A Preservation Framework for Development of Digital Repositories

Babak Hamidzadeh

University of Maryland; College Park, MD 20742

Abstract

This paper specifies a preservation framework in which the semantics and parameters of digital preservation are discussed. Using this framework, the paper then proposes a methodology and a high-level breakdown of tasks for the development of digital repositories that aim to address digital preservation as part of their features and functionalities. The task breakdown enables analysis of feasibility of development of each task and the suitability of various development methodologies for each task. The preservation framework also facilitates decision making regarding automation and general-purpose nature of software that is developed for digital preservation of various digital content types and formats.

Introduction

Digital preservation is a nascent field and development of effective and scalable preservation tools and systems remains a challenge. Preservation requirements have not fully been addressed in the community.

Digital repository development has been underway for several years. Several of these repository platforms [1],[2],[3],[4],[5] have been deployed with varying degrees of success in managing different types of digital content. Many different open questions remain, however, in the development of digital repositories, which are subject of active research and development. Digital repository development, in general, has faced difficult software engineering challenges. One such challenge has been the design and development of general-purpose systems that can handle all content types and formats. These repositories can also be a natural platform where digital preservation requirements are addressed.

Traditionally, repository development has been done in a content-specific way. That is, for each set of like materials a custom-made system is built that meets the specific business, functional and technical requirements of that content set. This type of development leads to siloed systems (See Figure 1) that perform differently and do not share information efficiently and effectively. It has been argued that much of what goes into each system too is redundant and repeated in other systems. Development costs are also high as designing and implementing each new system starts from scratch, though not strictly.

These shortcomings of the content-specific repositories have been the impetus behind large and costly initiatives [6],[7] to develop single, general-purpose systems (See Figure 2) that manage preservation and other lifecycle phases of all content types, no matter how heterogeneous. These efforts have proven to have problems of their own. Long and complex requirements gathering exercises led to very large set of requirements. Designing and building one system that meets all requirements became costly and unmanageable. Part of the problem was that gathering all requirements in detail was difficult, as some requirements were not known. Some content types with new and some times conflicting requirements from other content, emerged after the requirements were collected. Designing and building one system for very large sets of requirements is a difficult task. Modifying and adding features to such systems also pose challenges for post-implementation and post-deployment phases of a repository system of this type. We argue, as approximated in Figure 3, that building general-purpose tools becomes rapidly complex and ineffective as heterogeneity between the types of content those tools are to mange, increases.

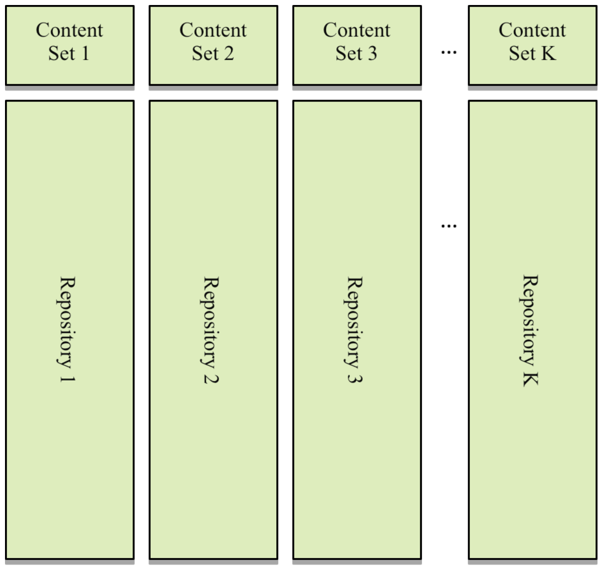

Figure 1. Content-Specific software development

In this paper we address development of scalable digital repositories. The research and experience discussed in this paper was motivated by the need to design digital preservation repositories with effective software development methodologies, in order to meet the requirements of various digital content types and formats and to be able to manage large-scale digital content in all phases of their lifecycle. Given the advantages and disadvantages of each of the content-specific and general-purpose approaches to repository development, we propose an approach that aims at building general-purpose subsystems that can serve all content types while leaving other parts to a more content-specific approach. We divide the boundary between our general-purpose subsystem and the content-specific parts along semantics of digital preservation. These semantics are centered around the definition of digital objects and features thereof that are important for preservation.

