Operating System” column shows “true” if the entry is an operating system, “false” if it is not.
Objects	default, Remote Objects, Local Object Archive; Object name, ID, Actions	Uploaded digital files. An artwork’s files and a software installer are both listed as “objects” with no way to tell them apart.
UVI	Object upload, “Upload a file to render”, “Use writeable media”	Universal Virtual Interactor (Cochrane et al., 2019). The user uploads a file and the system opens it inside an emulated environment.
Networks	Available/Running Networks, “Add Machines to the Network”, “Environments with network”, Network (CIDR), DNS/DHCP	Connects multiple emulated systems in a virtual network. Uses both “machines” and “environments” for the same things.
Images	Image name (with ID), Actions	Lists disk images used by environments. Each entry shows a name and ID but no indication of which environment uses it.
Administration	Keyboard Settings, User Account Settings, Download Error Log, Manage Emulators, Manage Updates, Manage Security Realm, Runtime Overview, Default Environments, OAI-PMH Sync, Synchronize Backend, Migrate old DB data	Eleven system-level settings panels for managing the EaaS server.

Across all eight tabs, the Demo-UI’s vocabulary is technical. The terms describe how to configure and run emulated systems, not how to document the preservation of cultural objects.

4.1.1 The Environments Tab in Detail

The Environments tab lists emulation configurations in sub-categories (Virtual machines, Object Environments, Containers) with three visibility levels (Private, Public, Remote). When an environment is opened, the detail view (Figure 5) reveals three sub-tabs: **Details** (ID, name, OS, “Last change” free-text field, runtime options, Drive Settings), **Advanced Settings** (emulator and configuration), and **Revisions** (version history with Fork and Revert).

The Details tab shows the environment's UUID, modification date, name, operating system, a “Last change” free-text field, runtime checkboxes, and Drive Settings listing the attached disk and media. The field labeled “Last change” is in fact a misnomer: in the Demo-UI's source code, this field is stored as the environment's *description* — a naming convention inherited from the EMiL project (R. de Vries, personal communication, 2026). Because environments can have parent environments, this description field has been repurposed as a revision history. The auto-date feature visible in some revision entries was implemented as a single-line modification in the

Demo-UI frontend, because adding it properly to the EaaS server was considered too complex (de Vries, 2026). This is a concrete example of how the Demo-UI's terminology has been shaped by technical convenience rather than deliberate design — and it means one of the very few free-text fields available to a conservator for documenting their work carries a label that misrepresents its function.

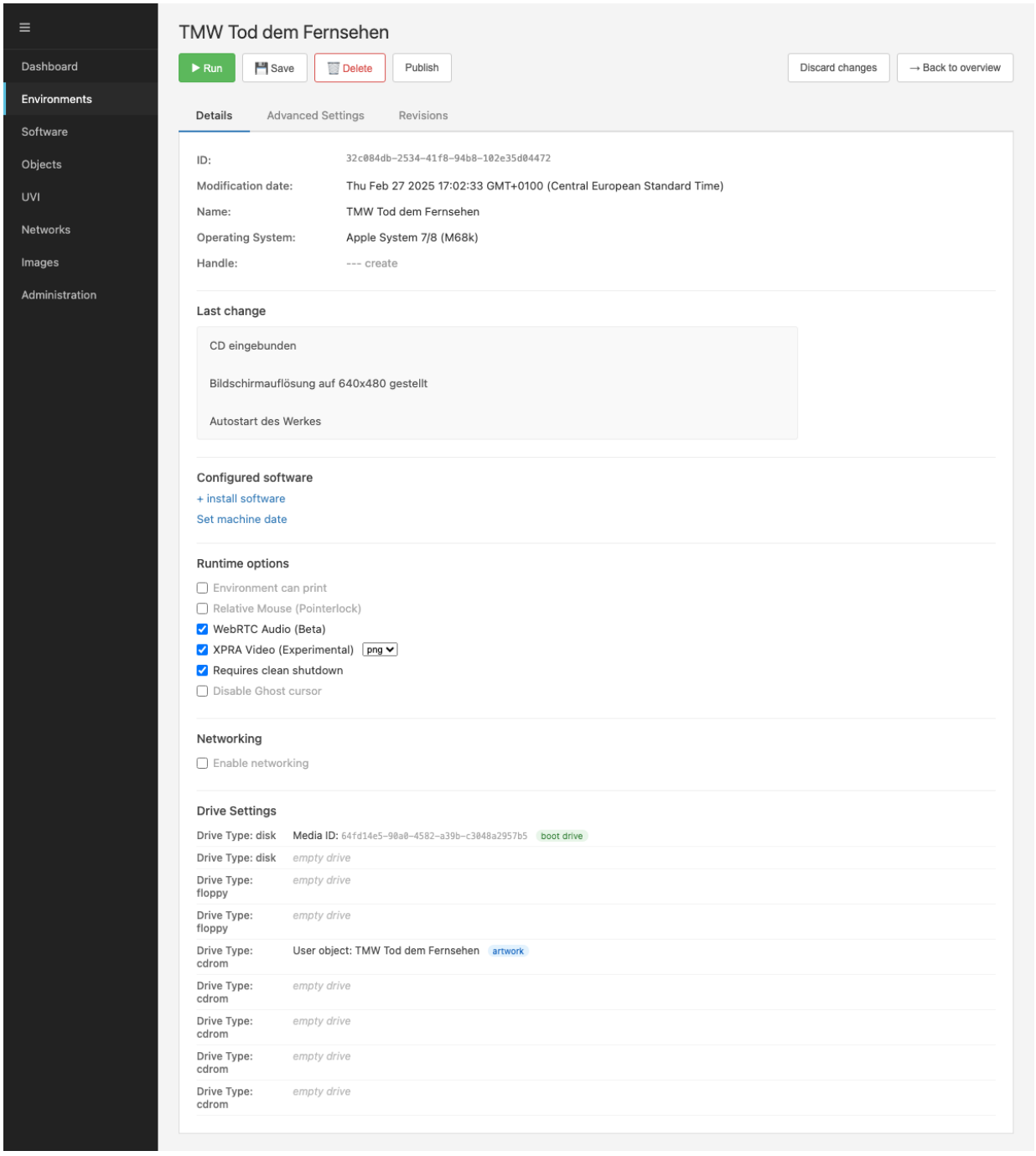
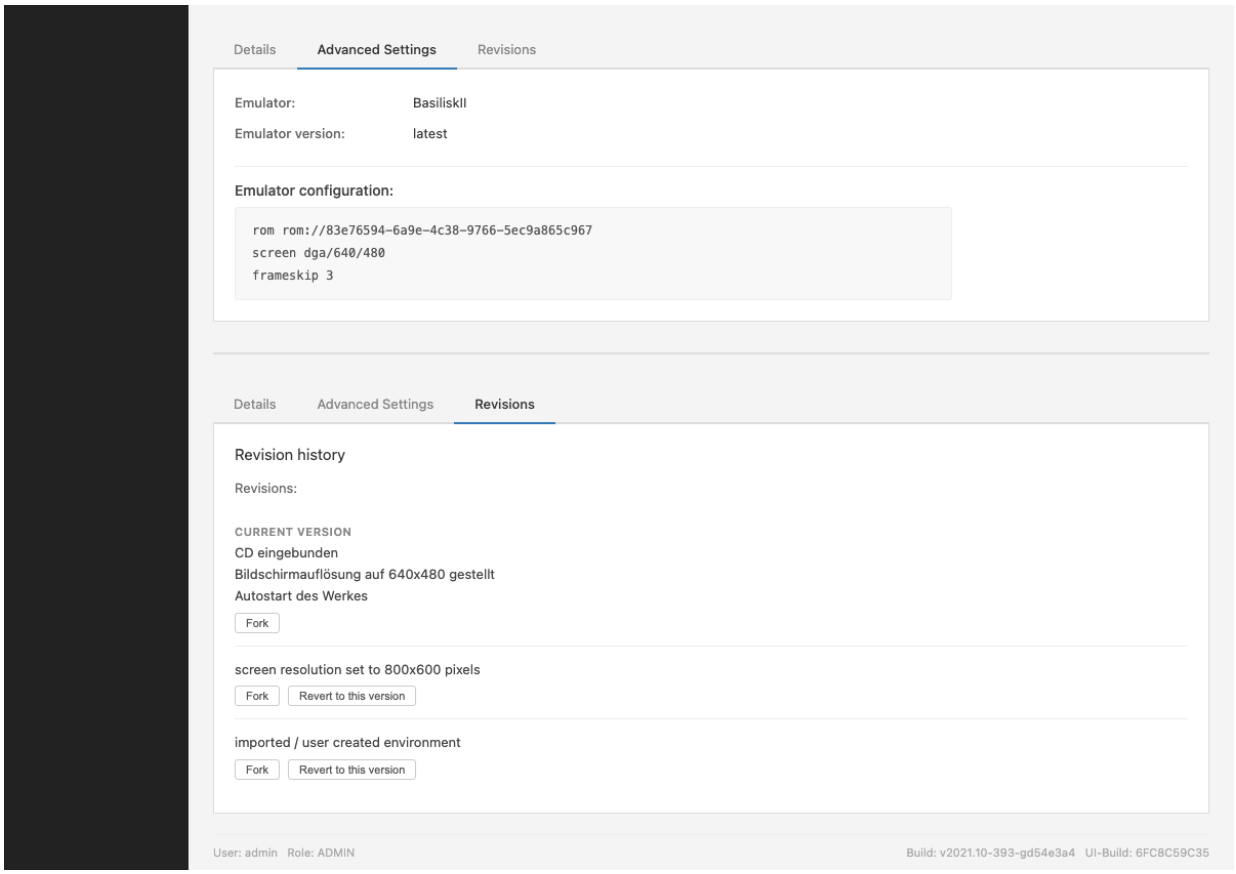


