International Journal on Digital Libraries (2006) 6(2): 115–123
DOI 10.1007/s00799-005-0128-x

REGULAR PAPER

Robert Kahn · Robert Wilensky

A framework for distributed digital object services

Published online: 13 March 2006

The following paper was written by the authors over a period of approximately 16 months during the period November 1993 to May 1995 in an attempt to explore a set of open research issues and to integrate them with certain ideas of a small group of researchers who had been briefed on the notions inherent in the Digital Object Architecture that one of the authors (Kahn) had been developing at the Corporation for National Research Initiatives (CNRI). This research group had been organized by CNRI as part of its "Computer Science Technical Reports" (CSTR) project that was funded by the Defense Advanced Research Projects Agency (DARPA).

The CSTR project had several objectives, of which one was to digitize existing collections of technical reports from five leading computer science departments at universities in the U.S and make the reports available on the Internet. A second objective was to fund research at these institutions on digital libraries, especially research that made good use of their local collections. A third objective was to link heterogeneous electronic libraries, such as were being developed in the program.

The difficulties inherent in this third objective ultimately led to this paper. In particular, the motivation arose from a set of concerns first expressed by these researchers at a meeting at CNRI on October 25, 1993 when the Digital Object Architecture was first presented to the research group by Kahn. In the discussion that resulted, many issues emerged, some of which we ultimately decided must be resolved in any architecture, and some of which—perhaps just as crucially—could be deferred for the time being. For example, questions arose about the role of semantics in identifiers; we decided to address them early on. Another issue that proved controversial was how to deal with situations in which the digital objects were actually mobile programs in the network. Since a digital object could contain other digital objects, this led directly to the notion of a mobile repository. The basic question was how best to describe this kind of situation and explain how to access information that was not at a known specific location or IP address on the net. We decided not to cover this aspect of the architecture in the interest of getting closure on the paper.

A critical part of this effort was dealing with various intellectual property issues; and the architecture was designed to take this important aspect into account. We were fortunate to have the help of Ms. Patrice Lyons, an experienced intellectual property lawyer, in the formulation of certain key elements of the architecture dealing with terms and conditions for access to information, and in the drafting of the paper. Although her name does not appear on the paper, her insights and contributions played an important role in the preparation of the paper.

One key component of the architecture is a general-purpose resolution system, known as the Handle System®, an implementation of which has been operational on the Internet since 1994. At present, close to 50 million identifiers are assigned to digital objects and resolvable by the system, which has been available 7x24 for many years. It is anticipated that the number of resolvable identifiers in the system will grow rapidly and could easily exceed a billion in the not too distant future.

Much progress has occurred in the past decade on the remainder of the architectural components, such as repositories, metadata registries, and the associated implementations; and the Handle System continues to evolve to meet new requirements. Uses in both government and the private sector continue to expand. Digital ID World selected the Digital Object Architecture for its 2003 award for balancing innovation with practical reality. Many groups are now using the technology. For example, the International DOI Foundation relies on the Handle System for administering its identifiers, and the Advanced Distributed Learning project, an effort of the U.S. Department of Defense, requires the use

R. Kahn (✉)
Corporation for National Research Initiatives, Reston, VA
E-mail: rkahn@cnri.reston.va.us

R. Wilensky
University of California at Berkeley, Berkeley, CA
E-mail: wilensky@cs.berkeley.edu

of the Handle System for identification of its materials and for resolution of identifiers.

Since the mid-1980s, CNRI had been working on various aspects of digital libraries and mobile programs, referred to as "Knowbot programs." Certain key aspects of this work were subsequently written up in a patent application that was filed with the U.S. Patent and Trademark Office prior to its presentation to the research group in late 1993. The purpose of the patent filing was to insure that the technology would not be constrained by others and its integrity could be maintained. It has since been made available by CNRI under an open source license. The CNRI patent was issued in 2000, but during the intervening years significant progress was made by CNRI and by others in implementing key components of the architecture.

One reason the Internet was broadly appealing was its openness to applications of all kinds that could use it as an underlying infrastructure. The same basic attribute can be seen in the digital object architecture. It neither constrains the choice of applications, nor mandates how they must be implemented. However, as a powerful reference model, with demonstrated implementations of its components, it enables others to confront difficult system design and implementation projects from a well-formulated starting point that enables, and, more accurately, reinforces the ability for heterogeneous information systems to interoperate with each other.

By taking a complex subject and rendering it accessible in simpler terms, it was possible to make a basic set of ideas more understandable to a larger audience. It was not easy to write the paper, and indeed we went through close to a dozen iterations over more than a year in the sometimes elusive search for clarity of expression. We are pleased to see the paper included in this special issue.

1 Introduction

This document describes fundamental aspects of an infrastructure that is open in its architecture and which supports a large and extensible class of distributed digital information services. Digital libraries are one example of such services; numerous other examples of such services may be found in emerging electronic commerce applications. Here we define basic entities to be found in such a system, in which information in the form of *digital objects* is stored, accessed, disseminated, and managed. We provide naming conventions for identifying and locating digital objects, describe a service for using object names to locate and disseminate objects, and provide elements of an access protocol.

