The Web-at-Risk:
A Distributed Approach to Preserving our Nation's Political Cultural Heritage

CDL's Web-at-Risk Service Vision

DRAFT of 28 April 2005

Prepared by:
John Kunze
Patricia Cruse
Tracy Seneca
Web-at-Risk Project
California Digital Library

The Web-at-Risk: Service Vision

Contents:

Overview

For the Web-at-Risk project, the California Digital Library (CDL) and its project partners, with funding from the Library of Congress, are creating tools to enable curators to build, manage, and expose collections of materials harvested from the World-Wide Web. In fulfillment of the Web-at-Risk service vision, the tools will be developed incrementally and in close consultation with the curators who will be using them. This document describes that web archiving service vision from the ground up.

Introduction to Web Archiving

Web archiving is the process by which web-published materials are acquired, curated, and preserved. *Acquisition* is a set of activities centered around content capture, and preceded by organizational deliberation about general collection goals and specific site selection priorities. Acquisition is not considered complete until the captured material has been reviewed for quality assurance and copyright clearance.

Curation is the set of activities concerned with managing and describing the archived content, as well as making it searchable and browseable as required by the curator community. *Preservation*, the third area under web archiving, is the set of activities aimed at safeguarding the viability (intact bit streams), renderability (processable by computers), and understandability (to human beings) of the captured content. This kind of future-proofing is simultaneously the most important and least understood aspect of web archiving.

Web archiving was once a difficult and expensive undertaking. The institutional tools that we are creating, however, coupled with falling storage costs, will make it much easier for curators to build their own collections in a relatively self-service manner. They will include user interfaces that allow curators to initiate and monitor crawls, and to review, edit, and describe crawled content. Besides the project partners, the tools will be open-sourced and, when mature, distributed so that the entire digital library community can take advantage of them. With centralized, cooperating institutional repositories, there is less need for redundant collecting and for departmental repositories. Having said that, some specific, planned redundancy (replication) forms a core part of our preservation strategy.

Web Archiving Challenges

There are unique challenges for curators to bear in mind as they undertake web archiving. First, web crawling can be a resource-intensive process that takes time, consumes local storage, burdens remote web servers, and clogs networks. Before initiating what might be a heavy crawl, it may be useful to perform some lightweight “draft” crawls, the results of which you expect to analyze and then throw away. After crawl parameters have been refined over several drafts, the curator may initiate a real crawl and, if the results are worthy enough, decide to keep it for the long term.

Second, the sheer quantity of information that people are interested in harvesting from the web implies a heavy reliance on automation for all aspects of web archiving. Unless the captured site is small and static enough to verify each captured page by hand, quality assurance will be imperfectly performed using sampling, guesswork, and establishing a threshold of confidence ("good enough") that the desired information was captured completely and accurately. For large-scale crawling, the community is using the WARC/ARC (Web ARChive) file type to permit many retrieved web "pages" to be stored one-after-the other in one large file, thereby significantly reducing the number of files that need to be created. CDL is involved in the design of the WARC file.

Another challenge is coping with the wide diversity of formats and levels of format compliance in the harvested material. Non-standard uses and errors are very common, and this undermines a mainstream preservation strategy that calls for obtaining certain knowledge of the original source format in order to be in strong position in case a format conversion (called a *migration*) is required in the future. One possible mitigating strategy is to create and archive what is essentially a screen-capture (a *raster* file) of the material rendered with contemporary software. Although this would take up more storage, it would retain a document image (or a feasible two-dimensional projection of the material if it is not a document) that is as faithful as possible to the originally published form and that may one day be useful as a backup format should the original fail.

Versions present a major challenge in storing, searching, and browsing digital content. For web archiving in particular, version problems can become exacerbated when curators, by crawling a site again and again, end up capturing many copies of material that has changed not at all or very little. Because of the ease with which content can be gathered and the difficulty of eliminating duplicates and near-duplicates, the practice of re-crawling can waste storage and clutter search results. Some of our tool discussion and development efforts will be devoted to clarifying and containing these problems.

In general, it is not obvious how to curate and make available for browsing and searching a set of crawls representing snapshots of one or more sites over a period of time. The Wayback Machine at the Internet Archive (IA) illustrates a way to support browsing via a "time-travel" metaphor that may require adjustment when applied to the focused crawls conducted for this project. Part of the problem is that instead of one internet-wide crawling regime, this project will create collections that reflect crawling results of many different curatorial styles. Unlike the IA, it will be important for this project's archive to allow views restricted by curator, by time, by domain, etc. Another problem facing us (and the IA) is how to prevent a user's archival browse session from leaving the confines of the archive without warning.