Figure 5. Mockup of the Demo-UI environment detail view for TMW Tod dem Fernsehen. Three tabs: Details, Advanced Settings, and Revisions.



(Figure 5, continued)

When an environment is launched via “Run,” a session menu appears with controls: Actions, Screenshot, Download Print Jobs, Save Environment, Checkpoint, Send Ctrl-Alt-Del, Send Esc, Restart Session, Connection Info, and Stop. “Checkpoint” saves the current state; “Save Environment” preserves changes made during the session.

4.2 Comparative Terminology Analysis: ArtBase and Stabilize

The ArtBase catalogs many of the software objects that are preserved through Rhizome's Demo-UI, and Stabilize is an alternative EaaS interface oriented toward institutional software preservation. Comparing their vocabularies with the Demo-UI's shows where the same concepts are named differently and where conservation-relevant distinctions exist in one system but not another.

4.2.1 The ArtBase

Rhizome maintains the ArtBase, a Wikibase-based catalog for its collection of born-digital artworks (Rossenova, 2021). While this thesis uses the broader term *software object* (see Section 2.2.1), the ArtBase uses a more granular conservation ontology. Where the Demo-UI lists all digital content as undifferentiated “Objects” (see Figure 4 / Table 3), the ArtBase distinguishes three levels:

- **Artwork** — the conceptual work (e.g., Q3958 for The Web Stalker). Properties for artist, inception date, description.

- **Variant** — a specific version, linked via “has variant” (P45). An artwork can have multiple variants: archived copy, emulated version, video documentation.
- **Artifact** — the active data files. Property P139 (“artifact”) is defined as “the data part of an artifactual part of a variant, such as disk images, web archives, etc.”

The Demo-UI does not make these distinctions. Table 4 lists a selection of ArtBase properties that are relevant to conservation work but have no equivalent in the Demo-UI — covering areas like attribution, condition, risk, and preservation history.

Table 4: Conservation-Relevant ArtBase Properties (absent from the Demo-UI)

Property	ID	Conservation Function
instance of	P3	Distinguishes artworks from variants and artifacts
artist	P29	Attribution
has variant / variant of	P45 / P56	Artwork–variant relationships
made of	P81	Material composition as file formats
archival plan	P121	Strategy: reconstruction, repair, re-enactment
generated by	P117	Preservation rationale
associated with	P118	Provenance: who created this variant
risk from external links	P88	Risk: web dependencies
risk from external media	P94	Risk: media format dependencies
risk from external services	P95	Risk: third-party service dependencies
completeness	P100	Condition assessment with controlled values
Stoplight Value	P99	Traffic-light condition indicator
derived from	P102	Preservation history chain
reperformance platform	P128	EaaS or Webenact environment used
browser plug-ins	P86	Dependency documentation

The ArtBase also created EaaS-bridging properties (P139–P148) to connect its model to the Demo-UI’s infrastructure. These properties allow a conservator working in the ArtBase to reference emulation environments, but the connection does not work in the other direction.

4.2.2 Stabilize

Stabilize (stabilize.app) is developed by OpenSLX at the University of Freiburg — the same team behind the EaaS backend, including the Demo-UI. It uses emulation to give users access to old hardware-dependent workflows, such as running legacy scanners or specialized equipment that requires a specific operating system and software setup. It is positioned as a commercial service for businesses and institutions. It builds on the

same emulation technology as the Demo-UI. Figure 6 shows the Stabilize Projects overview, based on Lozana Rossenova's wireframes (January 2020).

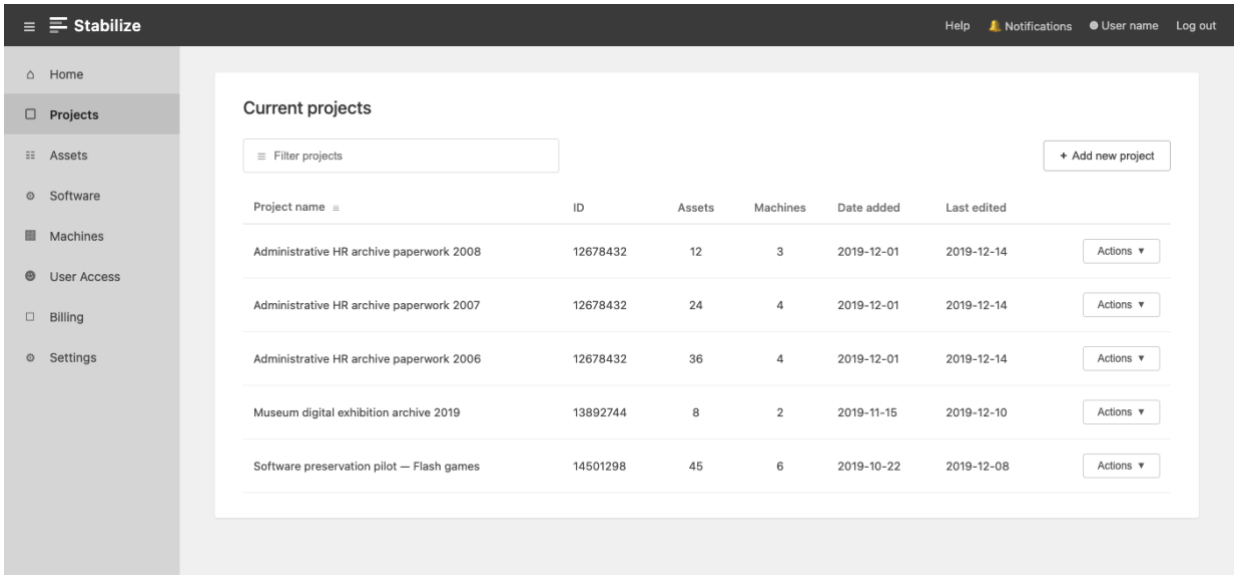


Figure 6. Stabilize UI — Projects overview, based on wireframes by Lozana Rossenova (January 2020).

