Fencing Apparently Infinite Objects

Defining productive object boundaries for performative digital objects

Dragan Espenschied
Rhizome
New York, USA
dragan.espenschied@rhizome.org

Klaus Rechert
University of Freiburg
Freiburg, Germany
klaus.rechert@rz.uni-freiburg.de

ABSTRACT

Todays digital preservation practice focuses mostly on fixed or complete objects, which are defined by their manifestation as files or records with the assumption that if one has the file(s) at hand, the preservation or curatorial effort can be focused on these materials only. With increasing importance of networked objects or software, object boundaries appear increasingly "blurry": for instance, many software applications are staged to look and behave like locally running binaries, when in fact an orchestration of networked processes is required for their operation. To cope with such apparently infinite objects and their increasing complexity, this paper explores an expanded definition of object boundaries for performative objects.

1 INTRODUCTION

The digital preservation discipline has traditionally worked with object definitions in which object boundaries are mostly based on established metaphors like "files" and "records," which seem largely self-explanatory, and help with establishing systems for measuring completeness and preservation success.

As performative digital objects—"software" in the broadest sense rather than digital simulacra of documents—are sought to be preserved, object boundaries appear increasingly "blurry": for instance, many software applications are staged to look and behave like locally running binaries, when in fact an orchestration of networked processes is required for their operation. This ranges from simple software installers checking for license servers to desktop applications presenting networked resources such as maps, videos, or communication messages exclusively or mixed with local artifacts.

Furthermore, creating such performative objects has become increasingly accessible to general computer users, for example via popular frameworks like React Native, Electron, or Jupyter Notebooks,[7] and is already an established practice in many areas like digital humanities, net art, and computer games.

It appears that the good old "desktop application" is becoming a metaphor similar to "files" and "records": it quickly approaches the limits of its usefulness and is creating conceptual limitations to preservation practice.

Hence, this paper explores an expanded definition of object boundaries for performative objects.

2 BOUND, BLURRY, AND BOUNDLESS OBJECTS

The demarcation of a digital object is usually done in an "at rest" state, during storage, i.e. a static representation of data while no computing activity is happening, and is therefore shaped by units of storage alone, such as files and storage media. As part of this practice, the term "object" is usually used synonymously with "file" or a collection thereof. Even the simplest file, however, is dependent on technical performance to transform it from a static bitstream-preserved artifact into a an active object that is fit for human consumption or interaction. [3]

For that reason, we propose to examine such objects in a "switched-on" state and to identify components based on their effect on the *performative potentials* of the object.

When it comes to software in the form of executables, it can be assumed that any binary that is fully locally available can be executed or *performed* via hardware emulation, re-creating the the same potentials as if the binary was made to run on actual hardware.

Software preservation frameworks like EaaS [8] have demonstrated that productive abstraction layers can be drawn by gradually separating "common" or "mass-produced" artifacts from "unique" ones, and when put together, each component changes a system's performance. For instance, every basic installation of a Microsoft Windows operating system offers the same capabilities, such as hardware abstraction and the possibility to execute a wide range of binaries; adding a copy of the popular QuickTime 6 player extends these with capabilities to replay certain types of media files; finally, a specific QuickTime movie is added which can be enacted by the previously combined components. Each artifact in this example is clearly **bound** and therefore all potentials, known and unknown, can be reproduced.

A **blurry** boundary is introduced once a locally kept artifact is performing interactions with remote objects or reacts to states specific to the context of the actual execution. Remote APIs or resources could be required on every abstraction layer: the operating system might query a license server (Windows XP), the QuickTime player's installer might attempt to download system-dependent components (QuickTime 6.01), the QuickTime movie itself might embed remote resources via SMIL (supported since QuickTime 4). All of the technical interfaces designed to interact with remote resources in this example have changed over time or disappeared completely.

If an object is in its entirety located remotely, exposing an unknown range of possible performances, it must be considered **boundless**. A typical examples might be any social media platform like Twitter, which provides many modes of access to items with complex relationships inside the platform and further remote sources, but offers no way to inspect the defining processes or even creating an index of provided items from an outside perspective.