In the context of successive re-crawling of a set of sites, we think of each crawl as a plane, each point of which represents content captured during that crawl. Over time, the next re-crawl is seen to create a new plane of crawled content that is laid on top of the previous plane. The end result is a stack of planes representing successive crawls of the same sites, the newest crawl on top, as in Figure N.

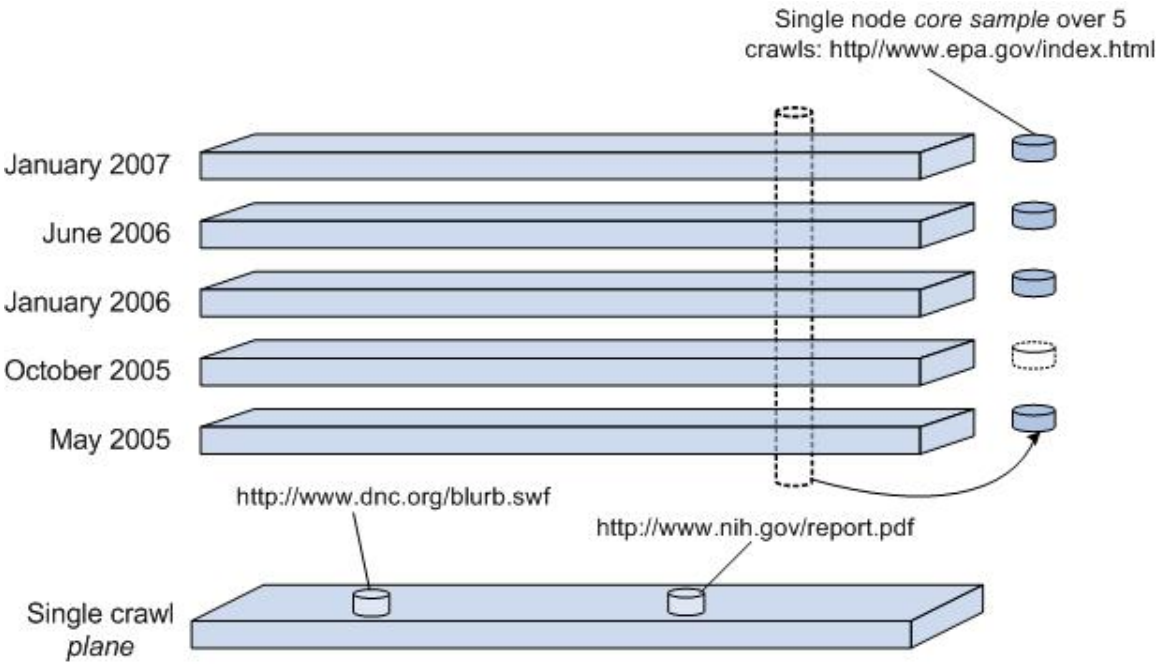

Fig. N Modeling Multiple Crawls: Planes and Cores

Within that stack, the same URL will often appear many times as a kind of vertical "core sample" in the stack, the oldest version on the bottom. Browsing and searching functions will include the discovery of a single URL result anywhere in the stack, and from there, may also offer the exploration of the containing plane (navigation within the same crawl) or the containing "core sample" (visiting newer or older versions of the URL's content).

Collections are Built by Curators

Although CDL creates the tools, responsibility for building collections is distributed to the curatorial partners. All selection, capture, and archive management decisions are made by the curators. The following kinds of operations will be supported.

- ❑Crawl content (register candidate sites and initiate web crawls)
- ❑Monitor crawl progress (estimate completion time, tune parameters)
- ❑Assess crawl quality (confidence rating)
- ❑Manage and describe collections, crawls, and harvested nodes
- ❑Search and browse crawled content (build indexes, navigate site mirrors)
- ❑Preserve content (organize into AIPs and replicate)

Principles of Tool Set Construction

The components of the web-archiving service will be compatible with the CDL Common Framework, which is a set of subsystems and interfaces that comprise CDL's technical infrastructure. This is a major leverage point, as CDL has too many services not to look for every opportunity to re-use existing, generalized technical components. Components built for this project are likewise expected to be re-usable.