Table 5 compares how the Demo-UI and Stabilize name the same concepts. The differences show that the Demo-UI's vocabulary is not the only way to organize emulation work.

Table 5: Demo-UI vs. Stabilize — How the Same Concepts Are Named

Demo-UI	Stabilize	What changed
Dashboard	Home	Oriented toward the user's workflow instead of server status.
Environments	Machines	"Machine" refers to a specific computing system.
Software	Software	Same term in both.
Objects	Assets	"Asset" signals that the content has institutional value.
UVI	*(absent)*	Integrated into the Project creation wizard.
Networks	*(within Projects)*	Part of the Project workflow, not a separate tab.
Images	*(absent)*	Disk images managed within Machines.
Administration	Settings	Simplified.
(absent)	**Projects**	Groups machines, assets, and software under one purpose.
(absent)	**User Access**	Dedicated user management.
(absent)	**Billing**	Usage tracking for service-based deployment.

Stabilize introduces a *Project* level above machines and assets, allowing them to be grouped together. In the Demo-UI, environments, objects, and software exist as separate lists.

Stabilize also includes provenance metadata for assets (who created it, when, who imported it) and a Revisions system for machines with Fork/Revert capability. As Table 5 shows, several Demo-UI concepts are reorganized — for example, Networks and Images are not separate tabs but part of other workflows, and the term “Assets” replaces “Objects.” The company entity behind Stabilize has since ceased operations (R. de Vries, personal communication, 2026), though OpenSLX continues to develop the underlying emulation technology.

4.3 Case Study Walkthrough: Three Software Objects

The following tables document the fields visible for each work across the Demo-UI’s tabs. All three software objects are shown side by side.

4.3.1 Environment List

Table 6 shows how the three objects appear in the Environments list. *Bomb Iraq* has two entries with the same name.

Table 6: Environment List Entries

Field	Tod dem Fernsehen	Bomb Iraq (env 1)	Bomb Iraq (env 2)	Every Icon
Name	TMW Tod dem Fernsehen	Bomb Iraq	Bomb Iraq	Every Icon
ID	32c084db-…	292b6b1f-…	cc526972-…	*(UUID)*
Operating System	Apple System 7/8 (M68k)	*(empty)*	*(empty)*	Apple System 7/8 (M68k)
Last modified	Feb 2025	Nov 2023	Nov 2023	Jan 2026

4.3.2 Details Tab

Only *Every Icon* lists configured software — Netscape Navigator Gold 3.04, the browser required to host its Java applet. *Tod dem Fernsehen* and *Bomb Iraq* need no such declaration (Table 7).

Table 7: Details Tab Fields

Field	Tod dem Fernsehen	Bomb Iraq	Every Icon
Name	TMW Tod dem Fernsehen	Bomb Iraq	Every Icon
Operating System	Apple System 7/8 (M68k)	*(empty)*	Apple System 7/8 (M68k)
Handle	— create	Returns a server error (500) when clicked	*(not checked)*
Last change	“CD eingebunden / Bildschirmauflösung auf 640x480 gestellt / Autostart des Werkes”	“imported base environment”	‘Set “every icon” as home page / visual styling’

Field	Tod dem Fernsehen	Bomb Iraq	Every Icon
Language of “Last change”	German	English (system-generated)	English
Configured software	*(empty)*	*(empty)*	Netscape Navigator Gold 3.04
Runtime: WebRTC Audio	Enabled	Enabled	Enabled
Runtime: XPRA Video	Enabled (PNG)	Enabled (PNG)	Enabled (PNG)
Runtime: Requires clean shutdown	Enabled	Enabled	*(not checked)*
Networking	Disabled	Disabled	Enabled
Drive: disk (boot)	Media ID: 64fd14e5-…	Boot disk	Boot disk
Drive: cdrom	User object: TMW Tod dem Fernsehen	*(empty)*	*(empty)*

4.3.3 Advanced Settings Tab

All three use BasiliskII with the version set to “latest.” The resolution and frameskip values differ across all three objects (Table 8).

Table 8: Advanced Settings

Field	Tod dem Fernsehen	Bomb Iraq	Every Icon
Emulator	BasiliskII	BasiliskII	BasiliskII
Emulator version	latest	latest	latest
ROM	rom://83e76594-…	*(UUID)*	*(UUID)*
Resolution	640x480	640x480	800x600
Frameskip	3	6	0

4.3.4 Revisions Tab

The revision histories differ in length, detail, and completeness (Table 9).

Table 9: Revision History Overview

	Tod dem Fernsehen	Bomb Iraq	Every Icon
Number of revisions	3	1 per environment	5
Dates recorded	No	No	Yes
Authors recorded	No	No	No
Available actions	Fork, Revert	Fork	Fork, Revert

Every Icon has the most detailed revision history — five entries with dates. *Tod dem Fernsehen* has three entries without dates (Table 10). *Bomb Iraq* has one system-generated entry per environment with no human documentation. No revision records the author. The Environment list shows an Owner column with a UUID linked to a Demo-UI user account, but individual revisions do not record who made the change.

Table 10: Revision Entries

Object	Version	Description	Date
Tod dem Fernsehen	Current	“CD eingebunden, Bildschirmauflösung auf 640x480 gestellt, Autostart des Werkes”	—
Tod dem Fernsehen	Previous	“screen resolution set to 800x600 pixels”	—
Tod dem Fernsehen	Initial	“imported / user created environment”	—
Bomb Iraq	Current (both envs)	“imported base environment”	—
Every Icon	Current	‘Set “every icon” as home page / visual styling’	Jan 2026
Every Icon	v4	“Installed netscape / Cleaned up HD”	Dec 2025
Every Icon	v3	“screen resolution set to 800x600 pixels”	Oct 2025
Every Icon	v2	“Installed Disk Copy 6.3.3”	Sep 2025
Every Icon	v1	“Installed Aladdin Stuffit Expander 5.5”	Aug 2025

4.4 Contribution 1: Where Conservation Information Can Be Entered

If the vocabulary gap cannot be fixed, it can at least be mapped. Table 6 to Table 10 document what the Demo-UI records. Table 11 lists information that the conservator has but for which the interface has no dedicated field. The last column identifies which existing free-text field could hold each type of information.

Table 11: Information Without a Dedicated Field

The last column uses colored labels to indicate which existing Demo-UI field each piece of information could be entered into. The legend texts at the bottom of the table explain each label.