As a first step to developing digital preservation requirements, we define the semantics of preservation as a service or functionality. Such semantics state what we mean by preservation. They also specify various characteristics and qualities of a digital object, and state which of those characteristics must be maintained, to preserve an object over time. Based on these preservation semantics, we discuss challenges and approaches to developing software for long-term preservation of large-scale digital objects.

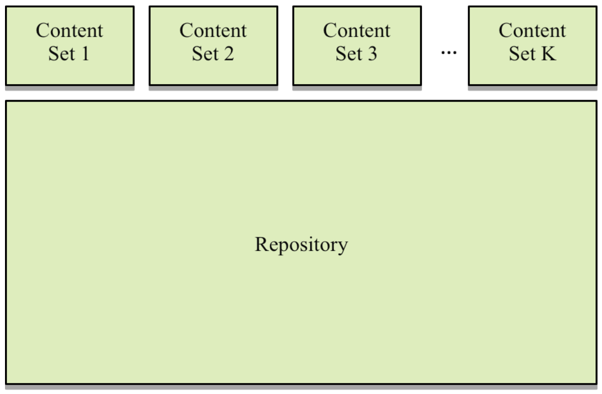

Figure 2. General-Purpose software development

This paper takes an authenticity-centric approach to preservation. We argue that maintaining authenticity over time, if not the most important, is one of the most important characteristics or qualities in preserving digital objects and their aggregations. To maintain authenticity of digital objects or their aggregations, attributes of the objects that reside outside what is considered their intellectual contents, such as author, date of creation, and others, may also have to be preserved and must reliably and permanently remain associated with and linked to the object, over time.

We also look at digital preservation of large-scale content from a cost point of view. From that point of view, developing software to automate preservation tasks is a cost-saving effort. Using authenticity as the central quality to preserve, and regarding software development as a cost-saving endeavor, we develop our framework and our development methodology.

Using our preservation framework, we develop a high-level architecture that classifies various aspects of developing preservation repository software. We identify the parts of the architecture that lend themselves well to developing generic and general-purpose software, and parts for which, we conclude, developing generic software that works for all digital content types and formats is infeasible. Using this classification, we have devised development methodologies to maintain a cost-effective approach to developing preservation repositories for various large-scale digital content types and formats.

Digital Object

Digital objects are often drastically different from their analog counterparts in format, structure and representation. Digital objects may have a complex, hierarchical or other structure in that they may consist of other digital objects or be interrelated with other objects. One digital object may be part of multiple structures.

What constitutes and represents a document in the analog world with one piece of paper, for example, may consist of a multitude of files each containing a piece of the document, or each of which be a document in itself. Some of the files constituting an object may contain pictures and drawings. Others may contain an audio or video segment embedded in the document. A webpage is an example of such a document. A simple photograph may be tiled and each tile may be represented and stored in a separate file. When opening or rendering the photograph, each tile must be opened and placed in the right space within the frame of the photograph on the computer screen. What will look like a plot or graph on the screen, may be represented by a table of data in a file, which is transformed into a graph upon opening the document, using a software application.

As you can see from the above examples, digital objects have two main distinguishing characteristics. One is that they have a structure (i.e. set of files or other objects and their interrelationships) that will need to be modeled and represented in a way that the computer understands and can manage. We refer to this as the content model or the logical model of the digital object(s). Without this map or representation of the object and its relationships with other objects, it will be difficult to reconstitute the object using its subobjects or the files that store pieces of the object.

