Pace University
DigitalCommons@Pace

Wilson Center for Social Entrepreneurship | Academic, Policy and Research Centers

1-1-2012

500 Year Documentation

Francis T. Marchese, PhD

Maninder Pal Kaur Shergill

Follow this and additional works at: http://digitalcommons.pace.edu/wilson

 Part of the Business Administration, Management, and Operations Commons, Entrepreneurial and Small Business Operations Commons, and the Interactive Arts Commons

Recommended Citation
Marchese, PhD, Francis T. and Shergill, Maninder Pal Kaur, "500 Year Documentation" (2012). *Wilson Center for Social Entrepreneurship.* Paper 9.
http://digitalcommons.pace.edu/wilson/9

This Article is brought to you for free and open access by the Academic, Policy and Research Centers at DigitalCommons@Pace. It has been accepted for inclusion in Wilson Center for Social Entrepreneurship by an authorized administrator of DigitalCommons@Pace. For more information, please contact rracelis@pace.edu.

500 Year Documentation

Francis T. Marchese
Pace University
Computer Science Department
New York, NY 10038
1 212 346 1803
fmarchese@pace.edu

Maninder Pal Kaur Shergill
Pace University
Computer Science Department
New York, NY 10038
ms34286n@pace.edu

ABSTRACT
Museum visitors today can regularly view 500 year old art by Renaissance masters. Will visitors to museums 500 years in the future be able to see the work of digital artists from the early 21st century? This paper considers the real problem of conserving interactive digital artwork for museum installation in the far distant future by exploring the requirements for creating documentation that will support an artwork's adaptation to future technology. In effect, this documentation must survive as long as the artwork itself – effectively, in perpetuity. A proposal is made for the use of software engineering methodologies as solutions for designing this documentation.

Categories and Subject Descriptors
D.2.7 [**Software Engineering]:** Distribution, Maintenance, and Enhancement - d*ocumentation; restructuring, reverse engineering, and reengineering.*

General Terms
Documentation, Design, Management.

Keywords
Digital art, conservation, requirements engineering.

1. THE PROBLEM
Over the past decade cultural institutions have begun acquiring works by digital artists that have ranged in design from ephemeral performance to immersive installation. Conservators of digital media, whose job it is to preserve these works, are faced with the daunting task of managing a diversity of art so as to make any artwork displayable at any time in the future. The issues conservators face for maintaining a digital artwork's longevity are manifold. Digital artists employ a wide variety of contemporary computer languages, sometimes in combination, building upon a range of development libraries and environments, many of which may be either open source or of an artisanal nature. Software interfaces, formats, and protocols continue to evolve, and globally accessible resources either disappear or become redistributed. Finally, computer hardware is guaranteed to become obsolete.

In order to gain some sense of these issues consider the artwork *I Want You to Want Me* (2008), an interactive installation about online dating designed and built by Jonathan Harris and Sep Kamvar [1], commissioned by the Museum of Modern Art (MoMA) in New York and installed on Valentine's Day 2008 as part of its *Design and the Elastic Mind* show [2]. Displayed on a vertically mounted high definition 56" touch screen monitor, the artwork portrays a sky filled with hundreds of pink (female) and blue (male) balloons, each representing an individual's online dating profile that has been harvested and coalesced from several dozen Internet dating websites. Viewers can touch individual balloons to reveal personal information about the dater found inside, and can rearrange the balloons in various ways to highlight different aspects of the world of online dating, including the most popular first dates, top desires, self-descriptions, and interests (Figure 1).

I Want You to Want Me is based on a client-server architecture in which the client locally controls a graphics display, with the application backend housed in a California server farm. An URL server provides addresses to a web crawler which sends dating site information to an information extractor, the responsibility of which is to fill a database with information about individuals, approximately one million elements in size. Data from this database is accessed and passed to the front-end application, using an API that configures search strategies and queries. The programming languages and components used to build the artwork include C++, Java, PHP, OpenGL, and SQL. The servers run the UNIX operating system and the client runs Windows XP.