Information	Tod dem Fernsehen	Bomb Iraq	Every Icon	Where it could be entered
Why this resolution	640x480 matches CD-ROM specs; 800x600 was tested and distorted the layout	Not documented	800x600 provides space for browser chrome + applet	[LC] [RD] [EC]
Why this frameskip	Default value	Selected to match original hardware speed	0 minimizes rendering delay for continuous animation	[LC] [RD] [EC]
Condition	Partially functional — VRML works, QuickTime codecs missing	Functional — HyperCard runs, system ambience intact	Functional — applet runs, behavioral fidelity not formally verified	[LC] [RD]
Risk	QuickTime codec unavailability; emulator version not pinned	System ambience could be lost if environment is modified	Java runtime version changes could alter execution speed	[LC] [RD]
Link to catalog	No ArtBase entry. TMW has a separate EaaS instance (eaas.demo.tmw.at)	ArtBase Q4893	ArtBase Q2428	[NM] [LC]
Which environment is current	One environment	Two identically named entries	One environment	[NM] [LC] [EC]
Revision author	Not recorded	Not recorded	Not recorded	[LC] [RD] [EC]

Note. ***Legend*** *— Existing Demo-UI fields where this information could be entered:*

- [NM] **Name** (pink) — Free-text field. Can encode institutional context, version indicators, or catalog references as part of the environment title.
- [LC] **Last change** (yellow) — Free-text field in the Details tab. Can hold any note about the environment, including rationale, condition, or author.
- [RD] **Revision description** (gray) — Free-text note attached to each revision entry. Can document what was changed and why.
- [EC] **Emulator configuration** (blue) — Free-text field in Advanced Settings. Can include comments alongside technical parameters.

All four fields accept free text. None of them prompt the user to enter conservation-related information, and there is no structure to ensure it is recorded consistently — but the fields are available.

4.5 Contribution 2: Suggested Clarifications for the Interface

Several terms and controls in the Demo-UI are unclear without prior technical knowledge. Table 12 lists these elements alongside a concrete suggestion for what could be added to the interface — such as writing out an abbreviation, adding a tooltip, or fixing a broken feature.

Table 12: Suggested Clarifications for the Demo-UI

The last column uses colored labels to indicate the type of change suggested. The legend texts at the bottom of the table explain each label.

Element	Where	What it currently says	Suggested change
UVI	Sidebar tab	"UVI"	[WO] Write out the full name: "UVI (Universal Virtual Interactor)"
Handle	Details tab	"— create"	[BF] Intended to create a permanent URL to the environment (using Handle.net). On the Rhizome instance, clicking this returns a server error (500). Fix or hide until functional.
Checkpoint	Session menu	"Checkpoint"	[TT] "Saves a snapshot of the running environment. Next time you start it, it resumes from this point instead of starting fresh."
Frameskip	Emulator configuration	e.g. "frameskip 3"	[TT] "Controls how many display frames the emulator skips. Lower values = smoother animation, higher values = less server load."
Fork	Revisions tab	"Fork" button	[TT] "Creates a copy of the environment at this point. Use this to test changes without losing the current version."
Emulator	Advanced Settings	e.g. "BasiliskII"	[TT] Add information about which emulator is designed for which platform — e.g. BasiliskII for Macintosh 68k, QEMU for x86.

Note. ***Legend*** *— Types of suggested change:*

- [WO] **Write out** (green) — Expand an abbreviation or acronym so the user can understand the term without looking it up.
- [TT] **Tooltip / help text** (blue) — Add a short explanation visible on hover or next to the element, so the user understands what it does.
- [BF] **Bug fix** (beige) — The feature is broken or produces an error. It should be fixed or hidden until functional.

Although the three objects differ markedly — in operating system, configuration, the language of their documentation, and the depth of their revision histories — the findings converge on the same point. In all three, the interface records the technical setup in full but offers no dedicated field for the conservation information in Table 11 (rationale, condition, risk, catalog links, authorship), so the same four free-text fields remain the only places to record it; and the clarifications proposed in Table 12 apply regardless of

which object is open. The gaps are therefore structural to the interface, not peculiar to any one work. Their implications are discussed in Chapter 5.

5 Discussion

This discussion draws the findings together around the conservator. Conservation is trans-technological (Section 2.7): the medium changes, but the ethos — to understand a work, judge what is essential to it, and care for it over time — does not. Faced with an emulated work, the conservator does what conservators have always done, deciding what the work actually is and separating what is essential to its identity from what belongs only to a single showing. The medium only sharpens the question: because each run is a slightly new actualization, even where a recording of an earlier showing offers a reference, the conservator must still judge which differences leave the work intact and which begin to change it. And as creative practice across disciplines — painting, design, and beyond — increasingly moves through software, emulation will become a conservation problem for fields that have never needed it, and the judgement examined here will only grow more central.

5.1 The Conservator and the Tool

Section 2.5 argued that the conservator is a decision maker — someone trained to determine what has value, what is at risk, and how to document preservation choices. When that conservator opens the Demo-UI, they bring this knowledge with them. But as the walkthrough in Section 4.3 showed, the interface has no place for most of it.

The case studies in the walkthrough differ widely — a German-language VHS and CD-ROM work, a found HyperCard environment, and a browser-hosted Java applet — yet the same conservation information goes unrecorded in each (Table 13).

Table 13: The same gaps across three different works

Finding	Evidence	Consequence for the conservator
Two environments share the name “Bomb Iraq,” with nothing to mark which is current	Table 6	The authoritative version cannot be identified — a question of the work's identity
No revision records its author; only *Every Icon*'s revisions are dated	Table 9 – Table 10	Decisions carry no provenance — who changed what, and when, is unrecoverable
“Last change” holds a German rationale note for one work and an auto-generated string for another	Table 7	The one field used for reasoning is mislabeled and applied inconsistently
Configured software is listed only for *Every Icon*	Table 7	Format and obsolescence risk is invisible for the works that don't declare it
Resolution and frameskip differ across all three, with no rationale recorded	Table 8	Technical and value decisions are made but never justified

The pattern is more telling than any single entry. The three works differ in language, complexity, and configuration, but the categories the interface cannot hold are identical across them — identity, provenance, condition, risk, rationale. The decisions behind those categories are still being made: someone set the resolution, judged a work functional, chose which of the two *Bomb Iraq* environments to keep. The interface

simply does not treat such judgments as information worth holding, so they surface only as technical residue — a value in a frameskip field, a note in the wrong box — or disappear. That the same categories fall away regardless of the object points back to the vocabulary rather than the works: the Demo-UI was built to record how an environment runs, and a conservator's reasoning about why it should run that way has no dedicated place in that scheme.

The Demo-UI is, however, already functioning as a shared professional space. Almut Schilling, a freelance conservator working with software objects across several Viennese museums, describes the Demo-UI as a meta layer where conservators and curators can speak the same language (A. Schilling, personal communication, April 15, 2026). In a field where no dedicated conservation interface for emulation-based preservation exists, the Demo-UI has become the meeting point by default — the one tool that both professions use when working with emulated environments. This makes its vocabulary not just a technical concern but a professional communication issue: the terms visible in the interface shape how conservators and curators discuss, document, and make decisions about the objects they share responsibility for.

Even though Schilling sees the Demo-UI as a shared space where metadata about the emulation of case studies is stored and communicated across professions, the language available in this space is entirely technical — which risks losing dimensions of the preservation work that are not technical. The Demo-UI records the technical setup of each environment — emulator, resolution, frameskip, drives, software packages. But the conservator also knows *why* that resolution was chosen, *what condition* the object is in, and *how* this environment relates to the object’s broader history. Chapter 4 identified four free-text fields (Name, Last change, Revision description, Emulator configuration) where this information can be entered, and six interface elements that need clarification — whether through expanded abbreviations, tooltips, or bug fixes. These are the two contributions of this study.