In some cases, the executable instructions that cause interactions with remote resources might be removed from a digital artifact. Skipping calls to obsolete usage tracking services or licensing checks

could remove a major blockade and make a piece of software usable again. However, commercial, closed-source software (esp. in binary form) cannot be easily modified or adapted. Even if the effort might be economically justified— when it happens on a high abstraction layer like an operating system, positively affecting the re-performance of many other objects—, simple (binary) patches that, let's say, redirect requests to a different network target may not be sufficient to re-enable the object's original behavior; more sophisticated patches may have unpredictable side effects.

In order to keep a blurry object operational, three different technical approaches are possible:

(1) Auxiliary machines, e.g. databases, web-servers, or even license-servers may be preservable using an emulation strategy, and included into an execution environment for a class of software or a single specific artifact.
(2) If auxiliary machines are not within the curator's reach, stub-interfaces, emulating the original external interface, could be implemented to allow a reduced but functional set of communication with a locally available software object.
(3) Finally, if a software is sending a countable amount of queries in a fixed range to a remote system, the occurring network traffic could be recorded and bundled with the software. For instance, a software installer that downloads components from the web would be stored with a web archive containing the required resources.

The option to fully preserve a computational service via emulation (1) seems to be the most desirable, as it offers the full range of potentials of the original setup. For bound units, emulating and orchestrating such services is technically within reach.

When it comes to web-based services such as YouTube, twitter, etc, technical complexity and size poses limits. These objects are to be regarded as boundless, there is no way to *preserve* them while ensuring the continuous availability of all provided interfaces and potentials. Even if the technical infrastructure to create a complete copy of YouTube would be available, the main purpose of preservation—reducing the actively maintained surface and maintenance frequency of an object by abstracting its complexity—would be economically unattainable. YouTube the service requires YouTube the organization to provide its full performative potential.

Even a small to mid scale web service under the control of the organization that seeks to preserve it might turn out to be spread across several virtual machines or making use of external microservices, requires new strategies and concepts.

Web archiving as a discipline has probably the longest development history with the preservation of black-box networked resources: storing only requests and responses occurring between a client and servers on the HTTP protocol layer, web archiving is abstracting any software running on remote servers, and therefore effectively creating *documentary* rather than performative resources.

Still, concepts for object boundaries have not been articulated with too much clarity in web archiving. Either the web as a whole is defined as a single object, or, in the common scenario of crawler-based web collecting, object boundaries are assumed to technically match a hierarchical structure of URLs pointing to web resources, usually being located under a single domain.

Both assumptions are problematic. "The whole web" doesn't really exist, since the most relevant web sites today are interactive and customizable so they appear different on each access for each user's (or robot's) context. The responses to a request for data from a URL such as `https://twitter.com` naturally has to differ for each user in order for the service to make any sense. As twitter in itself has no generalizable form, it remains boundless from the perspectives of web archiving, and, as previously laid out, software preservation.

Additionally, in today's web, within a single session, static resources, services, and complex JavaScript building blocks are pulled in from from dozens of CDNs, service providers, social media sites, advertising networks, and more, in order to create the impression of a single web page object by the browser. Many web sites, such as Instagram, do not implement a hierarchical URL scheme at all.

Even if all URLs under a single domain could be accessed and stored, blurriness would occur under a host-centric boundary definition when the amount of meaningful requests and responses approaches infinity; this can easily be the case with for example database-driven sites where the relationship of request and response is dependent on computation of complex or unknown states on client and server. For instance, while given enough time and resources, it is possible to preserve every map tile graphic and its URL from a mapping service such as Google Maps by web archiving practices, it is impossible to store all possible requests and responses for lat/long coordinate queries to the Google Maps service.

New web collecting mechanisms and concepts, as implemented by Rhizome's tool Webrecorder [6] and described by Hawes as "the act of archiving," have introduced the possibility for creating contextual object boundaries, taking into account the perspective and intentions of the curator as a web user, shifting to storing HTTP traffic occurring during time-bound web sessions rather than namespace-defined delimiters. Actions performed in collecting sessions are recallable in access sessions:

> The recording represents the curator's own vantage point reflecting the specific "requests" made (commands, clicks) — and tracing the actual path taken. Unlike a traditional web-crawler, which is provided with a seed URL and automated to explore a site in full, Webrecorder is curator-operated: subjectivity and selection *replace* automation and exhaustivity. [2]

This approach has the curator defining a synthetic boundary inside a boundless object, consciously discarding any performative potential outside of it, creating an attainable and verifiable goal: the actions performed during the capturing of the object need to be reproducible in the future.

