The Current State of Software Preservation

Nicole Contaxis, National Digital Stewardship Resident at the National Library of Medicine

Executive Summary
Software preservation is an important part of the stewardship of cultural, scientific and social history. This report outlines several ongoing projects and their preservation strategies. While many of these strategies include preserving software as an interactive digital object, strategies that simply document the software, like recording game play in online video games, are also included. The paper begins by outlining key tools and concepts as well as the over-arching obstacles to software preservation, including a lack of contemporaneous documentation, the wide variety of use cases for software, and copyright. Afterwards it discusses three categories of software preservation projects: software collections in libraries and archives, digital art conservation, and computational reproducibility. With in-depth discussions of the benefits and problems with many of these projects, the paper discusses how different strategies best serve different types of software, institutions, and budgets. Both technical and administrative strategies are considered. Projects addressed include the National Software Reference Library (NSRL), the Olive Archive, Emulation as a Service, the work at Rhizome, the Game Play Capture Project at the Strong Museum of Play, the Astrophysics Code Library (ACL), and others.

Introduction
Preserving software is a vital aspect of the stewardship of cultural, scientific, and social history. The 2015 National Agenda for Digital Stewardship calls for serious action in the realm of software preservation. It states, "[Software] is both the key to accessing and making sense of digital objects and an increasingly important historical artifact in its own right."[1] The document argues that preserving software is necessary for two reasons: (1) it is a cultural artifact worthy of historic study in its own right; and (2) it is a means to access data and content held in obsolete formats. Ensuring long-term access to executable products is necessary in order to preserve the content created, managed, and accessed through software as well as to preserve the history inherent in the creative endeavor of software development itself. The digital stewardship community cannot afford to ignore executable files.

The impetus for this paper is a project at the National Library of Medicine (NLM) called "NLM-Developed Software as Cultural Heritage." This is a National Digital Stewardship Residency (NDSR) project, funded by the Library of Congress (LC) and the Institute of Museum and Library Services (IMLS), which aims to better understand the history of software development at NLM and to create a preservation strategy for at least one of NLM's software products.[2] NLM has a long history of developing software, both for internal needs and for its users. This overarching goal of the NDSR project is to preserve this history for future use.

---

[1] "2015 National Agenda for Digital Stewardship." National Digital Stewardship Alliance. September 1, 2014. Accessed November 12, 2015. http://www.digitalpreservation.gov/ndsa/documents/2015NationalAgenda.pdf

[2] "NDSR Project: NLM-Developed Software as Cultural Heritage." National Digital Stewardship Residency. December 1, 2014. Accessed November 12, 2015. http://www.digitalpreservation.gov/ndsr/NLM NDSR NLM-Developed Software as Cultural Heritage-1.pdf

Examples of such software include an internal cataloging and acquisitions system, conceived in the late 1970s and created because contemporary software vendors could not meet the library's needs. Only in the late 1990s did the library begin to use a vendor's cataloging software, with serious adaptations to accommodate the library's unique holdings. MEDLARS, another example of NLM-developed software that began in 1961, was a pioneering project that provided public access to computerized library records, pre-Internet or other networked technologies.

This project is divided into two components: (1) understand of the history of software development and implementation at NLM; and (2) the creation of a preservation strategy for at least one piece of software. This latter aspect of the project necessitates in-depth research into the strategies being implemented at other institutions, which is reflected in this paper. As will be established later, there is no model framework for software preservation since the needs of the institutions, the functionality of the software, and the current state of the software can all vary wildly, but this paper hopes to provide guidance for institutions as they consider how to preserve software for their own needs and the needs of their patrons.

An extreme illustration of the need for software preservation was featured in a post on *The Signal*, the Library of Congress's digital preservation blog in an interview with Doug White, head of the National Software Reference Library (NSRL). A medical supply company had distributed a shipment of Botox that had been improperly produced, and although the FDA had all of the information necessary to perform a recall, the information was formatted for obsolete software. Luckily, the NSRL possessed a copy of the software so that the data could be read and the FDA could recall the shipment. In this instance, software preservation proved life-saving.[3] Less sensational examples include government emails that are created and stored in obsolete formats.[4] If a copy of the original software does not exist, accessing those emails may be expensive or even impossible. Preserving software, in this way, is necessary to ensure government transparency in the digital age. Information access is dependent on software preservation strategies.

While preserving software allows one to access data held in obsolete formats, it also allows one to view and understand the manner in which data is created. In STEM fields, it is necessary to be able to reproduce findings, but as many STEM fields rely heavily on computation, it may be necessary to preserve the code that performs that computation. For example, if a computational biologist runs a certain algorithm on a set of data in order to produce her results, it is necessary to preserve that algorithm as part of her larger methodology.[5] Although this type of research is

---

[3] Owens, Trevor. "Life-Saving: The National Software Reference Library." Life-Saving: The National Software Reference Library. May 4, 2012. Accessed November 12, 2015. http://blogs.loc.gov/digitalpreservation/2012/05/life-saving-the-national-software-reference-library/

[4] Cush, Andy. "Texas Town Is Charging Us $79,000 for Emails About Pool Party Abuse Cop." Gawker. June 29, 2015. Accessed November 12, 2015. http://gawker.com/texas-city-is-charging-us-79-000-for-emails-about-pool-1714757746?trending_test_d&utm_expid=66866090-62.YkETBcIMTk2uX1oytHipyg.4

[5] Di Cosmo, Roberto. "Keynote - Preserving Software: Challenges and Opportunities for Reproducibility of Science and Technology." SciLab: Open Source Software for Numerical Computation. May 21, 2015. Accessed

more prevalent in STEM fields, it is also used in the humanities and social sciences with the growing popularity of digital humanities and big data in the social sciences.