The tools built will be open-source so that the archiving community can benefit from this publicly funded work, and our work can improve based on community feedback (e.g., usability reports, bug fixes). Tools will also be extensible, modular, and configurable so that our Web-at-Risk partners may be able to run and adapt the tools within their local repository environments.

The tools must also be compatible with broader, external web-archiving efforts. For archives to be able to exchange data, avoid redundancy, and conserve limited resources, it is important to use common storage formats and metadata standards.

Where possible these built components should be compatible with or augment other CDL efforts and services. An example is OAI (Open Archives Initiative) metadata harvesting, which has many similarities to web harvesting; in both cases, automated background processes remotely capture data that results in the creation, validation, and storage of well-formed information objects.

The development of this web-archiving service is a practical undertaking, but at the same time it is a "proof-of-concept" project. The initial product will not be perfectly polished; but will be refined through an iterative process based on feedback from curators. The most important practical concern will be to support and augment the workflows currently used by curators who are actively building collections.

Web Archive Collection Model

A *collection* consists of one or more crawls of related content. For example, a collection could be built from crawls in a subject area, during a crawling period, from a specific domain, or funded by a particular granting institution. The collection owner decides any and all membership and organizing principles that may apply. Separately, curators may choose to reflect or suppress their collection boundaries from the view of other system users, depending on how useful those crawl groupings are to their audiences.

A *crawl* is the content associated with a web harvest operation, and it starts from a seed list of Entry Point URLs (EPUs). An *Entry Point URL (EPU)* is a URL appearing in a seed list as one of the departure addresses for a crawl. During a crawl, other candidate URLs to harvest are usually generated. An EPU necessarily points at or into a website, but it is not the same as a website; even if the EPU points at a clearly identifiable website root, the crawl may only target a subset of the site, or it may lead to the harvest of material (through generated URLs) that are beyond what most people would agree are the confines of the site. A curator may create a descriptive record for any EPU or generated URL.

A crawl and its seed list are defined and described in a crawl record, which may be edited and re-used over time by the curator who created it. Because delays can exist between crawl record creation and crawl execution, content associated with a crawl may not yet have been harvested. The content associated with a crawl is built up as a series of captured content blocks, which may be reviewed by a curator and selectively suppressed. A *suppressed* block remains in the crawl but behaves as if it were not there during normal browsing, indexing, and searching; this provides for limited post-crawl filtering that avoids permanently altering the captured content.

Curators can copy, rename, and delete collections, and can add or drop crawls from a collection that they own. Although these operations might appear to manipulate large quantities of content, a curator need not be too concerned with their cost because internally the storage abstractions are lightweight. A collection "contains" crawls by virtue of a list of crawl identifiers in its collection record, and a crawl "belongs" in one or more collections by virtue of a collection name list stored

in the crawl record. Every crawl must be contained in at least one collection (e.g., a "test" collection).

The service vision relies on an open, collaborative repository environment in which content that you gather is accessible to other curators in a read-only manner, such as for browsing, copying, and linking. The service will also allow for curators to collaborate in the construction of shared collections.

The acquisition, curation and preservation activities described earlier take place throughout the process of creating crawls and collections. For example, some of the decision-making involved in curation occurs as you design the initial parameters of a crawl, while others will take place as you analyze the results and determine how much of that crawl to add to a collection.

Internally, a crawl is stored in a file following the WARC (Web Archive) file format. In this file, content blocks retrieved from visited URLs are concatenated (stored one after the other) in one long file, with the addition in front of each block of a header line specifying such things as URL, crawl time, unique content identifier, and length in bytes of the retrieved content block.

Core Functional Service Areas

There are three major tasks involved in web archiving: crawling content, managing content and retrieving content from the archive. The remainder of this document will detail the specific tasks involved in those three major functions. Light grey items are still under consideration.

1. Crawl content:

BEFORE CRAWL

A. Submit a seed list of URLs from which to begin crawling (for new crawls)
   1) enter a single URL
   2) enter multiple URLs
   3) enter a text file with URLs
   4) enter URLs for an emergency or quick crawl
   5) save seed list as a pending crawl to execute later
   6) provide the ability to add notes to pending crawl
   7) QA the seed list of URLs
      a) validate that the URL syntax is correct
      b) fix URL syntax if incorrect
      c) system shall validate that the URL is reachable – does not return 404 error.