3 PERFORMATIVE BOUNDARIES AND REPRODUCIBLE PROPERTIES

Digital preservation practitioners have been dealing with *variable* objects by defining "significant properties," core attributes of objects that should resist change over time even as performance environments change [4]. With *blurry* and *boundless* objects discussed above, all of which expose aspects of infinity, unavailability, and unknownness, significant properties cannot provide much guidance for preservation purposes. For instance, countless works of

net art achieve their affect by performing computation on resources from all over the web, and make use of or are fully located on black box services. The same might be true for a modest Excel sheet that attempts to load tabular data from a remote data source.

Describing significant properties of that kind means setting up preservation projects for failure, as there isn't a way to meaningfully address blurry or boundless objects within this framework. In the field of art preservation, this often leads to "remakes," in which large parts of a dysfunctional piece are re-created from scratch. This process represents a very large "actively maintained surface" and therefore big economical demands.

This paper suggest to define *reproducible properties*, clearly defining what potentials or "paths" will be preserved from a curatorial perspective, and then to take the required preservation actions. These properties shall be reproducible in an objective manner, to guide the development or adaptation of future preservation strategies and technical systems.

The tighter such a performative boundary is drawn and the fewer reproducible properties are included, the closer an object moves from full performative preservation towards documentation.

The following curatorial measurements can be used to define and preserve reproducible properties:

(1) Network traffic occurring during certain performance states of a software could be recorded, bundled with the preserved software, and be made available to the software during re-performance.
(2) The breadth of possible execution paths varies throughout performance sessions of an object. A curator could identify a state in a session that is the most relevant and snapshot the environment during performance, in a "switched-on state." With that snapshot being stored and used for re-enactments later, the object becomes pre-configured with resources available and input occurred before the freeze, making it more independent from these sources being available in the future.
(3) The access environment is modified to remove the ability for the user performing certain actions; for instance, an emulation framework would prevent local input events like mouse clicks or key presses to be routed to the re-performed environment under certain conditions, preventing the user from bringing the system into undesired states.
(4) The environment used for re-performance is configured or modified to reduce the potential breadth of certain computational performances by deactivating any interaction with external entities that are determined to be not relevant for the preservation goal.
(5) Finally, the *mis-en-scène* of a preserved object could highlight certain affordances to the user inside the environment that lead to a successfully bound re-performance, while not technically preventing other paths from being explored. For instance, a browser could be configured to launch with a certain home page, a limited set of icons could be placed in the center of an otherwise empty desktop, and so forth. Rhizome has used this technique for its online exhibition program *Net Art Anthology*[1]. Similar techniques have been successfully used for gallery space ("offline") exhibitions of net art [1].

None of these methods provides a generalized solution, but can be applied in combination to reduce an objects performative complexity for preservation purposes. Specifically, the shortcomings are:

- Methods (1), (2) and (3) require a careful, transparent structuring of artifacts, such as software components and recorded network sessions, in order to allow for their orchestration to be changed in the future. While single artifacts in such an ensemble can be regarded as stable, it must be avoided that future enhancements, e.g. due to unpredicted availability of currently unavailable external resources, become difficult to orchestrate or implement, or are risking to compromise previously established reproducible properties.
- Additionally, whenever network traffic should be recorded and replayed (1), detailed timing and cryptography might prevent reliable reproducibility. Especially software using security measurements like certificate pinning or authentication procedures that take the possibility of "replay attacks" into account might be impossible to re-enact.
- Because detecting state changes in a running software environment is very unreliable, input fencing as described in method (3) would need to be applicable during a complete session with an object, or be changed based on very simple measurements of for example time passing during the session. It can in general not be applied to software performing undesired function independent of user interaction.
- Changing an execution environment to reduce the breadth of an object's performance (4) works very well in the semantically rich sphere of the web, but not necessarily for binary executables. For example, Webrecorder prevents video streaming sites from switching in between low quality and high quality versions of the same video mid-stream, and instead forces loading a single, reproducible stream. Doing similar things with binary executables is much harder and potentially object-specific.
- While the mis-en-scène (5) is an easy, effective, and very economical way of defining a performative boundary by essentially guiding users during access, it might be technically specific to the context of the re-enactment, or culturally specific to the time the re-enactment was staged. For instance, if future users cannot understand a written hint to "click" a mouse because they have never used a mouse, or cannot interpret in what direction an arrangement of icons is supposed to nudge them since that form of interaction has in general fallen out of use, that boundary definition becomes ineffective.