Software also constitutes a creative enterprise in its own right and will be useful to future political, economic, and cultural historians. Matthew Kirschenbaum, for example, writes about the history of word processing and its cultural importance in his book, *Track Changes: A Literary History of Word Processing,* to be published later this year.[6] In discussions of this topic, he talks about the adoption of word processing software, its depiction in popular culture, and how the word processor itself affected the creation of text and literature throughout the late 20th century. Stephen King, for example, was one of the early adapters of the word processor, and its use appears in some of his lesser-known works like the short story, "Word Processor of the Gods." While King's thoughts are important because he is a cultural figure, word processors have also changed how day-to-day activities are accomplished and how everyday communication now takes place. If these pieces of software are not saved, how will our future historians adequately interpret our actions, our practices, and our creations? Concerns about preserving video games, as recently acknowledged by the Library of Congress with its additional copyright exceptions for the preservation and care of historic video games, are also a key part of preserving software as cultural heritage.[7]

In the face of this need, how should an institution or individual approach creating a software preservation strategy? Software preservation is not always necessary. For example, a library would be unlikely to emulate Word 5.0 in order to read documents saved in that format. However, it may be necessary to emulate Word 5.0 in order to migrate those documents into new formats if they were not migrated in a timely fashion.[8] Furthermore, as stated earlier, although it is sometimes necessary to preserve software in order to view data, it is also necessary to preserve software in order to understand how that data was created. Preserving software is necessary in order to thoroughly preserve the research methodology of many burgeoning and emerging fields. In conjunction with the cultural importance of software, it is clear that it is important to preserve software, but considering the variety of use-cases, it is not always clear how preservation should be accomplished.

This lack of clarity exists for several reasons: (1) the ubiquity of software; (2) the complexity of software and its dependencies; and (3) the variety of software designs and applications. Software is present in almost every aspect of modern life, and due to the wide variety of use cases, it can

---

November 12, 2015. https://www.scilab.org/community/scilabtec/2015/Keynote-Preserving-Software-challenges-and-opportunities-for-reproducibility-of-Science-and-Technology

[6] Kirschenbaum, Matthew G. "Stephen King's WANG: A Literary History of Word Processing." Lecture, Stephen A. Schwarxman Buildling, Wachenheim Trustees Room, New York Public Library, New York CIty, December 16, 2011.

[7] Higgins, Parker, Mitch Stoltz, Kit Walsh, and Corynne McSheery. "Victory for Users: Librarian of Congress Renews and Expands Protections for Fair Uses." Electronic Frontier Foundation. October 27, 2015. Accessed November 12, 2015. https://www.eff.org/deeplinks/2015/10/victory-users-librarian-congress-renews-and-expands-protections-fair-uses

[8] Rosenthal, David. "Emulation and Virtualization as Preservation Strategies." https://mellon.org/media/filer_public/0c/3e/0c3eee7d-4166-4ba6-a767-6b42e6a1c2a7/rosenthal-emulation-2015.pdf

be difficult to pin down the exact context and infrastructures that allow a particular piece of software to communicate meaning. In other words, when software encompasses everything from the algorithms for computational chemistry to the video game, *Street Fighter 2*, the preservationist needs to educate herself on the particularities of that specific piece of software in order to understand how best to preserve it.

Each piece of software has a series of dependencies that can be difficult to discover. Most software is dependent on specific hardware to operate properly, but code libraries can also present a serious obstacle for the software preservationist. A code library is a resource for software development that provides pre-written code for a piece of software to implement. However, if that code library is no longer available at the point of preservation, the piece of software that utilizes code from that library may no longer function in the way that it is intended or there may be limited documentation about the components taken from the code library.[9] The same can be said of pieces of software that pull data from APIs, which may supply vitally important data by pulling it from other sources but not storing it locally.[10] So, the preservationist needs to do more than educate herself on the software as a discrete object; she needs to investigate the legal and technical infrastructures that the software relies on in order to function and convey meaning. Because these dependencies are not always explicitly mentioned in documentation or described at a consumer level, this research process can be arduous.

In the face of the complexity and ubiquity of software, the tools and strategies for its preservation are varied and, in many ways, still forming. Depending on the type of software being preserved, the institution preserving the software, and the copyright holder, there are different tools that may prove useful. In order to outline what these tools and strategies are, this paper will discuss four subjects currently under scrutiny: VMs, video games, digital art, and computational reproducibility. Before examining these subjects, however, it is necessary to discuss some of the universal obstacles to software preservation.

Concepts and Tools

At this point, a quick explanation of two basic preservation strategies is necessary. An institution can choose to emulate software or migrate that software's capabilities. Emulation means re-creating a computing environment so that the software can run as originally designed while migration means preserving the behavior of a piece of software in a new context. Migration may not preserve the look and feel of a piece of software, but it can preserve the basic functionality, making it a popular strategy in computational reproducibility efforts in academic fields. [11]

Due to recent developments in emulation, emulation is becoming more popular as a digital preservation strategy. Virtual machine projects, described more thoroughly below, present an

---

[9] "Code Libraries." Developer Network. Accessed November 12, 2015. https://msdn.microsoft.com/en-us/library/kews042w(v=vs.71).aspx

[10] Giaretta, David. *Advanced Digital Preservation*. Berlin: Springer, 2011. 126.

[11] McDonough, Jerome, Robert Olendorf, Matthew Kirschenbaum, Kari Kraus, Doug Reside, Rachel Donahue, Andrew Phelps, Christopher Egert, Henry Lowood, and Susan Rojo. "Preserving Virtual Worlds Final Report." IDEALS @ Illinois. August 31, 2010. Accessed November 12, 2015. Page 6. https://www.ideals.illinois.edu/handle/2142/17097

appealing method for emulation that requires less technical expertise than earlier methods. However, before discussing virtual machines in depth, it is necessary to talk about several other tools.

Computer forensics offers several important strategies for the preservationist. The first is a disk image. A disk image is a copy of an entire computing system. It allows a user to access hidden files, deleted files, and other aspects of a computing system that may not be available if they copied each files from a hard drive rather than creating a disk image.[12] Disk images will be mentioned in many of the projects highlighted below. There are a variety of tools for creating disk images including Guymager, an open source tool.[13] There are also a variety of tools for accessing disk images. BitCurator offers a Disk Image Access Tool which provides an interface to browse disk images, export files and deleted items, and view disk image metadata. This tool provides a Graphical User Interface (GUI) that does not necessarily correspond to the original look and feel of the digital object.[14] The virtual machine projects highlighted below provide access to disk images in their original interface rather.

Virtual Machines

Out of the four categories of projects discussed in this paper, virtual machines (VMs) are unique. Rather than being concerned with a particular type of software or within a specific institution, VMs are a tool that can be used in many situations. Software tends to have many dependencies. It relies on a certain operating system, which in turn may rely on a certain type of hardware. VMs allow a user to run a different machine on their current computer, thereby circumventing some of these dependencies.