The values debates discussed in Section 2.6 — Ruskin’s insistence on authenticity, Viollet-le-Duc’s ideal restoration, Riegl’s framework of age value and use value — are not abstract for a conservator working in the Demo-UI. When choosing between 640x480 and 800x600 for *Tod dem Fernsehen*, the conservator is making a value judgment: faithfulness to the original specification versus a more usable display. When Espenschied preserved the system ambience of the Macintosh TV containing *Bomb Iraq*, he was recognizing age value — the patina of an anonymous life — in a digital object. These decisions happen in the Demo-UI, but the Demo-UI has no vocabulary for them.

5.2 The Rhizome and Difference in Repetition

A rhizome (Figure 7) is a horizontal underground stem — white clover, couch grass, bamboo — that sends out roots and shoots from its nodes without hierarchy or central trunk. Deleuze and Guattari (1987, pp. 6–13) adopted this botanical structure as a philosophical concept. The digital arts organization Rhizome takes its name from it.

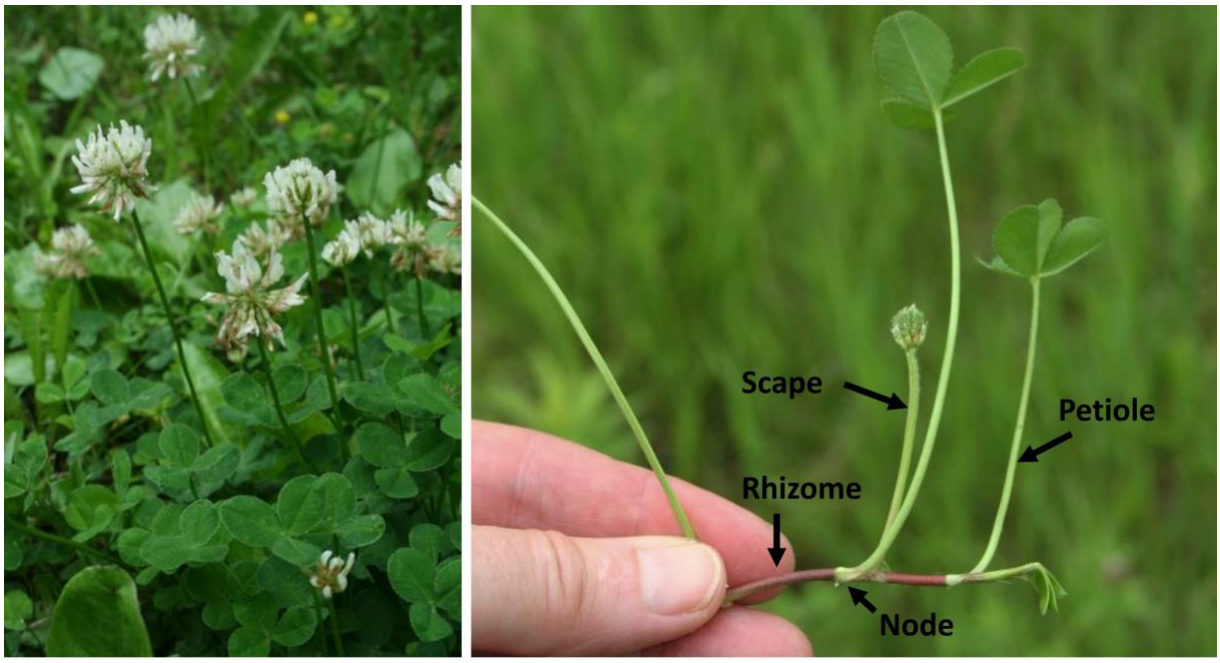


Figure 7. A white clover colony showing the rhizome — a horizontal stem connecting nodes without hierarchy. Source: blogger.googleusercontent.com. Used with permission.

Emulation-as-a-Service operates rhizomatically: old operating systems, old software, old browsers — heterogeneous components from different decades — are connected and rendered inside modern web browsers. The emulated environment is an *assemblage* of elements that operates as a functional whole (Deleuze & Guattari, 1987, pp. 88–89). A digital museum object may be a single piece of software — a Java applet, a HyperCard stack — but it can only be experienced through the assembled whole: the emulator, the operating system, the browser, the rendering pipeline.

In *Difference and Repetition* (1994), Deleuze (Figure 8) argues that every repetition produces difference rather than identity (Deleuze, 1994, pp. 76–79). Each time a conservator clicks “Run” in the Demo-UI, the emulator does not resurrect the original machine — it produces a new actualization. The EaaS rendering pipeline transmits the emulator’s output through Apache Guacamole as a stream of PNG images. The user never encounters the original display signal. Each frame is a repetition that produces difference.

For conservation, this means that emulation is not recovery of an original state but production of something new each time. The free-text fields identified in Chapter 4— particularly the revision descriptions — are where these differences can be noted. Each revision is a new actualization, and the conservator who documents it is acknowledging what Deleuze’s philosophy insists upon: the emulated experience is not identical to the original. This is sharpest when the stack itself changes: an Apache Guacamole upgrade, a new browser codec, a host-OS update — none of which leave a trace in the Demo-UI's revision descriptions — still alter what the viewer sees. The difference Deleuze insists on compounds with every infrastructure choice, not just with every “Run”.

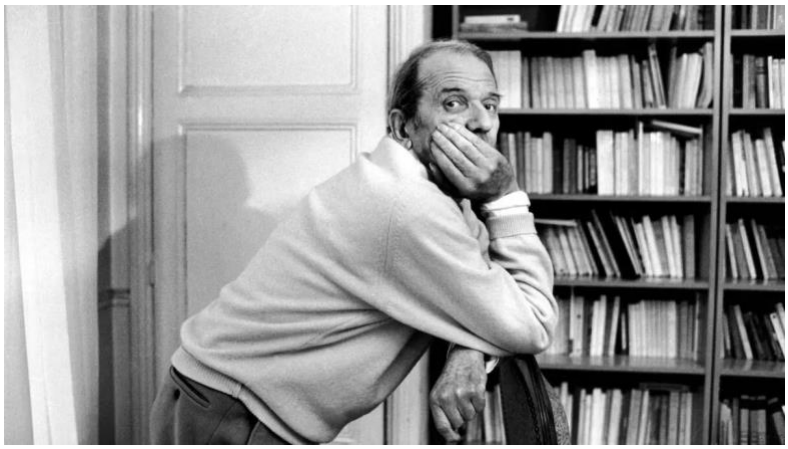


Figure 8. Gilles Deleuze (1925–1995). Courtesy of Vive la Culture.

5.3 Media Genealogy and the Demo-UI's Origin

Figure 9. Clemens Apprich. Photo: Maria Ziegelböck / Angewandte.

Apprich (Figure 9; 2017) develops the concept of *media genealogy*: digital tools carry within them the design decisions, power structures, and cultural assumptions of the

communities that built them (Apprich, 2017, pp. 5–7). The institutional history traced in Section 2.4 is a concrete example: the Demo-UI's vocabulary reflects a chain of decisions made by computer scientists and software companies, not by conservators.

Apprich's account of 1990s net cultures (Apprich, 2017, pp. 33–48) connects directly to the case studies in this thesis. *Tod dem Fernsehen* emerged from the Viennese media art scene — Public Netbase, critical net culture. *Bomb Iraq* preserves a found artifact from an earlier computing culture. Both works are entangled with their technical infrastructure. Preserving them through emulation means preserving specific moments in cultural history — but the Demo-UI makes no distinction between emulating a net art work and any other piece of legacy software.