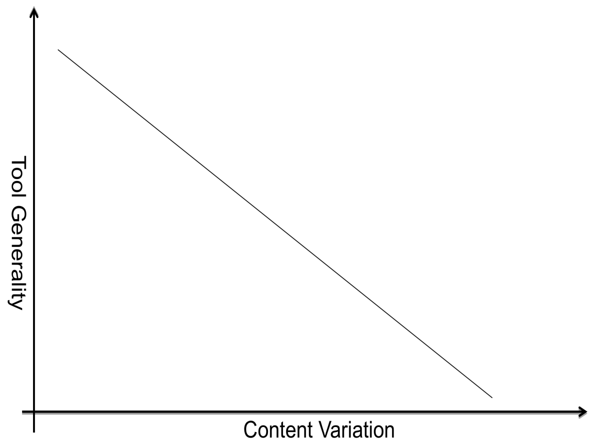

Figure 3. Development complexity of general-purpose systems

The other distinguishing characteristic of digital objects is that they need a software application to make them presentable (e.g. viewable, readable, audible) and useable. This is in addition to the hardware that presents the object and the hardware that stores files that constitute the object. A digital object has to have other attributes that are important in preserving it in a useable way, over time. Each object has to have an identifier that ensures the object is uniquely specified and distinguished from other objects. Each object is also assumed to have intellectual content. That is the content that conveys informational message to the people or systems to whom the object is presented, or with whom the object interacts.

Descriptions (also referred to as metadata) are zero or more facts and attributes about the object, each defined and labeled by a name, unique among that object's descriptions. The author or creator of a digital object, the date of creation, type of object in a classification scheme, are examples of such descriptions. An object's descriptions can be designated such that they are inherited by and applied to the object's components and to other objects contained in it or otherwise related to it.

A digital object can also have behaviors associated with it. Interactive objects can respond and behave in various ways to input and signals from various sources. A simulation system is a complex, interactive system that can react and respond dynamically to the way people interact with it. The responses can be vastly different from one another. Simulators may even be designed to give different and random responses to the same sequence(s) of input. The next section, discusses high-level preservation requirements of digital objects.

Preservation Framework

Identity

An object's identity plays an important role in preserving the object. If a digital object cannot be uniquely and persistently distinguished from other objects, its preservation will naturally be in question. Therefore, every digital object must be identified by at least one name, unique to that object, which may be used to directly and synonymously refer to the object and its elements by other objects, systems and persons. The identity of a digital object must be persistent over time, or persistently maintained over time.

As mentioned earlier, descriptive elements or attributes of the object also have identities that must be managed by the same requirements as the ones for the object itself. Uniqueness, for example, is a requirement for the descriptive elements and attributes of the object, within that object. In other words, outside each object, the descriptive elements do not have to be unique, as they can be uniquely referred to and managed in association with the object's identity.

Integrity

To preserve an object's integrity is to ensure that the object remains fixed, complete, accessible, and readable. We will discuss these requirements next. An object's fixity refers to keeping unchanged, that object, its components, relationships between its components and other attributes of the object, over time. Completeness refers to availability and intactness of all of that object's components, descriptions and the components' contents, which make up the whole of the object's intellectual content and its behaviors, the way they were meant to be at any point in time, by the author of that object at any time. Maintaining completeness over time also requires that relationships between the object of preservation and other digital objects are captured and managed over time. Accessibility refers to ensuring that the object and all its parts and attributes are locatable in a file and storage system, and that they are retrievable as appropriate and needed for rendering and presenting the object.

Readability of an object is the ability to render the object the way it was intended to be presented to various groups of users, by its author(s), at any time. In the digital world, to be readable, an object must open using a computer software application that is intended to open and execute the object. Example actions to open digital objects on personal computers is to double-click on an icon representing that object on your desktop screen, or to render it via a URL in your Web browser, or to open it from within the rendering software application software itself. In order to remain readable over time, all components of an object, in addition to being known, relocatable, and accessible, must also reconstitute the object accurately and present the whole of the object appropriately, as their author intended.

It is important to note that digital objects may have various levels and modes of presentation to various users. A partly classified document, may be presentable in full to a person with the right roles and credentials, but be redacted and partially presented to another person with lower credentials and different roles. As discussed above, digital objects may have various functionalities and behaviors associated with them. If such functions and behaviors exist, maintaining the accurate behavior and execution of the functions in response to various sets and sequences of input is another requirement for preserving the integrity of a digital object.

Understandability

Once a digital object is rendered accurately and presented the way it was meant, the object is still to be understood by the person or system it is presented to. In the case of a dataset representing a geographic map, for example, understandability refers to the ability to comprehend an object's intellectual content and to knowing how to interact with the object and its components as originally intended. In the case of the geographic map data, understandability amounts to viewing and understanding the legend and knowing the message that the map conveys by understanding what the lines, symbols, labels and colors mean. Understandability also requires instructions that enable the user to zoom and pan the map. Understandability of a document depends on existence of several qualities and characteristics. These characteristics include intelligibility and usability. As in the integrity requirements, elements of understandability are interrelated.