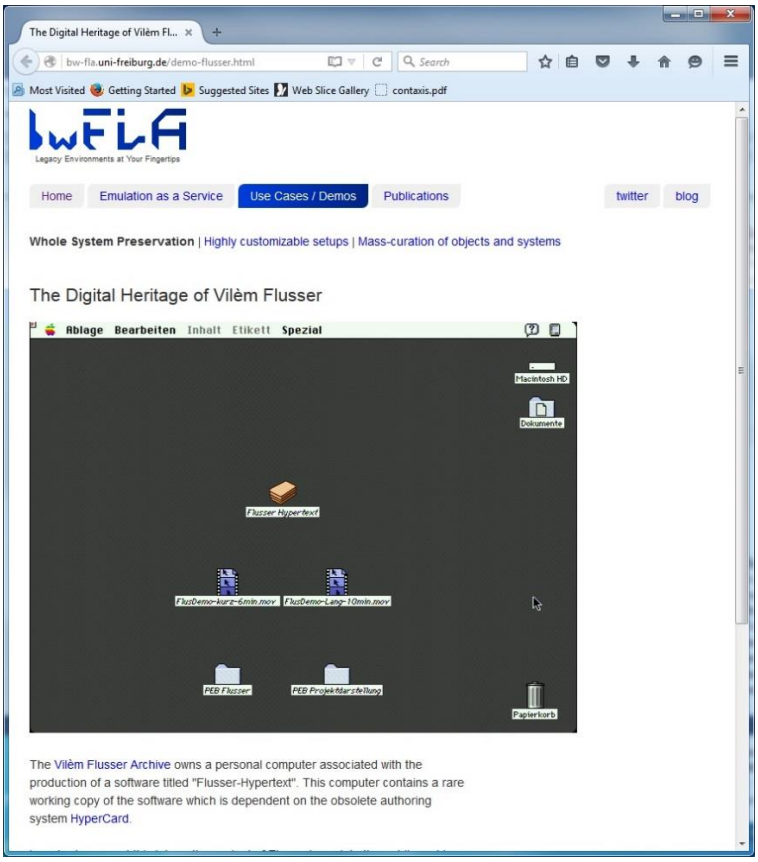
Before delving into two VM projects, an example may be helpful to illustrate exactly how a VM could work for software preservation and how it compares to other strategies. One example of an individual preservation project that implements a VM is the Vilem Flusser Archive. Vilem Flusser was a Czech-born journalist, writer, and philosopher. The Vilem Flusser Archive provides access to Vilem Flusser's digital materials via a VM, created by Emulation as a Service (EaaS), online. Users can peruse the materials from his computer online and view it through a window that mimics the experience of viewing materials on a mid-1990's Machintosh. A screenshot[15] is supplied below, as the emulation exists online in a modern browser:

[12] Ibid. Page 85.

[13] "Introduction." Guymager Homepage. Accessed November 12, 2015. http://guymager.sourceforge.net/.

[14] "BitCurator Access." BitCurator. October 18, 2014. Accessed November 12, 2015. http://www.bitcurator.net/bitcurator-access/.

[15] Screen shot taken on August 3, 2015 from http://bw-fla.uni-freiburg.de/demo-flusser.html



The Archive employs Eaas because Flusser's materials were created on a rare software that is dependent on an obsolete system.[16] In order for patrons to view his materials, the Archive thus employs a VM so that the software can function. In this way, a user can run another computer, with a different operating system and unique content, on their own computer. With this approach, users can access cultural materials in the same manner that the original creator handled them and accessed them himself.

A similar project at the University of California, Los Angeles attempts to re-create the computing environment of writer Susan Sontag complete with her born-digital materials as part of a larger collection of her materials. Within the confines of the library, patrons can request a computer that contains all of Sontag's born-digital materials. This single IBM laptop contains the content of several of Sontag's computers, which were originally donated as two external hard drives. This IBM laptop employs a product called "Deep Freeze," which allows patrons to use the computer while restoring the laptop to its original status after it is rebooted.[17] In other words, regardless of what patrons do on the laptop, it always returns to its archival form. The laptop is

[16] "The Digital Heritage of Vilèm Flusser." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. http://bw-fla.uni-freiburg.de/demo-flusser.html.

[17] Schmidt, Jeremy, and Jacquelyn Ardam. "On Excess: Susan Sontag's Born-Digital Archive." The Los Angeles Review of Books. October 26, 2014. Accessed November 12, 2015. https://lareviewofbooks.org/essay/excess-susan-sontags-born-digital-archive/.

not connected to the Internet, and the files cannot be accessed online in any way. Both the Flusser project and the Sontag project provide access to born-digital cultural heritage, but the Flusser Archive is able to provide wider access with its use of EaaS.

EaaS and the Olive Archive are currently the two most prominent virtual machine-based emulation projects. EaaS is a collaborative project from the University of Freiburg and several other partners. It aims to create legacy computing environments that people can customize to their specific needs and therefore enabling them to run other programs or environments that are dependent on those legacy systems.[18] One of the use cases supplied by EaaS is the conversion of obsolete Microsoft Office Word files to current formats.[19] A user can run the VM in order to access the original files and then migrate them to more standard, preservation friendly formats, so that they will be easier to access in the future. This sort of strategy can be quite helpful when a piece of software needs to be preserved just so that the data it reads can remain accessible. Preserving Word itself could be difficult because of copyright concern, but creating an environment in which older versions of Word can run and be used skirts some of these issues. Because EaaS is used, or is proposed to be used, in a variety of ways, many projects that employ EaaS will be highlighted throughout this paper. Since VMs are a flexible tool, it is important to discuss how they are implemented in different projects, not just that they are implemented.

The Olive Archive functions similarly but has been used in different circumstances. This project is housed at Carnegie Mellon University, and bills itself as an archive of VMs. Some of the machines in their collection include Netscape Navigator 1.12, NCSA Mosaic 1.0, DOOM for DOS, Microsoft Office 6.0, and TurboTax 1997.[20] In effect, the Olive Archive has created these machines as pre-existing environments for users to implement on their own computers, having retrieved the disk image from the Olive Archive library. Rather than requiring users to have in-depth technical knowledge on the computing environment that they require, this allows them to implement an environment easily. While users need to download a client in order to run these environments, the interfaces retain their original user friendliness.[21]

These machines can be used to access data stored in obsolete formats or to view old and obsolete software in an emulation of its original computing environment. For example, the TurboTax1997 VM was created because of the unique historical value of TurboTax software. TurboTax is updated every year to reflect new tax laws, and political science students of the future could use these VM to better understand the individual ramifications of tax law change over time.[22] A

---

[18] "Emulation as a Service." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. http://bw-fla.uni-freiburg.de/

[19] "Use Cases." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. http://bw-fla.uni-freiburg.de/demos.html

[20] "Virtual Machines in Our Collection." Olive Archive. Accessed November 12, 2015. https://olivearchive.org/docs/collection/

[21] "Installing the Client." Olive Archive. Accessed November 12, 2015. https://olivearchive.org/docs/vmnetx/install/.