Apprich introduces technotopia — drawing on Michel Foucault's heterotopia — to describe digital spaces where different communities and their discourses coexist (Apprich, 2017, p. 3). The EaaS Demo-UI is such a space: the systems-engineering community that built it and the conservation community that uses it share an interface without sharing a vocabulary. What looks like a utopian achievement from one side — an interface that boots any obsolete environment on demand, at scale, for any institution — can look dystopian from the other, where the same interface offers no field for condition, no field for attribution, and no place to record the reasoning behind a preservation decision. The contributions identified in Chapter 4 — using the existing free-text fields deliberately, and clarifying the interface's own terminology — are small steps toward making this meeting point more legible to the conservator.

5.4 Implications for Practice

The two contributions of this study are practical. Contribution 1 (Table 11) shows that conservation information — rationale, condition, risk, catalog links, authorship — can be entered in existing free-text fields today, without any changes to the software. This requires no development work, only awareness. Contribution 2 (Table 12) identifies six interface elements where a short explanation — a tooltip, a written-out abbreviation, a bug fix — would make the tool more accessible.

As Molander (1993) warns, without theory, conservation becomes merely applied science. The Demo-UI works as a technical tool. The question is whether it can also work as a conservation tool — and this thesis argues that it already can, if the conservator knows where to enter the information and what the interface’s own terms mean.

5.5 What the Interface Does Not Show

Some conservation-relevant information cannot be entered at all — not even as a workaround.

In the preservation workflows carried out by practitioners such as Maltar (2020) at Rhizome and Schilling (personal communication, April 15, 2026) across several Viennese museums, file format identification is a standard step — using tools such as Siegfried (Lehane, n.d.), which matches files against the PRONOM file format registry (National Archives, n.d.) and returns structured metadata: format name, version, and PUID. This information feeds into each institution's conservation or archiving

documentation. The ArtBase stores it under “made of” (P81). But the Demo-UI does not expose it — objects are listed with a name and an ID, and the conservator cannot see what formats they contain.

Similarly, packaging standards like BagIt (Boyko et al., 2012) — which provide checksums and manifests to verify the integrity of imported digital content — leave no trace in the interface. The conservator cannot see whether an object arrived intact or how it was packaged.

These absences bear directly on the conservator's work, because both answer questions conservation has always asked, here of a digital material. A file's format tells a conservator whether it risks becoming unreadable and may need migrating to a newer one — the digital form of knowing what an object is made of. Because the Demo-UI does not show formats, the conservator cannot judge that risk while working in the interface, and the materials knowledge conservation depends on (Section 2.7) has to come from elsewhere. Integrity data answers a different conservation question: whether the object is authentic and unaltered. For an object of any material, the conservator must be able to establish what is original and what has changed; for software, checksums and manifests are how that is established, and the Demo-UI shows none of it. In both cases the conservator is left to decide on a partial picture, or to work across disconnected systems — holding in another window the very facts the decision in the Demo-UI depends on. These are not gaps that free-text fields can bridge: they are information practitioners already produce and document elsewhere, but which stays invisible in the tool where preservation decisions are actually made.

5.6 Limitations: The Case Against Workarounds

A software developer reviewing the contributions of this thesis would raise a legitimate objection: free-text fields are not conservation fields. Writing condition notes into “Last change” or encoding catalog references into environment names does not make the Demo-UI a conservation tool — it makes it a conservation tool being used incorrectly.

From a software engineering perspective, the objection is well-founded. Free-text fields have no validation, no structure, and no consistency. Nothing prevents the next user from overwriting a condition note with a technical remark. Nothing ensures that revision descriptions follow a format. There is no search, no filtering, no reporting based on conservation-specific data — because the data is buried in fields designed for other purposes. A system built for conservation documentation would have dedicated fields for condition, rationale, risk assessment, and attribution — fields with defined types, controlled vocabularies, and clear separation from the technical configuration. The approach proposed in Table 11 offers none of this.

This criticism points to a real weakness in the study's first contribution. Identifying where conservation information can be entered is not the same as showing that it should be entered there. A purpose-built conservation layer — whether integrated into the Demo-UI or connected to it — would be more reliable, more searchable, and more sustainable than notes scattered across general-purpose text fields. Rossenova's (2021) research on the ArtBase redesign demonstrates what such structured approaches look like in practice: distinct data models for an artist’s works, variants, and artifacts, each with fields designed for the information they need to hold.

The second contribution — suggested clarifications such as tooltips and expanded abbreviations (Table 12) — is less vulnerable to this critique, since it proposes changes to the interface itself rather than workarounds within it.

The thesis does not claim that free-text workarounds are a permanent solution. It claims that they are available today, in an interface that is unlikely to get a new vocabulary in the near term, and that using them deliberately is better than not documenting conservation reasoning at all. The gap between what is possible now and what should be built remains open — and closing it requires collaboration between conservators and software developers, not one profession working around the other's tool.

6 Conclusion

This study asked how the technical concepts in the EaaS Demo-UI relate to conservation practice, how they can be used to better document the preservation of software objects, and what a conservation-oriented vocabulary for EaaS would look like. At its core, this is a question about the conservator. Section 2.5 described conservation as a practice of decision-making — judging what has value, what is at risk, and recording the reasoning behind each preservation choice so that it can be understood and revisited later. For software objects, much of that work now happens in the Demo-UI, which is why its vocabulary is not a side issue: the terms an interface makes available shape what a conservator can record, and so what survives of their judgment. The findings below are technical in their detail, but the question beneath them is a conservation one — whether the conservator's reasoning has anywhere to live.

How do the Demo-UI's concepts relate to conservation practice?

- The Demo-UI records what a systems administrator needs — emulator type, operating system, resolution, drives, software packages — but provides no dedicated fields for condition, rationale, or attribution.
- The field labeled “Last change” is a misnomer inherited from the EMiL project; it is stored as the environment's description (R. de Vries, personal communication, 2026). Even the fields that exist are not what they appear to be.
- The case studies showed that the same interface fields behave differently across *Tod dem Fernsehen, Bomb Iraq,* and *Every Icon*, while the same categories of conservation information are missing in all three.

How can they be used to better document preservation?

- Four existing free-text fields — Name, Last change, Revision description, and Emulator configuration — can accommodate conservation information today without changes to the software (Table 11).
- Six interface elements require clarification through expanded abbreviations, tooltips, or bug fixes (Table 12).

What would a conservation-oriented vocabulary look like?

- Alternative structures already exist. The ArtBase uses a three-level model (Artwork, Variant, Artifact) where the Demo-UI uses only “Object.” Stabilize introduces a Project concept that groups related resources. The Demo-UI's vocabulary is one design choice among several.
- Fields such as a preservation rationale, a condition assessment, or richer revision metadata would give conservators a place to record why decisions were made, not just what was configured — the kind of conservation layer Section 6.1 sketches in more detail.

Why does this matter?

Software environments degrade. Emulators lose maintainers. File formats become unreadable. When an emulated artwork stops working, what remains in the current Demo-UI is a name, a UUID, and perhaps a few undated, unsigned notes. There is no record of why this configuration was chosen, what was tried and rejected, or what the

artwork looked like when it last ran. The question is not whether artworks will become inaccessible, but whether enough has been recorded to bring them back. The changes proposed in this thesis require only minor interface updates, but they ensure the next conservator inherits reasoning, not just a broken configuration.

6.1 Outlook

For conservators working in the field of software conservation, this thesis can serve as a foundation for presentations at universities and conferences — showing what the role of a software conservator looks like in practice, what tools are available, and why conservation competence matters in their development. The gap between conservation knowledge and technical infrastructure is not widely understood outside the field; making it visible is itself a contribution.