I Want You to Want Me presents challenges for its future installation. Its processing and databases are distributed over multiple computing platforms and locations. It requires several programming languages for its construction and execution. The nature and structure of its database, and the information mining algorithms employed to extract data are unknown. And it is unclear how tightly coupled the display system is to the underlying computer graphics software. Thus, assuming this artwork's current technical state, it is uncertain whether it would be a good candidate for display in the distant future, given its scarcity of documentation and the current state of contemporary conservation practice.

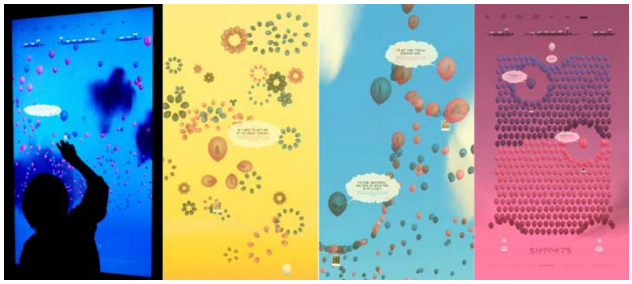

Figure 1: *I Want You to Want Me* (2008) by Jonathan Harris and Sep Kamvar

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DocEng'12, September 4–7, 2012, Paris, France.
Copyright 2012 ACM 978-1-4503-1116-8/12/09...$15.00.

2. CONSERVATION STRATEGIES

Digital art conservators have taken two approaches to preserving digital art: technology preservation and document compilation. In technology preservation computer technology is stockpiled to support the artwork in the inevitable case that a component fails. In document compilation an extended set of documentation is assembled to help define and contextualize the artwork with the express purpose of making the artwork displayable at some future date. Artist interviews, questionnaires [3], artist-conservator-curator collaborative discussions, conservation workshops [4], and documentation of a program's source code are all approaches that have been taken [5].

We believe the technology preservation approach to a long term conservation strategy for digital artwork is problematic for all but the most historically significant works. Museums and cultural institutions neither have the resources to stockpile computer parts, nor routinely maintain computer-based artworks to extend their lifespans. It must be remembered that museums collect far more artwork than they can exhibit at any given time. With the exception of works that either define the museum's collection or are critical to the art canon, all remaining art may be expected to rotate from storage into galleries pursuant to curatorial discretion. In such environments it may be decades before artworks are reinstalled. As a result, routine maintenance of these works becomes managerially prohibitive because of time, staffing, and financial constraints; leaving open the prospect that when an artwork is finally scheduled for installation it may not be possible to do so, because either part or all of the artwork will have reached technical obsolescence.

We believe as well that the best long term preservation strategy should be based on document compilation. Our hypothesis is that if an artwork can be transformed into an appropriate set of representations, then it will be possible to reinstantiate the artwork within future technology. Documentation that underlies this strategy must:

- support both abstract and detailed descriptions of static artwork structure and its dynamic processes.
- provide a diversity of representations to satisfy all stakeholders (e.g. artists, curators, and conservators).
- be sufficiently extensible to support organization, categorization, and systemization of digital art collections.
- be sufficiently flexible to sustain both individual document and corpora evolution.
- be integratable into art conservation practice.

The last item in the list is important because art conservation is a formal scientific activity defined by the International Council of Museums Committee for Conservation as the "the technical examination, preservation, and conservation-restoration of cultural property."[6] As such, any methodology instituted to augment traditional conservation practice must be suitably formal, mature, and rigorous to meet this profession's requirements, as well as the preceding four criteria.

These criteria may be met by adapting software engineering processes and practice.

3. ENGINEERING DOCUMENTATION

Software engineering provides a systematic methodology for creating and maintaining documentation to support communication, preservation of system and institutional memory, and processes such as system auditing. Within this context a computer system's documentation should supply comprehensive information about its capabilities, architecture, design details, features, and limitations. It should encompass the following five components [7]:

1. *Requirements* – The artwork's conceptual foundation. What it is supposed to do.
2. *Architecture/Design* – An overview of software that includes the software's relationship to its environment, and construction principles used in design of the software components.
3. *Technical* – Source code, algorithms, and interface documentation.
4. *End User* – Technical, installation, and user documentation.
5. *Supplementary Materials* – Anything else related to the system.