[22] Satyanarayanan, Mahadev, Gloriana St. Clair, Benjamin Gilbert, Jan Harkes, Dan Ryan, Erika Linke, and Keith Webster. "Olive: Sustaining Executable Content Over Decades." School of Computer Science at Carnegie Mellow University. November 1, 2014. Accessed November 12, 2015. Page 4. http://www.cs.cmu.edu/~satya/docdir/CMU-CS-14-115.pdf

screen shot of TurboTax 1997 being run through the Olive Archive, taken from the paper “Olive: Sustaining Executable Content Over Decades” by Satyanarayanan et al, is featured below.

VMs can be used in a variety of situations. As will be outlined later in this paper, VMs from both The Olive Archive and EaaS are implemented in several software preservation projects. Both EaaS and The Olive Archive provide a valuable service and are becoming increasingly important to the software preservation and digital stewardship community as time wears on and more digital formats become obsolete.

Universal Obstacles

Although there is significant variety in software, software development, and software preservation strategies, there are some universal obstacles that bridge these differences. These obstacles range from the theoretical to the practical and are generally deeply entangled. The decision of how to solve one problem directly affects the possible solutions to other problems. In this way, the process of preserving software necessitates a series of decisions that can have a domino effect on each other, and separating each decision can prove difficult.

The largest challenge for software preservation is deciding exactly what needs to be preserved. Due in part to the nature of executable files, there are several ways one can conceive of preservation and the assurance of long term access. One can consider it their obligation to preserve the code itself, the experience of using the software, or some combination of the two. Depending on how this issue is addressed, the preservation strategy will differ wildly. Choosing to migrate software may make sense for the Astrophysics Source Code Library since they only need to preserve the functionality of the software. [23] Emulation, however, is more reasonable for

[23] "Preserving.Exe: Toward a National Strategy for Software Preservation." October 1, 2013. Accessed November 12, 2015. Page 23.

for projects that will want to provide users the opportunity to see how the interface functioned in historic software. [24]

It is clear that the way an institution defines what aspects of a piece of software to preserve will affect the other obstacles it faces. However, there are other obstacles that all institutions will contend with regardless of which aspects of a piece of software they prioritize in the preservation process. Versioning, for example, presents considerable practical problems. Software is generally released in many versions, and because of patches and updates, defining the exact parameters of a version of software can be difficult. Unlike a book which has clearly defined editions, the boundaries of software can be murky. Software, particularly after the advent of the Internet, can be updated bit by bit. With a patch here and a patch there, a single piece of software can undergo many changes without even being considered a new, official version.[25] While technology like Git now allow developers and others to better understand software versioning by tracking changes as they happen, it can still be difficult to trace which version should be preserved and where along that version's timeline it should be preserved.[26] As an interesting new phenomenon, Microsoft has abandoned the idea of versioning entirely and has called Windows 10 "the last version of Windows." Instead of providing concrete, new versions of the operating system, Microsoft will instead offer the operating system "as a service" that updates in real time with development changes.[27] While most development companies still employ versioning, if the practice of providing software as a service is to become a design trend, its opaque fluidity model may present even further difficulties for the preservationist.

Practical considerations can also inform strategy choices. When attempting to preserve software that is already long out of use, the code available and the condition it is in can force many decisions. Even if it becomes apparent that one version of the software was seminal within that software's lifetime, examples of that version may no longer exist. Although one may be able to locate documentation about the version, establish that it is the ideal version to preserve, and define where along the version's timeline to capture the information, it may still be impossible to preserve that version depending on how the copies of that version have been cared for. Finding the code itself may be the biggest issue for some projects.

Regardless of the eventual preservation strategy, there are universal issues for software preservation. Two of these are copyright and a lack of documentation. These obstacles are more tied to the history and philosophy of software and software development than intrinsic to the structure of the software itself. This distinction is important because it draws attention to the fact that these obstacles will differ slightly between institutions and may change radically over time.

---

[24] Ibid, Page 24.

[25] The Mac Operating System, for example, is updated frequently to deal with security risks and small changes. But these slight changes may not be considered when naming new versions of the operating system for public consumption.

[26] This opacity may not be purposeful. Frequently, due to the way that software is developed, the team themselves may not be clear about what updates were done when if not employing a tool like GitHub.

[27] Kelly, Gordon. "Why Microsoft Announced Windows 10 Is 'The Last Version of Windows'" Forbes / Tech. May 8, 2015. Accessed November 12, 2015. http://www.forbes.com/sites/gordonkelly/2015/05/08/microsoft-windows-10-last-windows/

At the moment, these issues are universal to software preservation projects, but they may not be in the future.

Copyright exceptions that allow libraries to preserve materials do not exist for software in the same way as they do for books, prints or other analog materials. Libraries, for example, are legally able to digitize books for preservation; they do not need to worry about being sued for, technically, reproducing the book.[28] However, because the Anti-Circumvention Clause of the Digital Millennium Copyright Act (DMCA) makes it illegal for libraries to break Digital Rights Management (DRM) software, it is difficult to even view the source code of some software in order to preserve it.[29] Although this paper will not address copyright issues directly, it is important for individuals or institutions involved in software preservation to research their copyright standing.[30] Copyright can present a formidable obstacle to software preservation.[31]

Copyright is less of an issue for certain projects. At NLM, for example, the software under consideration for the preservation aspect of this project was developed in-house and most federal government products cannot be copyrighted. It is possible that a piece of software may have been developed through a partnership with another institution or business, and that partner may have a stake in copyright ownership. However, we have yet to locate an example of this in NLM's collection of software. Other institutions, like the Internet Archive, may be more risk-tolerate and skirt copyright law by preserving software whose copyright holder no longer exists, termed 'abandonware' and discussed later. While this is not technically legal, it does act as a functioning workflow. While copyright may not be an overwhelming concern for either of these scenarios, it is important to still keep copyright in mind as an aspect of risk management.