4 OUTLOOK

A major category of software prone to preservation will be (mobile) "apps." Most apps have been designed to run on "always-on" network-enabled devices, whereas the locally available executable only provides a viewer for remote content. Additionally, some apps rely on an extended user-context such as GPS data. An out-of-context execution of apps would then result in a defunct application.

[1]Rhizome Net Art Anthology, https://anthology.rhizome.org/

These circumstances are not tied to the rise of mobile apps: Defining object boundaries based on locally running binaries has already caused the effective loss of software created in the late 1990's, when Windows' OLE architecture made web resources easily available to developers. [5]

Also, the "traditional" software industry is changing, as app stores for desktop computers (e.g. Apples OSX App Store), so called in-application purchases with only basic software versions shipped to customers and further features being added on-demand, as well as remote applications like Office365 or rolling releases like Windows 10 are becoming more popular. These new kinds of objects form a huge body of novel challenges to the preservation community, requiring not only technical analysis but increasingly *curatorial agency*. The aspiration of fully preserving any type of computational performance has to meet a reality of highly complex, networked ensembles, limited access to core components, highly context-dependent operation, and—as always—limited resources for preservation.

To address these challenges the concept of object boundaries and reproducible properties can help to define preservation intentions and to measure the (future) success of preservation actions.

Working with boundless objects entails the option and sometimes need of future changes, additions or adaptation of preservation strategies. Reproducible object properties are therefore important to protect the results of past preservation activities. Furthermore, boundless objects require closer collaboration—different preservation initiatives may be contribute different aspects or views on the same object.

Similar, (better) orchestration of different preservation strategies is a precondition. It is difficult to imagine that there is a single strategy or technological solution to preserve complex, distributed and multi-faceted objects. Web archiving, emulation and migration are all able to contribute to an object's fidelity and future quality of access. A coherent way of orchestrating such objects—time context, technologies used, etc.—is yet to be defined.

REFERENCES

[1] Dragan Espenschied, Klaus Rechert, Thomas Liebetraut, and Oleg Stobbe. 2016. Exhibiting Digital Art via Emulation. In *Proceedings of the 13th International Conference on Digital Preservation, iPRES 2016, Bern, Switzerland, October 3-6, 2016.* http://hdl.handle.net/11353/10.503174

[2] Anisa Hawes. 2018. Collecting and Curating Digital Posters: a collaborative pilot study using Rhizome's Webrecorder. Accelerated Art History: Tools and Techniques for a Fast-Changing Art World, panel at CAA 2017, New York. (2018).

[3] Davis S Heslop, H and A Wilson. 2002. An approach to the preservation of digital records. (2002).

[4] Pip Laurenson. 2014. Old Media, New Media? Significant Difference and the Conservation of Software-Based Art. In *New collecting*, Beryl Graham (Ed.). Ashgate Publishing Limited, Surrey (UK) / Burlington (USA), 73–96.

[5] Lynsey Jane Moulds. 2017. Christmas as a Service. (2017). https://rhizome.org/editorial/2017/dec/21/christmas-as-a-service/ [Online; posted 2017-12-21].

[6] Anna Perricci. 2017. Web archiving for all! Web archiving with Webrecorder. (2017). https://dpconline.org/blog/webrecorder-blog [Online; posted 2017-09-11].

[7] M Ragan-Kelley, F Perez, B Granger, T Kluyver, P Ivanov, J Frederic, and M Bussonnier. 2014. The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication.. In *AGU Fall Meeting Abstracts*.

[8] Klaus Rechert, Isgandar Valizada, Dirk von Suchodoletz, and Johann Latocha. 2012. bwFLA – A Functional Approach to Digital Preservation. *PIK - Praxis der Informationsverarbeitung und Kommunikation* 35, 4 (2012). https://doi.org/10.1515/pik-2012-0044