Each component is important to the representation of digital art. Each may operate at a different level of abstraction or within a particular context. *Requirements* documentation presents the conceptual view of what the system is expected to do. It is written to be understood by all the stakeholders who comprise an art museum's business practice: directors, curators, conservators, artists, installers, and maintainers. *Architecture/Design*, *Technical,* and *End* User documentation are of importance to conservators, installers, and maintainers.

4. DOCUMENTATING DIGITAL ART

When a museum acquires an artwork, it is the conservator's responsibility to acquire sufficient documentation from the artist to ensure its proper installation. Other documentation may exist, including: examples of previous installations that have been approved by the artist, design histories, interviews, catalogues, scholarly works, alternative installation plans, drawings, models, documentary videos, websites, etc. Beyond that, conservators may resort to additional interviews, collaborative discussions, and questionnaires to flesh out the artwork's character.

As a whole, this documentation may exhibit varying degrees of incompleteness, inhomogeneity, and diversity in its content and format. In order to make this artwork maintainable, this documentation must be transformed into formats that clearly define the artwork's nature, allow individual document components to be assigned to one or more of the five document categories specified in Section 3, and make the documentation maintainable for the long term.

We have taken initial steps to this end by designing a checklist/questionnaire within a spreadsheet. Checklists [8][9], templates, and patterns [10] are integral to software engineering practice, facilitating timely solution of analysis and design problems, as well as providing for formal verification and validation of product and process. Integrating such methodologies into digital art conservation practice should facilitate the creation of a strategic maintenance strategy for digital artwork, as well as a means for assessing a digital artwork's installation requirements at some future time.

Our objectives in designing the checklist/questionnaire are threefold:

1. To define as completely as possible the exact technological state of a digital artwork upon its acquisition by a museum, along with the technological milieu from which it originated. This baselines the artwork.

2. To track and assess an artwork's increasing divergence from state-of-the-art technology at some future time. Here the documentation will help conservators determine which technologies need updating for a future installation. It should also support risk analysis for determining the degree of effort (e.g. costs) required for future installation.
3. To offer a means for comparing the technological underpinnings among artworks within a digital art genre and throughout digital art history. This latter point is important, because it will give the digital conservator a sense of the degree of technological heterogeneity within the museum's collection and provide information necessary for creating strategic plans for maintaining the digital art collection as a whole.

Our checklist/questionnaire's current version contains a large number of questions that we expect to expand with time. The questions shown herein have been condensed from the questions found in the spreadsheet and are provided to offer a sense of its breadth. It currently contains the following categories:

- **Algorithms Used**: What is the form of any special algorithm employed? What is its relationship to any standard algorithm?
- **Application Software Requirements**: What were the development languages, libraries, and interfaces used, and their versions? What development tools were employed? Is source code provided?
- **Authorship**: Who contributed to the design and building of the artwork and its parts? Who or what were the artistic or technical influences of the artwork?
- **General**: Questions related to administration of the artwork. What are the preservation priorities for this artwork? What are the preservation strategies to be used? How and where will the artwork be stored and accessed? What are the environmental constraints? May the artwork be reengineered?
- **Hardware Requirements**: Questions related to the complete description of the hardware substrate including: motherboard, CPU, RAM, video card, network adaptors, BIOS, system timings and interrupts, display devices and resolutions, etc.
- **Installation Requirements**: Questions related to the artwork's installation, including sources of materials, handling instructions, construction, etc.
- **Interview Questions**: What questions were asked of the artist and the answers?
- **Media Requirements (Audio / Video)**: What are the file formats, image and video resolutions, timings, codecs, and compression schemes used for various media?
- **Networking and Communication Requirements**: What is the network topology employed? What are the network protocols used by the artwork? What are the networking and communication services required?
- **Preservation Strategies**: What preservation strategies are expected to be used for this artwork?
- **Quality Assurance Procedures**: How will reliability issues be addressed? Does the artwork exhibit instability issues? Under what circumstances does the artwork fail? How is the faulty artwork to be maintained and tested?
- **References**: References related to the artwork itself, its artistic milieu, etc. This may include documents or links to resources from scholarly journals to videos.
- **Rights and Permissions**: What are the legal constrains placed on this artwork? What are the licensing terms for the use of media, software libraries, etc.?
- **Security Requirements**: Does the artwork have built-in security components? Is a firewall required? Does the artwork contain hidden files? Are encryption schemes uses for any files?
- **Supporting Documentation**: Technical, maintenance, and owner's manuals for hardware and software. Design histories, books and catalogs, alternative installation plans, drawings, models, documentary videos, websites, etc.
- **System Constraints:** What are the budgetary, architectural, technical, staffing, and scheduling constraints related to this artwork? What are the risks related to its installation and maintenance?
- **System Software Requirements**: What operating systems, systems libraries, etc. are required?
- **Web Requirements**: What web protocols are required (e.g. http, ftp, etc.). What web servers are required? What data formats (HTML, XML, style sheets) are required? What browsers and their version are required?