Researching copyright status can be quite difficult. In fact, finding any information about the development process can be difficult, particular when investigating an older piece of software. Without a robust records management plan, documentation of the development process is frequently lost, and tracking down relevant information can be quite time-consuming. While the amount of documentation needed will vary according to the exact preservation strategy implemented, having information about the development process can answer questions about copyright, creator intent, and user interaction with the software.

Knowledge-finding in an environment of frequently sparse documentation can be frustrating, but there are still options. At NLM, for example, there is a plethora of documentation for educating patrons on user-facing software. This documentation helps illustrate both how users interacted with the software and how developers understood user needs. This information can be helpful in choosing how to preserve software or deciding whether it requires preservation at all. As software development has changed since the early years of development, a deeper understanding

---

[28] "17 U.S. Code § 108 - Limitations on Exclusive Rights: Reproduction by Libraries and Archives." Legal Information Institute. Accessed November 12, 2015. https://www.law.cornell.edu/uscode/text/17/108

[29] "17 U.S. Code § 1201 - Circumvention of Copyright Protection Systems." Legal Information Institute. Accessed November 12, 2015. https://www.law.cornell.edu/uscode/text/17/1201

[30] "Digital Millennium Copyright Act." Legal Information Institute. Accessed November 12, 2015. https://www.law.cornell.edu/wex/digital_millennium_copyright_act

[31] "Preserving.exe." Page 28.

of the need of documentation has manifested through tools like Git, and hopefully the issue of finding documentation will be less prevalent for software produced later.

The variety of preservation strategies currently being implemented deal with these obstacles in different ways and to varying degrees of success. It is important to keep in mind, however, that regardless of the differences for the strategy and type of software being preserved, there is a common ground and common concerns across these projects. Preserving software may be a wide and varied practice, but it aims to solve issues that will become more and more present as time goes on.

Solutions

There are four main domains were ongoing projects are attempting to preserve software: (1) Software and computing collections; (2) Digital art collections; and (3) Computational reproducibility efforts. Each of these domains reflects a different type of project currently underway regarding software preservation. These projects rely on varying intellectual histories in order to form and implement their strategies, including art conservation, librarianship, and data science. While these three domains are not discrete and concepts and practices overlap, they represent four dominant pushes in software preservation currently.

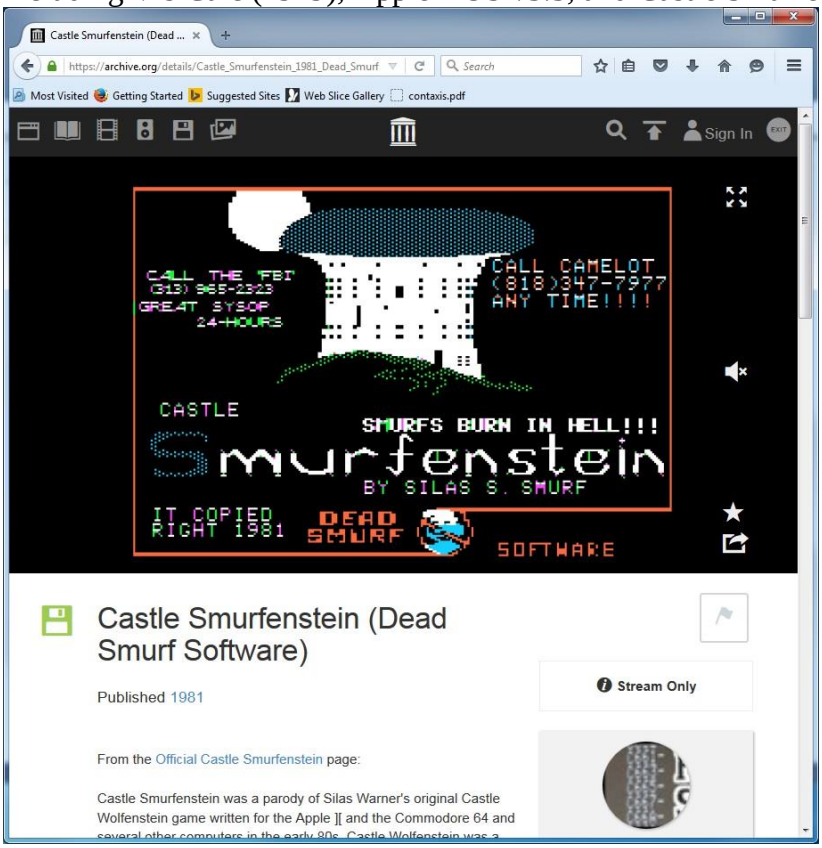
It is important to note that many of these issues and projects are mentioned in or stem from the Library of Congress report, *Preserving.exe: Toward a National Strategy for Software Preservation.* Reporting on the 2013 National Digital Information Infrastructure and Preservation Program (NDIIPP) summit of the same name, this document focuses on the need for software preservation, identifying valuable and at-risk software, and ways to ensure long-term access to executable files. Two of the goals of the summit were to encourage new partnerships to support software preservation and to advise the Library of Congress about next steps to ensure long-term access to software and the data under its care.[32] Similarly, NLM's software preservation project is a continuation of these goals and a fist attempt at confronting these issues in the same way as many projects described below.

Software and Computing Collections

Materials about software and computing can be found in many libraries and archive. The Computer History Museum in Mountain View, California, holds large amounts of archival material and allow for some access to those collections online. These types of institutions need to preserve software in a way that is cohesive with larger understandings of archival and library practice. In other words, these institutions have to preserve and provide access to software collections through the mechanisms, like catalogs, that they currently employ for a wide range of mediums and content formats. While individual practices vary between institutions, these libraries and archives have large collections and need to create access and preservation strategies with limited resources. In this approach, there is less of an interest in individual-object conservation and more of a focus on batch-preservation or mass-preservation strategies.

[32] "Preserving.exe." Page 2.