Intelligibility of a document refers to the ability of a viewer or user of the document to comprehend the intellectual content of the document. This quality is distinct from readability in that a document may be rendered correctly but the person it is presented to may, despite seeing, hearing, and sensing the document, still not understand its contents. A textual document, for example, will need to be intelligible to the reader. Intelligibility in this case may require that the reader understands the language and the lingo used in the textual document.

Usability of a document refers to the ability of a user or viewer of the document to take necessary actions upon reading or viewing or sensing the document, if required, or perceive some impressions, or draw necessary conclusions and things of that nature, as intended by the author(s) of the document. If there are functionalities associated with the document, the person interacting with the document needs to know the functions, the way to invoke them and to interpret their outcomes. If an electronic form has a number of buttons labeled "review," "submit," "save," "sign," or "dispose," the reader of the document will need to understand what each button does and is used for, and the circumstances and sequences each must be pushed or clicked to obtain the desired outcome.

It is important to note that an object's use or usability may vary based on the roles and responsibilities of various persons or entities the document is presented to. The intent for the creation and dissemination of the object is an important aspect of the document that, if captured and recorded, can increase its understandability. The type of a document (e.g. Memorandum, Announcement) can carry some information about the intent of the document. Just as the intellectual content and use may vary from one person or entity to another, the intent of the document and its communication may vary from one person or entity to another.

System Development Approach

A major question that has preoccupied many archivists and software engineers is whether tools can be built to maintain all of identity, integrity and understandability requirements of digital objects over time. It should be evident that automating management of some requirements such as fixity, identity, completeness and access of digital objects is easier than maintaining and managing their readability and intelligibility.

Consider, for example, a digital land and sea map object that resides in a file in a computing environment. It is conceivable that, using check summing and sound storage management and inventory techniques, we can develop software that ensures the fixity of the object over time by assigning checksums to the map object file, storing multiple copies of the file, checking the checksums on a schedule, and recovering a file from archival storage when the file's fixity is compromised. Developing software to maintain intelligibility of the digital map object, on the other hand, is considerably more difficult. The complexity comes from enabling the computer to understand the semantics and meaning of various lines, shapes, symbols and colors on the map, so it can ensure that if and when opened and presented on a display, the map appears and, more importantly, is understood the way it was intended, possibly many years earlier.

Another, just as important, question is whether general-purpose tools can be built to perform preservation actions for data in any and all digital media and formats. This is to keep the software development and maintenance costs down. Again, some aspects of preservation such as maintaining and managing fixity may, and we believe they do, lend themselves well to general software development. Given the difficulty of developing customized software for managing other preservation requirements, such as intelligibility, for any and each particular digital media and format, it should be evident that developing general-purpose software that manages these aspects of preservation for any and all media types, formats and subjects becomes particularly difficult. Given today's knowledge representation and semantic technologies, this does not seem readily feasible and scalable, and makes for a challenging area of research and development.

As a strategy and approach for development of a digital preservation repository, we can divide the overall system development work into two main categories (See Figure 4). One category, to which we refer to as bit management, addresses mainly issues related to maintaining fixity, accessibility and completeness of digital data at the file level. In this level of preservation, the unit of information is a file. That means, in preserving digital content at this level we neither address units of information at finer levels of granularity than files, nor will we address preserving the semantics (i.e. readability and understandability) of digital data. Another way of stating this is the system's attention goes to ensuring that the bits that make up a digital file and their order will remain intact, complete, fixed and accessible over time, without the need to know or requiring to preserve the readability, understandability and other semantic aspects of the data.