The theoretical research undertaken for this thesis — drawing on Deleuze, Apprich, Ruskin, Riegl, and Rossenova — opens several directions for longer essays and further studies. The relationship between Deleuze's concept of difference in repetition and the practice of emulation deserves deeper treatment than a bachelor thesis allows. Apprich's media genealogy could be applied more broadly to other preservation tools and institutional infrastructures. A comparison between the 19th-century conservation values debate (Ruskin, Riegl, Viollet-le-Duc) — where Riegl (1903) introduced the idea that monuments carry multiple, potentially conflicting values such as age value, historical value, and use value — and Muñoz Viñas's (2005) contemporary theory of conservation, which develops this further by arguing that preservation decisions should be negotiated among stakeholders rather than determined by the conservator alone, would test which framework better fits the practice of conserving software objects.

As discussed in Section 2.8, the digital preservation community has already developed structured standards for the kind of information this thesis identifies as missing from the Demo-UI. PREMIS defines Events and Agents — a standard way to document what was done, by whom, and why. Rhizome's ArtBase already implements many of these standards through its Wikibase model, holding rich conservation metadata for the works in its collection. The gap is that this metadata stays in the catalog and never reaches the Demo-UI, where the conservator actually works. A future interface — whether a redesign of the Demo-UI or a conservation layer built alongside it — could surface PREMIS Events as part of the revision history, link Agents to individual preservation actions, and connect each emulated environment to the ArtBase record of the work it preserves. One central question is how this metadata could be transferred from the catalog into the EaaS backend and made visible to the conservator working in the Demo-UI, without introducing redundancy or allowing the two records to drift apart.

As a lighter-weight alternative, the free-text fields already available in the Demo-UI could adopt conventions for structured metadata — similar to YAML frontmatter in Markdown or trailers in Git commit messages — embedding key-value pairs such as “Condition:”, “Rationale:”, or “Author:” within the existing description and revision fields. This would require no backend changes and could be implemented immediately as a conservation practice convention. As discussed in Section 4.1, the field labeled “Last change” is actually stored as the environment's description — a naming choice

inherited from the EMiL project (R. de Vries, personal communication, 2026). This means the one free-text field available to conservators already carries a misleading label, making structured conventions all the more important for giving that field a clear and consistent purpose.

Finally, this thesis analyzes the interface from the perspective of a single conservator with a dual background in data science and conservation. A follow-up study using think-aloud protocols with conservators of different technical backgrounds would test whether the ambiguities identified here are experienced the same way in practice, and would strengthen the evidence for the clarifications proposed in Section 4.5.

6.1.1 Development of the Demo-UI

On the technical side, the Demo-UI's AngularJS 1.x framework is end-of-life and will eventually need to be replaced. Among modern alternatives, *SvelteKit* — a compiler that produces vanilla JavaScript with no runtime dependency — aligns with the Minimal Computing principles advocated in digital humanities (Gil & Ortega, 2016). A Svelte-built interface ships as static files that can run on any web server without runtime dependencies (Svelte, n.d.). *React/Next.js* offers the largest ecosystem; *Vue 3/Nuxt* has adoption in European academic computing; *Angular* (v17+) offers the most direct migration path. The choice of framework for a digital preservation tool raises a broader question: how can tools designed for long-term access be built on technologies with short life cycles?

The author's continuing professional work with the EaaS ecosystem provides an opportunity to test the findings of this thesis in practice — working with the Demo-UI daily, communicating with the development and preservation teams, and observing how conservators actually use the interface. The question of how preservation tools can better serve conservation practice is not one that a single thesis can answer. But it is one that a conservator with both technical and conservation training is well positioned to keep asking.

7 References

Altendorf, A. (1994). R.A.M.S. — Tod dem Fernsehen: CD-ROM / Video 1994. https://altendorf.radiofabrik.at/1994/11/09/r-a-m-s-tod-dem-fernsehen-cdrom-video-1994/

Appelbaum, B. (2007). *Conservation treatment methodology*. Butterworth-Heinemann.

Apprich, C. (2017). Technotopia: A media genealogy of net cultures. Rowman & Littlefield International.

Arcangel, C. (2005). Bomb Iraq [HyperCard readymade]. http://www.coryarcangel.com/things-i-made/2005-020-bomb-iraq

Boyko, A., Kunze, J., Littman, J., Madden, L., & Vargas, B. (2012). The BagIt file packaging format (V0.97). Library of Congress. https://www.digitalpreservation.gov/documents/bagitspec.pdf

Brown, A. [@realAdrianBrown]. (2019, February 19). *It was indeed contrived — as I recall it was derived from 'Nom', an obsolete Vietnamese writing system known only from historical records, with the 'PRO' prefix which was applied to most of our IT systems at the time* [Tweet]. X (formerly Twitter). https://x.com/realadrianbrown/status/1097960463483187200

Caple, C. (2000). *Conservation skills: Judgement, method and decision making*. Routledge.

Cather, S. (2006). Trans-technological methodology. In D. Saunders, J. H. Townsend, & S. Woodcock (Eds.), The object in context: Crossing conservation boundaries (pp. 89–93). International Institute for Conservation of Historic and Artistic Works.

Civic Creative Base Tokyo. (2023). Playing the Time Stratum series back(wards). https://ccbt.rekibun.or.jp/en/what-we-do/time-stratum-series

Cochrane, E., Rechert, K., Mueller, J., Anderson, S. R., Godfrey, J. M., & Gates, E. (2019). Towards a Universal Virtual Interactor (UVI) for digital objects. In Proceedings of the 16th International Conference on Digital Preservation (iPRES 2019). https://ipres2019.org/static/pdf/iPres2019_paper_128.pdf

Deleuze, G. (1994). Difference and repetition (P. Patton, Trans.). Columbia University Press. (Original work published 1968).

Deleuze, G., & Guattari, F. (1987). A thousand plateaus: Capitalism and schizophrenia (B. Massumi, Trans.). University of Minnesota Press. (Original work published 1980).

Digital Preservation Coalition. (2014). Digital Preservation Awards 2014. https://www.dpconline.org/events/digital-preservation-awards/digital-preservation-awards-2014

EaaSI. (n.d.). EAASI current roadmap and idea backlog [GitHub project]. https://github.com/orgs/eaasi/projects/5/views/6

Gil, A., & Ortega, E. (2016). Global outlooks in digital humanities: Multilingual practices and minimal computing. In C. Crompton, R. J. Lane, & R. Siemens (Eds.), Doing digital humanities: Practice, training, research (pp. 22–34). Routledge.

Harrer, A. (2017). The legacy of Alois Riegl: Material authenticity of the monument in the digital age. Built Heritage, 1(2), 29–41.

ICOM-CC. (1984). The conservator-restorer: A definition of the profession. International Council of Museums — Committee for Conservation. *https://www.icom-cc.org/47/about/definition-of-profession-1984/*

ISO. (2025). *Space data and information transfer systems — Open archival information system (OAIS) — Reference model* (ISO 14721:2025). International Organization for Standardization. *https://www.iso.org/standard/87471.html*

ISO. (2017). *Information and documentation — The Dublin Core metadata element set* (ISO 15836-1:2017). International Organization for Standardization. *https://www.iso.org/standard/71339.html*

Laurenson, P. (2006). Authenticity, change and loss in the conservation of time-based media installations. *Tate Papers*, (6).

Lehane, R. (n.d.). Siegfried — signature-based file format identification [Software]. https://github.com/richardlehane/siegfried

Library of Congress. (2015). PREMIS data dictionary for preservation metadata (version 3.0). *https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf*

Library of Congress. (n.d.). Metadata Encoding and Transmission Standard (METS). https://www.loc.gov/standards/mets/

Maltar, N. (2020). The ArtBase migration — from archive.rhizome.org to a new host [Internship report, State Academy of Fine Arts in Stuttgart, Conservation of New Media and Digital Information]. Mentor: Dragan Espenschied, Rhizome.