Perhaps the most well-known project for software and computing preservation is the Internet Arcade, which allows users to play old video games online through an emulation. However, the Internet Arcade is only a subset of a much larger collection of historic software being preserved by the Internet Archive. It currently holds 110,168 pieces of software in their online collections, including VisiCalc (1979), Apple DOS v3.3, and Castle Smurfenstein.[33]



Castle Smurfenstein, pictured above as an emulation in a modern computing envrionment, is an example of abandonware. Abandonware is software that is ignored by its owner or manufacturer, generally because that owner no longer exists. As a subset of orphan works, there is rarely a clear or enforcing copyright holder for abandonware. Most, if not all, of the Internet Archive software holdings are abandonware.[34] The Internet Archive is relatively risk-tolerate and posts software because, many times, there is no existing entity likely to claim copyright ownership. Many libraries and archives are uncomfortable with this level of risk management, but it seems to work for the Internet Archive.

The technical implementation underlying the IA's approach is the JSMESS emulator, which ports the machine emulator MESS to Javascript.[35] JSMESS is open source and freely available

[33] Screen shot taken on September 28, 2015 from https://archive.org/details/software

[34] Scott, Jason. "Change Computer History Forever: Well, Here We Are." ASCII Text Files. April 13, 2013. Accessed November 12, 2015. http://ascii.textfiles.com/archives/3947

[35] Scott, Jason. "Still Life, With Emulator: The JSMESS FAQ." Internet Archive Blogs. December 31, 2013. Accessed November 12, 2015. https://blog.archive.org/2013/12/31/still-life-with-emulator-the-jsmess-faq/

on GitHub.[36]JSMESS runs in a contemporary browsers and only runs the requested software application. In other words, while EaaS is frequently used to run entire computing environments, JSMESS only runs the particular piece of software in question. Furthermore, unlike the Olive Archive, Internet Archive users employing the JSMESS emulator do not need to download a VPN or additional software in order to run it. In effect, a user can enjoy the JSMESS emulator without knowing what an emulation is or how it functions. Instead, they can simply enjoy playing Castle Smurfenstein.

The practices at the National Software Reference Library (NSRL) offer a stark comparison. As a National Institute of Standards and Technology (NIST) project, NSRL is a reference library, not a lending library. While people often come to them asking to borrow historic pieces of software, they do not lend software to outside agencies. The purpose of the National Software Reference Library is "to collect software from various sources and incorporate file profiles computed from this software into a Reference Data Set (RDS) of information."[37] Originally intended to assist federal, state, and local law enforcement with computer related crimes, encouraging and allowing widespread access to its materials is not the principal mission of NSRL. Because it generally functions as a reference library, it is primarily concerned with providing local access within their own computing environments. This lowers risk and the number of obstacles that they need to contend with.

The NSRL currently partners with the Stephen M. Cabrinety Collection at Stanford University Library. Both organizations are working to disk image large amounts of software. The Stephen M. Cabrinety Collection is a donated collection of materials that deal with the history of mini-computing and computing more generally.[38] While this process is ongoing, Henry Lowood, Curator for History of Science & Technology Collections and Film & Media Collections at the Stanford University Libraries, is considering a variety of other options, such as using the Olive Archive to run the historic software.[39] Of course, running the software and making it available and accessible are two separate considerations. However, in the case of the Cabrinety Collection, a VM seems like it will be integral to the future functioning of the archive, regardless of how accessible it will be.

Other organizations do not use emulation at all, and instead resort to a more simplistic preservation strategy. The Video Game Play Capture Project at The Strong National Museum of Play addresses preservation in a very different way. Rather than preserving the gaming software and providing continue access to the interactive aspects of the software, the Video Game Play

---

[36] "JSMESS." GitHub. Accessed November 12, 2015. https://github.com/jsmess/jsmess

[37] "National Software Reference Library." National Institute of Standards and Technology. Accessed November 12, 2015. http://www.nsrl.nist.gov/

[38] "Cabrinety-NIST Project." Stanford University Libraries. Accessed November 12, 2015. http://library.stanford.edu/projects/cabrinety-nist-project.

[39] Guins, Raiford. *Game After: A Cultural Study of Video Game Afterlife*. Cambridge, Massachusetts: MIT Press, 2014. 292.

Owens, Trevor. "Video Game Preservation at Scale: An Interview with Henry Lowood." The Signal: Digital Preservation. February 1, 2013. Accessed November 12, 2015. http://blogs.loc.gov/digitalpreservation/2013/02/video-game-preservation-at-scale-an-interview-with-henry-lowood/.

Capture Project records videos of players actually playing the games using the original gaming consoles.[40] Clearly, this strategy has some limitations. It will not allow future users to interact with what was an interactive object. Users will only be able to view someone else's experience interacting with the software. However, this strategy has several benefits, both for the institution and possible future users. First, it removes many of the legal and technical preservation issues for interactive objects. There is a longer history of preserving a/v materials than there is of preserving interactive materials, and the institution will be able to call on this history to inform their own strategies. Second of all, the user may benefit from watching the original, intended users interact with the game.

Many of the games in the Video Game Play Capture Project are Massive Multiplayer Online Role-Playing Games (MMORPGs), and in order for them to function accurately, the player needs to interact with other Internet-connected players. For future historians to understand what it was like to play the game, it may be more effective to view game play rather than to pay the games themselves. The social interactions on these games are historically important, and game play capture will allow them to better understand how users talked to each other and planned game action contemporaneously. This aspect of game play would be lost even if the game was perfectly preserved for future use. Additionally, future users will not need to learn how to play the game in order to enjoy it. These benefits, however, may not outweigh the loss of the actual digital object and its interactivity.

Perhaps because many of those spearheading these current attempts at software preservation are themselves video game and computing enthusiasts, there does not seem to be an extensive conversation about the issue of knowing how to use or interact with the software in the future, even if that software is accurately preserved. While game capture does not allow future users to play the game or interact with the software, it does avoid the issue of teaching future users the controls of a video game. My own experience trying to play games on the Internet Arcade illustrates how difficult it can be to teach oneself the controls without proper documentation, especially when a console-based game is being emulated on a desktop PC without all the same functionality as the original console.

Lastly, *Preserving Virtual Worlds* was a collaborative research project focused on how to preserve interactive media, particularly *Second Life*, a MMORPG. While the initial project was completed in 2008 and the subsequent one, *Preserving Virtual Worlds 2.0*, has also been completed, the partners were able to outline what specifically was at risk in video game preservation and offered early case studies of how to deal with video game archiving.[41] Much of their reporting is now outdated, however they did outline the uses of disk imaging, emulation, metadata, and hardware preservation that have been successfully employed in later projects.\.