These questions should afford sufficient information to fulfill three documentation categories given in Section 3: *Requirements, End User,* and *Supplementary Materials*. If the artwork's software configuration contains source code, then the *Technical Documentation* category requirements would be fulfilled as well.

Architecture/Design documentation contributes significantly to a total understanding of the artwork system, making it possible in principle to recreate all or part of the artwork as required. Because these representations are intended to communicate what the artwork is supposed to do, as opposed to how it is supposed to do it, they are designed to articulate the system's high-level static and dynamical designs and interfaces, while suppressing the low-level implementation details. Three kinds of representations are considered here: UML use-case scenarios that define an artwork's functional requirements, class diagrams that fix an artwork's static structure, and sequence diagrams that convey an artwork's dynamics.

Most artists are not trained as software engineers, and thus cannot be expected to create UML representations of their works. However, it may be possible to extract use-case scenarios for an artwork from its "supporting documentation" where its temporal designs may have been elaborated as storyboards and alike. Otherwise, use-cases may be captured by observing and interrogating the artwork running within a gallery setting. Class and sequence diagrams are another matter. Given a complete set of use-cases, and utilizing the remaining documentation categories as the interpretive context, it should be possible to generate these UML representations – in effect, create a complete design document for the artwork from scratch [7]. Although this would be a time consuming process, and it may generate a design that does not represent the artist's original architecture, the design should fulfill the artwork's functional requirements. Alternatively, in circumstances where the artwork has been written in a popular programming language such as C, C++, or Java, UML class and sequence diagrams may be generated automatically from either source or executable code exploiting mainstream UML CASE

tools [11]. The advantages of this transformation are that the resulting UML diagrams embody the artist's original software architecture, and the time required for creating these architectural designs becomes negligible.

Finally, *Architecture/Design* documentation offers a distinct advantage over source code. For example, if five hundred years from now the programming language employed by today's artist to create a digital artwork has disappeared into history, its source code upon which it is based becomes virtually useless. Although emulators [12] and virtualization technologies [13] may evolve to completely bypass source code issues by creating environments to allow an artwork's executable code to run as is, there is no way of predicting whether a future virtualized environment would be able to support part or all of an artwork from 500 years in the past. In contrast, *Architecture/Design* documentation provides a pathway for rebuilding part or all of the artwork, beginning from the artwork's high level requirements and designs, working down, refining implementation details in any language to suit.

5. SUMMARY

In this paper we have begun to consider issues involved in designing and maintaining documentation that will be expected to evolve in perpetuity, by analyzing the real problem of conserving digital artwork so that it may be installed in a museum in the far distant future. The approach we have taken is to employ a software engineering methodology to documentation that focuses on five classes of documents: *Requirements, Architecture/Design, Technical, End User,* and *Supplementary Materials* to set a systematic framework for capturing and organizing all materials related to a digital artwork. As part of this process we have created an extensive checklist/questionnaire, the objectives of which are to define the exact technological state of a digital artwork when it is acquired by a museum, and track the artwork's deviation from up-to-date technology over time. Finally, we have put forward a procedure for capturing the artwork's architectural design using software engineering's Unified Process model, and have proposed how this model could be applied to recreate the artwork in the distant future.