B. Modify crawl parameters (for existing crawls)
   1) remove URLs from seed list
   2) temporarily deactivate URLs from seed list

C. Provide curator with information about past crawl history for URLs in seed list
   1) If crawled provide link to crawl metadata [see c. above] for content crawled by the curator initiating the work order
   2) If crawled provide link to crawl metadata for content crawled if it exists anywhere in the archive – initiated by another curator's work order
   3) If crawled provide link to content itself

D. Specify crawl parameters
   1) frequency/duration
      a) set frequency of the crawl (daily, weekly, monthly, etc)
      b) set duration of the crawl (e.g. weekly for six months)
   2) maximum file size
   3) maximum overall size of crawl
   4) maximum crawl time
   5) maximum # of connections
   6) maximum transfer rate (bytes/second)
   7) flow control rules: number of multiple connections
   8) number of retries, timeout limit (seconds)
   9) number of sub-domains within the domain to crawl
   10) depth of external links to crawl
   11) robot exclusions (obey or ignore?)

E. Initiate crawl

DURING CRAWL

F. Automatically capture metadata about a crawl
   1) date crawl was asked for
   2) date crawl was initiated
   3) response header metadata
   4) descriptive information: title of site, etc.
      Note: Heritrix supplies this info about the crawler's host in its fileheader: crawler software, host IP and name, operator, name of crawl (basically = directory the arcs and logs go into), crawl profile, date, archiveFile location. I believe they are adding request headers.
   5) Automatically capture metadata about the curator (from a login database)

G. Monitor progress of crawl
   both one-time and frequently scheduled crawls
   1) See which sites in seed list have been crawled within a 24-hour timeframe.
   2) See which sites in seed list have not been crawled within 24-hour timeframe
   3) View captured metadata about sites that have been crawled (bytes, formats, etc)
   4) Monitor the output of a frequent crawl for change or redundancy

H. Adjust the progress of crawl
   1) Speed up / slow down crawler performance
   2) pause crawl
   3) restart crawl
   4) kill crawl
   5) truncate crawl (get as far as you can in x hours)
   6) Increase or decrease the frequency of the crawl

AFTER CRAWL

I. QA the crawl: Establish reasonable confidence that the desired information was captured completely and accurately.
   1) Examine logs: numbers of files, number of bytes
   2) Compare the logs of previous crawl that used the same crawl parameters
   3) Random sampling that provides views of the content for human verification
   4) Scum Skimmer: automatically analyze the returned content and set anomalies aside for review
   5) Validate that the crawl parameters were adhered to (I said crawl daily to a depth of 5 how do I know that that happened? How do we build in human analysis)
   6) Verify that you got the whole site
   7) Produce confidence rating for crawl (80 on the scale of 1 to 100 - three star crawl -a spider with 1 or more legs)
   8) Actions as a result of QA process
      a) Toss the crawl
      b) Modify crawl parameters and re-execute work order
      c) Express righteous indignation

J. Add descriptive metadata to:
   1) crawl
   2) Entry Point URLs within crawl
   3) Content URLs retrieved by crawl

   Note: descriptive metatdata may be a combination of automatically generated data and information entered by hand. Metadata may include curator's notes, government agency, copyright status, etc.

2. **Manage and Describe Collections:**

   A. Create a collection

   B. Connect crawls to a collection

   C. Connect to another collection|

   D. Specify parent collection

   E. Delete a collection

   F. Rename a collection

   G. Copy a collection

   H. Add descriptive metadata to collection. (May include curator's institutional affiliation, NDIIPP affiliation note)

   I. Transfer a collection to another institution's repository

   J. Review existing collections by
      1) date crawled
      2) Entry Point URL
      3) curator
      4) institution

   K. Review collection statistics including:
      1) space consumed
      2) access statistics
      3) statistics by file type
      4) space by file type
      5) space used per crawl

3. Discover and Display

   A. Search the archive
      1) Across collections or within a collection
      2) Search against both metadata and full-text
      3) Search metadata separately from full-text

   B. Browse Metadata
      1) Browse by typological category
      2) Browse by site metadata: title, subject, creator, date
      3) Across collections or within a collection

   C. Display
      1) Initial search results will retrieve brief records (details to be determined via needs analysis)
      2) Full display of item record (details to be determined)

3) Display full look and feel of website (all files)

D. Navigate
   1) Within a website
      a) Provide user-friendly way to clearly identify
         a. When link will lead to a document in the archive
         b. when link will lead to a live website outside of the archive

   2) Across website snapshots
      a) Provide navigation through time when many versions of the site are available in the archive
      b) Identify when archived versions in timeline are significantly different