Software metadata will not be discussed at length here, but it is an important aspect of software preservation since it allows for access and assists in identifying software, a current concern for

[40] "Video Game Play Capture Project." The Strong National Museum of Play. April 29, 2014. Accessed November 12, 2015. http://www.museumofplay.org/about/icheg/play-capture

[41] "Preserving Virtual Worlds Final Report."

organizations like NSRL.[42] However, because the uses for certain software applcations can vary immensely, metadata approaches tend to vary according to the specific use case in question. For example, Jerome McDonough, from the *Preserving Virtual Worlds* project at the University of Illinois at Urbana Champaign, argues for using a combination of OAI-ORE and METS to best accommodate the metadata needs of game preservation.[43] However, Angela Dappert, writing on metadata for computing environments, argues that the best approach combines TIMBUS, an EU project on the long-term digital preservation of business processes and services, and PREMIS.[44] Because these two researchers are working on different kinds of software, they have isolated different priorities for the metadata preservation process. Creating standards, or even a set of best practices, for software preservation seems to be greatly impeded by the variety of uses and structures of software.

Digital Art Collections

There is significant overlap between digital art collections and software and computing collections. Video games, for example, could fall into either of these categories, depending on how they are treated and understood. The game Pong may be included in a software collection, but it is also archived and exhibited at the Museum of Modern Art (MoMA). The distinction between these two arenas is less about the exact nature of the software in them and more about the way that the software is understood and handled. While software and computing collections work with large aggregations of materials in a mass production and preservation environment, digital art collections often work in the tradition of art conservatorship and spend more time and resources on the details of a particular piece.

Rhizome, a digital art museum partnered with the New Museum in New York City, bridges these approaches. With an extensive collection of digital art, Rhizome employs EaaS to create environments for digital and "net" art to function.[45] Rather than focusing on individual art pieces, they emphasize batch preservation, attempting to decrease the amount of work necessary to complete the preservation of their large collection. This preservation model, which employs disk imaging and emulation of computing environments, currently provides a reasonable working model for both digital art and software collections. It allows for continued interaction with the software in a historically accurate interface and also provides a larger sense of the software's cultural and social context.[46]

---

[42] Guttman, Barbara. "The National Software Reference Library." Digital Preservation at the Library of Congress. 2013. Accessed November 12, 2015. http://www.digitalpreservation.gov/meetings/documents/preservingsoftware2013/Presoft_Pres-Guttman.pdf

[43] McDonough, Jerome. "A Tangled Web: Metadata and Problems in Game Preservation." In *Preserving Complex Digital Objects*, edited by Janet Delve. London: Facet, 2014.

[44] Dappert, Angela. "Metadata for Preserving Computing Environments." In *Preserving Complex Digital Objects*, edited by Janet Delve. London: Facet, 2014.

[45] Espenschied, Dragan. "Preserving 2000 Net Artworks: Practice Approaches to Object Identification, Risk Assessment and Permanent Access at Scale." Lecture, TechFocus III: Caring for Software-Based Art, Solomon R. Guggenheim Museum, New York, September 25, 2015.

[46] "EaaS: Image and Object Archive - Requirements, Implementation and Example Use-Cases." Open Preservation Foundation. Accessed November 12, 2015. http://openpreservation.org/blog/2014/07/23/eaas-image-and-object-archive-requirements-implementation-and-example-use-cases/

Two other approaches to use as a comparison are the Internet Archive's and San Francisco Museum of Modern Art's (SFMOMA). The Internet Archive makes large amount of material available through emulation, but the documentation is limited and description may be at the collection-level rather than the item level. SFMOMA, on the other hand, deals with a single piece of digital art in depth, and describes its many dependencies with in-depth documentation. While a video game can be considered a piece of digital art, and there is a certain amount of creative effort in all software production, the distinction between these two approaches lies less in an innate quality of the preserved object itself than in the curatorial manner in which that object is conceived and handled.

SFMOMA has done more than simply preserve the pieces in their collection. They have been working to create a documentation methodology to assist other institutions. This documentation, called the 'Technical Narrative' was recently discussed by Mark Hellar as a means of preserving new media art. New media art is complex and may not only consist of software components. As Hellar stated in his talk, "These artworks exist as multiple components and a complex system of behaviors."[47] In other words, new media art can be a large system of parts and the art itself is the way that those pieces behave together. Preserving the software that assists in running these behaviors is an important part of the conservation requirements.

The Technical Narrative, as Hellar outlines it, acts as guidance for art conservators to document the various parts and behaviors that constitute new media art. Documentation is an important part of software preservation, and understanding the different components that constitute a single piece of software is an important step to adequate software preservation. Hellar separates proper documentation into four parts: (1) a high level description of how the work operates; (2) a modular examination of the components, what they do and how they relate; (3) a detailed description of the artwork as it exists upon acquisition; (4) and an analysis of the current technology and an evaluation of its longevity. Outlining documentation in this way allows the new media art conservator to be more confident that they have documented all relevant aspects of the artwork that may affect its longevity.

With the assistance of this documentation process, SFMOMA was able to recreate "Agent Ruby," an early example of AI that was created by Lynn Hershman Leeson and commissioned by SFMOMA in 1999. [48] The public can view and interact with "Agent Ruby" online.[49] In order to make this piece of art accessible online through modern browsers, SFMOMA needed to update code and discuss any updates with Leeson. Although the piece was not adequately documented in 2008 at the time of official acquisition, the documentation process outlined by Hellar shaped the process of re-creating and then conserving the "Agent Ruby" piece. [50]

---

[47] Engel, Deena, and Mark Hellar. "Technical Narratives and Software-Based Artworks." Lecture, Technology Experiments in Art: Conserving Software-Based Artworks, National Portrait Gallery and Smithsonian American Art Museum, Donald W. Reynolds Center, Washington, DC, January 17, 2014.

[48] "Lynn Hershman Leeson, Agent Ruby, 1999-2002." SFMOMA. Accessed November 12, 2015. https://www.sfmoma.org/artwork/2008.230.

[49] Leeson, Lynn Hersman. "Agent Ruby." Accessed November 12, 2015. http://agentruby.sfmoma.org/

[50] Engel, Deena and Mark Hellar. "Technical narratives and Software-Based Artworks

Effectively, this documentation process helps outline the necessary knowledge to conserving software-based art over long periods of time.

