Archimedes Project Repository: Working Draft 4

Mark J. Schiefsky, Malcolm D. Hyman

October 9, 2003

1 Ontology of the Archimedes Repository

Work in the Archimedes Project involves two sorts of data: (1) data that are automatically generated and may be regenerated with ease, and (2) data that cannot be generated without significant human effort. We call the first category of objects **transient** or **volatile**, and the second **permanent** or **non-volatile.** For example, automatically generated morphological data are considered transient, whereas the files corresponding to primary source texts are considered permanent.

The **Archimedes Project ontology** comprises all permanent objects in the project, and only those objects. The ontology is not flat, but rather is structured in terms of hierarchical relations between objects. **Primary** objects are independent of any other object in the ontology. For example, the text of a specific edition or manuscript is a primary object. **First-order derivative** objects depend directly on at least one primary object in the ontology. We distinguish between first-order derivative objects that depend on exactly one primary object (e. g, photographic images of the pages of a text edition) and those that depend on multiple primary objects (e. g., text matching files). **Second-order derivative** objects depend directly on derivative objects, but they depend only indirectly on primary objects. For example, figures depend directly on the image of the page from which they are taken, which depends in turn on the source text. Also, any file that relates first-order derivative objects (such as one that associates instances of technical terms in primary texts with more general concepts) is a second-order derivative object.

Every object in the Archimedes ontology is assigned a unique locator. This locator provides a stable reference to the object, and should never be

changed or reassigned. We describe the conventions for such locators in section 3.

2 Implementation of the Archimedes Repository

Objects in the Archimedes Project ontology will be stored in the Archimedes Project repository. Transient/volatile objects may also be stored in the repository, but this document will not specify any further constraints relating to the storage of such objects.

We describe here first required characteristics of the repository and then additional desired characteristics of the repository.

2.1 Requirements for the Repository

- The repository must allow items to be referenced by their locator for both upload and download operations.
- The repository must allow for batch upload and download of objects.
- The repository must provide sufficient capacity for project work (that is, sufficient file storage and bandwidth). We estimate that the complete dataset for a single text in the project will amount to about 1GB. At present, we intend to have 160 separate texts, but we must allow for further capacity in the future. (It may be desirable in addition to add to the repository archival TIFF files from which all image files are produced. These files are approximately 50 MB in size. If these files are added, our storage needs increase by an order of magnitude.)
- The repository must allow for access restrictions appropriate to work in the project. In particular, upload access must be restricted to appropriate individuals within the project.
- The repository must allow for version control. It must allow for storage of more than one version of the same object under the same locator. A user must be able to retrieve any or all versions of an object.

2.2 Further Desired Characteristics of the Repository

- The repository should allow for multiple levels of a directory hierarchy, so that objects (both permanent and transient) may be stored in a fashion that reflects their semantics.

- It would be valuable for the repository to provide special functionality related to XML documents. Desired functionality includes flagging documents for well-formedness and for validity against a supplied DTD.
- It would also be valuable for the repository's versioning system to include diff functionality and version branching.

2.3 Upload/Download Service

We intend to implement at the MPIWG an additional service that will be layered over the repository. This service will offer the following functionality:

- The upload/download layer will provide a name translation service such that at upload time, local files will be associated with their locator in the repository, even though their file name may not be the same as their locator. At download time, files may be assigned a canonical file name that differs from the locator.
- The upload/download layer will ensure via a checksum mechanism that only changed files are uploaded to the repository. At download time, checksums will be computed and stored for downloaded files. When files that have been previously downloaded are submitted for upload, checksums will be regenerated and compared so as to determine whether any changes have been made.
- The upload/download layer will be implemented as a signed Java applet. It will be compatible with Netscape versions 4.5 and higher, Internet Explorer 5.0 and higher, and modern versions of Mozilla.

We anticipate that the upload/download service will be sufficiently general as to be useful to a wide range of projects outside Archimedes, with similar storage and versioning needs.

3 Conventions for Locators

A locator is a valid URI (see RFC 2396).[1] All locators consist of two components: the first a unique identifier (the ID portion), and the second an

[1] `http://www.faqs.org/rfcs/rfc2396.html`.

extension that serves as a typing mechanism. We define specific conventions for locators assigned to various categories of objects in the ontology.

3.1 Texts

The ID portion of a text locator is a three digit, zero-padded number in the range 001 to 999. The extension portion will be `.xml` for XML texts and `.txt` for raw texts that have not yet been structured as XML.

Example: `049.xml`.

3.2 Page Images

The ID portion of a page image locator is formed by concatenating the ID for the text to which it is related with a two digit image series number (range 01–99) and a variable length image number. The image number is zero-padded so that it will always be the same length for a particular image series. The parts of the ID portion are separated by a forward slash character. The extension portion will vary with the image type (e. g., `.jpg`, `.gif`).

In a case where a new ID must be assigned after images have already been numbered sequentially, one or more letters may be appended to the image number (grammar: [a-z]+).

Examples: JPEG image 101 in image series 02 of text 049 will have locator `049/02/101.jpg`; JPEG image 101a in image series 02 of text 049 will have locator `049/02/101a.jpg`.

3.3 Figures

The ID portion of a figure locator is formed by concatenating the ID for the page image from which the figure is taken with a single-digit figure number (range 1–9). If the figure does not come from a particular page, an arbitrary page number should be assigned that does not correspond to any actual page number. A forward slash separates the figure number from the ID portion of the page image locator.

Example: The second figure taken from page image `049/02/101.jpg` will have the locator `049/02/101/2.jpg`.

3.4 Matching Files

The ID portion of the locator for a matching file is formed by concatenating the ID portions of the locators of the texts that are matched by the file. The

separator character is a hyphen. The extension will be `.xml`.

Example: A file that matches texts 049, 067, and 068 will be given the locator `049-067-068.xml`.

3.5 Other Objects

Conventions for locators for other objects are left for further study.

4 Assignment of Locators

For primary texts, locators will be assigned in an Archimedes Project administrative database.

The image series portion of page image locators (characters 5–6) will also be assigned in an Archimedes Project administrative database. The image number portion will be assigned at the time of image correlation. Images are numbered sequentially.

The figure number portion of the figure locator will be assigned at the time that figures are produced from page images. Figures are numbered sequentially.

The locator of a matching file will be assigned by the author of the file.

5 File Naming Conventions

The locator mechanism allows for persistent reference to objects. File names for objects may or may not coincide with their locator. Software used in the Archimedes Project allows for mapping between locator and file name (or URL). In order to make this automatic mapping possible, we establish the following conventions for canonical file names.

5.1 Texts

The canonical file name for primary texts is an eighteen to twenty-four character designation plus the extension `.xml`. The designation is composed of five letters from the name of the text's author, five letters from the title, *the three-digit ID portion of the text locator,* a two- or three-letter language code (as per ISO 639), and optionally a year (normally, three or four digits; dates B.C.E. are indicated by a prepended "-").[2] An underscore character

[2]The year specified shall be that of the edition from which the electronic text is taken. In the case that a text is protected by copyright, we omit the year from the canonical file name.

occurs as the separator between the parts of this designation. The designation, together with the locator, is specified in an Archimedes Project administrative database.

5.2 Other Objects

For objects other than primary texts (page images, figures, and matching files), the canonical file name is derived from the locator by substituting a period for the forward slash character each time it occurs.

Example: the locator `049/02/101.jpg` is assigned the canonical file name `049.02.101.jpg`.

6 Changes to This Document

- The conventions for the canonical filename were altered on 2003-10-09. The two digit version identifier has been removed, and replaced with ID portion of the locator.