6. FINAL THOUGHTS

This research represents our initial foray into designing documentation for the long term preservation of digital artwork which, in essence, is a unique variation on a legacy system. Unlike a traditional legacy system, which is expected to be updated or overhauled at some point in time so as to capitalize on advances in state-of-the-art technology to improve its core attributes such as usability, speed, and performance; legacy digital artwork will become reliant on state-of-the-art technology to ensure that it functions identically to the first day it had been installed in a museum. To deal with this dichotomy of technological purpose, documentation will need to meet the objectives put forward in Section 4, as well as be able to characterize a digital artwork from its functional requirements through its technical details. In so doing, it will provide important information for adapting new technology to old art, by helping locate sources and kinds of incompatibilities that have evolved in technologies over time. To this end, we are exploring the expansion of our spreadsheet checklist into a database system, and categorizing a set of artworks with respect to their technological evolution.

7. ACKNOWLEDGMENTS

We thank those individuals who have contributed to the content of this checklist: Juan Amadiz, Eric Greene, Ingrid Grey, Samuel Martin, Jeffrey Pecan, Anthony Perrone, and Vishnu Sulapu.

F.T.M would like to thank the Helene & Grant Wilson Center for Social Entrepreneurship at Pace University for a fellowship supporting this work.

8. REFERENCES

[1] Harris, J. and Kamvar, S. 2008. *I Want You to Want Me*. Retrieved May 23, 2012 from http://iwantyoutowantme.org/

[2] *Design and the Elastic Mind*. 2008. Museum of Modern Art (MoMA). Retrieved May 23, 2012 from http://www.moma.org/interactives/exhibitions/2008/elasticmind/

[3] Ippolito, J. 2003. Accommodating the unpredictable: The variable media questionnaire. In *Permanence Through Change: The Variable Media Approach*, (Guggenheim Museum Publications and The Daniel Langlois Foundation for Art, Science, and Technology).

[4] ERPANET. 2004. The archiving and preservation of born-digital art workshop. *Briefing Paper for the ERPANET Workshop on Preservation of Digital Art*. Retrieved May 23, 2012 from http://www.erpanet.org/events/2004/glasgowart/briefingpaper.pdf.

[5] Yeung, T.A., Carpendale, S. and Greenberg, S. 2008. Preservation of art in the digital realm. *The Proceedings of iPRES2008: The Fifth International Conference on Digital Preservation*. British Library, London.

[6] International Council of Museums Committee for Conservation. The conservator-restorer: a definition of the profession, section 2.1. Retrieved May 23, 2012 from www.icom-cc.org/47/

[7] Larman, C. 2005. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd Ed. Prentice-Hall, New Jersey.

[8] Pressman, R. S. 2001. Adaptable process model software engineering checklists: Reviewing OOA and OOD models. Retrieved May 23, 2012 from http://www.rspa.com/checklists/ooadmods.html

[9] Pressman, R. S. 2001. Adaptable process model software engineering checklists: conducting and reviewing the software design model. Retrieved May 23, 2012 from http://www.rspa.com/checklists/designmodel.html

[10] Riehle D. and Züllighoven, H. 1996. Understanding and using patterns in software development. In *Theory and Practice of Object Systems* 2, 1, 3-13.

[11] Khaled, L. 2009. A comparison between UML tools. In *Second International Conference on Environmental and Computer Science*, 111-114.

[12] van der Hoeven, J., Lohman, B. and Verdegem, R. 2007. Emulation for digital preservation in practice: the results. *The International Journal of Digital Curation* 2, 2, 123-132.

[13] Crosby, S. and Brown, D. 2006. The virtualization reality. *Queue* 4, 10 (December 2006), 34-41.