Mellon Foundation. (2017, December 7). Software preservation infrastructure [Grant to Yale University]. https://www.mellon.org/grant-details/software-preservation-infrastructure-20444126

Mellon Foundation. (2020, June 4). EaaSI (Emulation-as-a-Service Infrastructure): Phase II [Grant to Yale University]. https://www.mellon.org/grant-details/eaasi-(emulation-as-a-service-infrastructure):-phase-ii-20446893

Mellon Foundation. (2022, June 30). EaaSI (Emulation-as-a-Service Infrastructure): Phase III [Grant to Yale University]. https://www.mellon.org/grant-details/eaasi-(emulation-as-a-service-infrastructure):-phase-iii-20451330

Meyerson, J., Vowell, Z., Hagenmaier, W., Leventhal, A., Rios, F., Russey Roke, E., & Walsh, T. (2017). The Software Preservation Network (SPN): A community effort to ensure long term access to digital cultural heritage. D-Lib Magazine, 23(5/6). *https://www.dlib.org/dlib/may17/meyerson/05meyerson.html*

Molander, B. (1993). Kunskap i handling (2nd ed.). Daidalos.

Muñoz Viñas, S. (2005). Contemporary theory of conservation. Butterworth-Heinemann.

National Archives, The. (n.d.). PRONOM — technical registry. https://www.nationalarchives.gov.uk/pronom/

OpenSLX. (n.d.). About us. *https://openslx.com/posts/About_Us.html*

Rechert, K., Falcão, P., & Ensom, T. (2016). *Introduction to an emulation-based preservation strategy for software-based artworks*. Tate. https://www.tate.org.uk/about-us/projects/pericles/emulation-based-preservation-strategy-for-software-based-artworks

Rechert, K., von Suchodoletz, D., & Welte, R. (2010). Emulation based services in digital preservation. *Proceedings of the 10th Annual Joint Conference on Digital Libraries (JCDL '10).* ACM. *https://doi.org/10.1145/1816123.1816182*

Rechert, K., Valizada, I., von Suchodoletz, D., & Latocha, J. (2012). bwFLA — A functional approach to digital preservation. PIK — Praxis der Informationsverarbeitung und Kommunikation, 35(4), 259–267. *https://doi.org/10.1515/pik-2012-0044*

Reeder, F. (2024, December 20). Samla på något udda: om Felixsnurran av Beck & Jung. https://reeder.se/samla-pa-nagot-udda/

Rhizome. (2014, June 24). Emulating Bomb Iraq. https://sites.rhizome.org/emulating-bomb-iraq-arcangel/

Rhizome ArtBase. (n.d.-a). Browse works by date of inception. https://artbase.rhizome.org/wiki/Browse/by_year

Rhizome ArtBase. (n.d.-b). Every Icon (Q2428). *https://artbase.rhizome.org/wiki/Q2428*

Riegl, A. (1903). Der moderne Denkmalkultus: Sein Wesen und seine Entstehung. W. Braumüller. [English translation: “The Modern Cult of Monuments: Its Character and Its Origin,” in Oppositions, 25, 1982, pp. 21–51.]

Rinehart, R., & Ippolito, J. (2014). *Re-collection: Art, new media, and social memory*. MIT Press.

Rossenova, L. (2020). Stabilize UI wireframes [Unpublished interface design]. The author's mockup reconstruction based on these wireframes is presented as Figure 2; the comparative terminology analysis appears in Section 4.2.

Rossenova, L. (2021). Model–Database–Interface: A study of the redesign of the ArtBase, and the role of user agency in born-digital archives [Doctoral thesis, London South Bank University]. LSBU Research Repository.

Rothenberg, J. (1999). *Avoiding technological quicksand: Finding a viable technical foundation for digital preservation*. Council on Library and Information Resources. *https://www.clir.org/pubs/reports/rothenberg/*

Ruskin, J. (1849). The Seven Lamps of Architecture. Smith, Elder, & Co.

Schilling, A. (2026, April 15). Personal communication [Interview]. Fine Arts Academy, Vienna.

Russler, N. (2015, October 27). Initial import [Source code commit]. GitLab. *https://gitlab.com/emulation-as-a-service/demo-ui/-/commit/6ec84777ee2fef1fd9af98c291eea50f9b1c4eb0*

Simon, J. F., Jr. (1997). Every Icon [Java applet]. http://www.numeral.com/appletsoftware/eicon.html

Svelte. (n.d.). Introduction. *https://svelte.dev/docs/introduction*

Software Preservation Network. (n.d.). Metadata. *https://www.softwarepreservationnetwork.org/core-activities/metadata/*

Software Preservation Network. (2026, February 25). Thoughtful sunset for the Software Preservation Network. https://www.softwarepreservationnetwork.org/thoughtful-sunset-for-the-software-preservation-network/

Viollet-le-Duc, E.-E. (1854–1868). Dictionnaire raisonné de l'architecture française du XIe au XVIe siècle (10 vols.). B. Bance / A. Morel.

von Suchodoletz, D. (2009). *Funktionale Langzeitarchivierung digitaler Objekte: Erfolgsbedingungen des Einsatzes von Emulationsstrategien* [Functional long-term archiving of digital objects: Conditions for the successful use of emulation strategies]. Cuvillier Verlag.

von Suchodoletz, D. (2012, December). The foundations of emulation as a service: An interview with Dirk von Suchodoletz (Parts 1–2) [Interview by T. Owens]. The Signal, Library of Congress. https://blogs.loc.gov/thesignal/2012/12/the-foundations-of-emulation-as-a-service-an-interview-with-dirk-von-suchodoletz-part-one/ and https://blogs.loc.gov/thesignal/2012/12/the-foundations-of-emulation-as-a-service-an-interview-with-dirk-von-suchodoletz-part-two/

von Suchodoletz, D., Rechert, K., & Valizada, I. (2013). Towards emulation-as-a-service: Cloud services for versatile digital object access. International Journal of Digital Curation, 8(1), 131–142. *https://doi.org/10.2218/ijdc.v8i1.250*

Yale University Library. (2014, February). New emulation-as-a-service technology opens new opportunities for libraries. *https://web.library.yale.edu/news/2014/02/new-emulation-service-technology-opens-new-opportunities*

Yale University Library. (2020). Foundations grant $1.5 million to take software emulation to the next level. *https://web.library.yale.edu/foundations-grant-15-million-take-software-emulation-next-level*

8 List of Tables and Figures

8.1 Tables

8.2 Figures

8.3 Image Sources

Figure	Description	Source
Figure 1	Tod dem Fernsehen — screenshots	Author’s screenshots from the emulated environment
Figure 2	Bomb Iraq — screenshots	Author’s screenshots from the emulated environment
Figure 3	Every Icon — screenshot	Author’s screenshot from the emulated environment
Figure 4	Demo-UI mockup — overview	Author’s reconstruction based on the Rhizome instance
Figure 5	Demo-UI mockup — detail view	Author’s reconstruction based on the Rhizome instance
Figure 6	Stabilize UI — Projects overview	Author’s reconstruction based on wireframes by Lozana Rossenova (2020)
Figure 7	White clover colony and rhizome	blogger.googleusercontent.com, used with permission
Figure 8	Gilles Deleuze (1925–1995)	Courtesy of Vive la Culture (cdn.sanity.io)
Figure 9	Clemens Apprich	Photo: Maria Ziegelböck / Angewandte