A similar project related to digital art and software-based art documentation is Source Code Documentation for Computational Art at the Museum of Modern Art. Headed by Deena Engel of the Department of Computer Science at New York University, this project aims to create documentation for software-based art by adapting the software engineering community's approach to documentation. Engel, similar to Hellar, suggests a three part documentation process. Her suggested documentation can be outlined as follows: (1) Documentation and annotation in the source code; (2) Narratives; and (3) Visuals.[51]

Because these pieces of documentation are built from software engineering standards, they require some explanation. In source code, it is possible to include comments in-line with the operational code. These comments are not intended for the machine to read. Instead, they are used for developers to talk to each other within the code base and as bread crumbs for future developers describing what each piece of code is meant to do. Engel suggests using a similar tactic with digital art so that documentation becomes an intrinsic part of the code. "Narratives," Engel's second descriptive element, are similar to Hellar's "high level description." Narratives provide background on the software, what it is intended to do, and how it is intended to do it. Lastly, "Visuals" are used during software development to help ensure adequate communication between developers, the design team, and other stakeholders about the architecture, look and feel, and intent of the software. Visuals can include workflows, wireframes, and other types of diagrams.

Hellar and Engel's documentation methodologies are designed for digital art projects. They are too detailed to efficiently work for projects that have large collections or need to ingest large amounts of information. Instead, these documentation models are ideal for working with intrinsically valuable artifacts. This is not to say that these model could not be useful in a different situation, but it would need to be adapted to reconcile the outcomes with the time and resources restraints of the institutions with large collections or wider collection strategies.

Digital art preservation efforts are not limited to documentation concerns. The Rose Goldsen Archive of New Media Art at Cornell University works with a wide variety of digital art that sometimes relies on software-based technology in order to properly operate. The manner in which these materials are preserved varies according to the particular needs of the art piece. Here, there seems to be no push for a single preservation strategy, but rather ones that are tailored to individual artworks or collections within the archive.[52]

---

[51] In her talk, Engel separates visuals and UML diagrams, although she admits that these two categories are closely related, if not the same. For the sake of this paper and for the sake of simplicity, I have combined these two categories since UML diagrams do not necessary serve a separate purpose than other visuals in the development process.

[52] "Rose Goldsen Archive of New Media Art." Cornell University Library. Accessed November 12, 2015. http://goldsen.library.cornell.edu/

Lastly, there are several institutions that research how humans interact with new media and digital art, and these institutions can provide helpful background for software preservation projects. Groups like "Time-Based Media and Digital Media," "Matters in Media Art" and "The Pericles Project" conduct research on how artists, museum-goers, and curators interact with new media art, particularly when the art is transmitted and created in new ways. Similarly, "The Trope Tanks" at MIT work extensively on humanist studies of computer use and can provide important background on the history of computing, the history of digital art, and sociological studies of computing and the human experience.

Computation Reproducibility Efforts

Software preservation is a large concern in the STEM fields as well as in cultural heritage institutions. STEM fields, however, are concerned with software preservation as a way to ensure that their research can be re-computated in the future as a way to verify or support original findings. These motivations within the STEM field are largely concerned with ensuring access to data over long periods of time and being able to prove that data were created in proper ways. There are several projects that are addressing these issues in fields like computational chemistry, computational mathematics, and astrophysics.

Not every project deals explicitly with software preservation. The Astrophysics Source Code Library (ASCL), for example, acts as a repository more than as an archival or preservation project. Originally intended to help astrophysicists cite code that they use to complete the computations necessary for their scholarship, the ASCL has slowly developed into an institution interested in archiving the code as well as allowing scholars to share and cite the code they use. The manner in which they are archiving their code is still under debate. However, as an organization, ASCL recognizes the importance of archiving their code collection and is investigating how to best ensure that future astrophysicists will be able to access the code they need to in order to understand the scholarly data that they rely on.[53]

Similar projects in STEM fields concentrate on ensuring that all research projects can be re-created and validated in the future. Recomputation.org is working on ways to ensure that the work of computational mathematics and sciences can be validated in the future, regardless of the software those experiments relied on.[54] A blog post on Recomputation.org states the problem with some assistance from *The Hitchhikers Guide to the Galaxy*. To paraphrase, if you build a computer to tell you the answer to a question, you need to write down the answer and the question in order for the knowledge to remain useful in the future.[55] It is important to note that these projects are interested in preserving the computational ability of software, not the interface or the aesthetic qualities outside of the code. While contextual information remains important to understand how and why these computational abilities were authored, the concerns with social

[53] "Welcome to the ASCL." Astrophysics Source Code Library. Accessed November 12, 2015. http://ascl.net/

[54] "Recomputation.Org: If we can compute your experiment now, anyone can recompute it 20 years from now." Recomputation.org/ Accessed November 12, 2015. http://recomputation.org/

[55] "How Do We Know the Answer is Not 43?" Recomputation.org. Accessed November 12, 2013. http://recomputation.org/blog/2014/09/14/why-isnt-the-answer-43

and cultural relevance that one sees in digital art and software collection preservation projects are absent here.

Conclusion

Preserving software is an important aspect of an overall digital stewardship practice, but there are no agreed upon best practices for accomplishing the multi-faceted desires and goals within this emerging field. While it is always important to establish best practices, it is important to remember that software is created for many purposes. With a wide variety of disciplines interested in the subject, there is a large amount of expertise from which to pull, and any individual or institution interested in starting a software preservation program will need to customize a strategy to fit their exact needs, the needs of their users and the nature of their collection. Best practices will need to be flexible in order to reflect the many use cases for and types of software.

VMs, in particular, present an interesting resource, as they may be of use in many situations and can accommodate software with many different dependencies from many different time periods. However, regardless of tools, it is clear that documentation of the development and preservation process is a key aspect to all strategies. Whether an institution can afford in-depth, individual documentation, like that done for Agent Ruby at SFMOMA[56], knowing what is being preserved and how will be vital to future users and preservationists as they try to access executable content in the years to come.

As NLM continues to research and pursue its own software preservation strategy, we hope to show our work, our mishaps, and our successes, in order to ensure that the digital stewardship community as a whole can grow and begin to tackle software in an enlightened and thorough manner.

[56] Sterrett, Jill, and Mark Hellar. "Virtualizing Agent Ruby: Collecting Web Art." Lecture, 2010 Summit of Documentation and Conservation of the Media Arts Heritage, Universite Du Quebec a Montreal, Montreal, March 5, 2010.