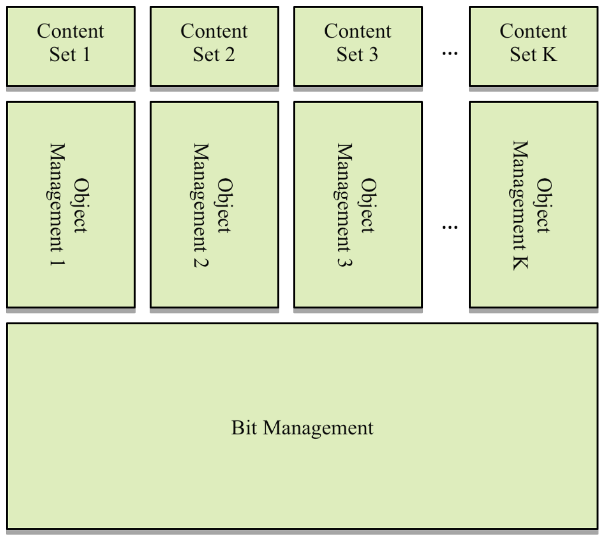

Figure 4. Hybrid software development approach

This separation of concerns compartmentalizes the simpler and more technically feasible system development tasks into the bit-management level. This approach facilitates development of scalable systems that can also address preservation, albeit at a lower level of service, at a general level that is applicable to all digital media, such as text, image, video and audio and formats, such as books, periodicals, manuscripts, photographs, music, speech, etc.

We refer to the more complex task of developing systems that address preservation of digital objects at the semantic level as semantic management or object management. Semantic management addresses maintaining readability and understandability of digital objects over time. The unit of information at this level is a digital object. A digital object, as discussed before, may consist of one or more interrelated files and contains intellectual content that has some meaning and intent to the entity it is presented to upon access. As we pointed out earlier, developing general-purpose software systems for this level of preservation, which can maintain readability and understandability of objects over time, is particularly complex. A main reason is that at the semantic level, the computer system will need to "understand" the meaning of the intellectual content of a digital object and its relationships with other objects. If the object is a photograph depicting a scene with various natural and physical objects represented in it, it will be difficult for a computer software application to understand the physical objects in the photograph and to understand the scene depicted in the photograph. Therefore, building software to maintain semantic aspects of digital objects in a particular media and format and even a general topic or theme is difficult enough and not always economically and technically feasible. Building such applications in a general-purpose manner to handle all media types, formats and themes becomes intractable.

Given the above level of complexities in building object management systems, our approach will need to be different from that of our bit management systems. To understand objects and their relationships, we will need to build models that represent knowledge at various levels, such as mapping of files to objects, relationships between physical and intellectual objects represented in the intellectual content of objects and between objects and aggregations thereof. It is building these models that makes object management complex. Our approach so far has been to build custom-made, content-specific models for digital objects of specific collections in specific media, usually a single media and not multimedia, for a few file formats within a specific media type and for a specific format, such as newspapers, manuscripts or books of few volumes.

Conclusions

We conclude that dealing with the large scale of digital content and their diverse types and formats is an essential business case behind developing preservation digital repositories, in order to manage preservation costs through automation. There is natural tendency to address scale and costs through building generic and general-purpose software that can handle all digital content format types and formats. Building such general-purpose systems, however, has proven difficult. We have developed a framework to address this challenge, which divides the general software development activity into two parts. One part has a general-purpose architecture to manage, at the file level and bit level without understanding digital object semantics, all content types in one system. The other part develops, in a content-specific manner, customized software to manage semantics of digital objects over time. Using this framework for software development enables us to manage the prohibitive cost of making customized and special-purpose software for each digital collection and each digital content type.

References

[1] http://www.fedora-commons.org/about
[2] http://www.dspace.org/introducing
[3] http://projecthydra.org/about-hydra-2/
[4] http://islandora.ca/about
[5] http://www.greenstone.org/
[6] http://www.archives.gov/era/
[7] http://www.gpo.gov/fdsys/

Author Biography

Babak Hamidzadeh received a Ph.D. degree in Computer Science from The University of Minnesota in 1993. From 1993 to 2002, he held assistant and associate professor positions in computer science and computer engineering at The Hong Kong University of Science and Technology and at The University of British Columbia, Canada. From 2002 to 2004 he was The Senior Manager of Information Management and Collaborative Technologies Research at The Boeing Company. From 2004 to 2011 he was the Director of the Repository Development Center at the Library of Congress. Currently, he is the Associate Dean for Digital Systems and Stewardship at the University of Maryland Librarie