We use the term *digital object* here in a technical sense, to be defined precisely below. Files, databases, and so forth that one may ordinarily think of as objects with a digital existence are not digital objects in the sense used here, at least not until they are made into an appropriate data structure, etc., as we will describe shortly.

Only the most basic elements of the infrastructure are described herein. These elements are intended to constitute a minimal set of requirements and services that must be in place to effect the infrastructure of a universal, open, wide-area digital information infrastructure system ("the System"). We anticipate that many other services and elaborations will come into existence as the System is further developed, either building upon or otherwise added to these elements.

This paper focuses on the network-based aspects of the infrastructure, namely those for which knowledge of the contents of digital objects is not required. Definition of the content-based aspects of the infrastructure is purposely not addressed in this paper. An important goal in limiting the description of the infrastructure in this way is not to constrain the higher level user and service level choices that, for many reasons, might be inappropriate to fix upon at this point in time. With only the most basic elements of the infrastructure in place, technological evolution would not be overly constrained. Further, the likelihood of achieving widespread interoperability of services at some early point in the future will be preserved. No doubt, the resulting capability will have a greater potential for enhancement and evolution through the participation of many others in helping to define it.

2 Overview and definitions

In this section, we first present an informal overview of the elements of the System, sketching its elements and how they are supposed to function together. These elements include the notions of *digital objects*, *handles*, *metadata* and *key metadata*, *repositories*, *handle generators*, *originators*, *users*, *global naming authorities* and *local naming authorities*, and a *repository access protocol*. Then we provide more formal definitions of these entities, and explicate their details.

2.1 Informal overview

Conceptually, the System works as follows: An *originator*, i.e., a user with digital material to be made available in the System, makes the material into a *digital object*. A digital object is a data structure whose principal components are digital material, or *data*, plus a unique identifier for this material, called a *handle* (and, perhaps, other material). To get a handle, the user requests one from an authorized *handle generator*. A user may then deposit the digital object in one or more *repositories*, from which it may be made available to others (subject, of course, to the particular item's terms and conditions, etc.). Upon depositing a digital object in a repository, its handle and the repository name or IP address is registered with a globally available system of *handle servers*. Users may subsequently present a handle to a handle server to learn the network names or addresses of repositories in which the corresponding digital object is stored.

Interactions such as depositing digital objects or accessing digital objects in repositories is accomplished using a *repository access protocol* (*RAP*), which all repositories must support.

A digital object stored in a repository, and whose handle has been registered with the handle server system, is called a *registered digital object*. Registered digital objects are of primary concern to us here, as they are explicitly constructed to be known about by others, presumably for widespread availability. However, we do not constrain repositories to contain only registered digital objects. Nor are repositories constrained to operate only via the repository access protocol, although they must all support it.

Handles are the primary global identifiers for digital objects. However, we do not anticipate that users will necessarily manipulate handles directly; nor is the system of handle servers intended as the only means by which users will locate objects. More likely, location services will be accomplished by various value-added providers not defined as part of the infrastructure. Rather, the handle server system provides a kind of public safety net, which facilitates the location of a digital object given only its handle.

We emphasize that the term *digital object* is used here in a technical sense of a particular sort of data structure, and not in the general sense of any object that may have digital form. Perhaps a term such as *digital infrastructure object* would better capture this intention. However, we have found this alternative terminology to be somewhat cumbersome in practice, and have therefore chosen to retain the simpler term digital object instead.

2.2 Definitions

We now define our terminology more formally, and describe the operation of the various components of the System in some detail.

Formally, a digital object is an instance of an abstract data type that has two components, *data* and *key-metadata*. The data is typed, as is described below. The key-metadata includes a *handle*, i.e., an identifier globally unique to the digital object; it may also include other metadata, to be specified. Possible primitive and composite data types for digital object data are discussed below.

A *repository* is a network-accessible storage system in which digital objects may be stored for possible subsequent access or retrieval. The repository has mechanisms for adding new digital objects to its collection (*depositing*) and for making them available (*accessing*), using, at a minimum, the *repository access protocol*. The repository may contain other related information, services, and management systems.

Repositories have official, unique names, assigned or approved to assure uniqueness by a global naming authority. In general, the *global naming authority* will assign a name to a local naming authority. The local naming authority may use this name as the name of a repository. In addition, it may extend this name to create new names by suffixing the name with a ".", followed by a new (relatively) unique name component. Each such name represents a naming authority and potential associated repository. (i.e., In general, repositories will have unique names of the form "X.Y.Z".)

Note that a repository name is not necessarily the name of a particular host. For example, it may correspond to a set of hosts at different physical locations.

A *stored digital object* is a digital object stored in a repository. In addition, handles are expected to be made known to a system of *handle servers*, as described below. Such a handle is a *registered handle*. A *registered digital object* is a stored digital object whose handle has been registered. (Note that a handle cannot be registered until its corresponding digital object is stored) Repositories provide users access to stored objects under terms and conditions that may be set by the depositor and/or a given repository.

Registered digital objects are the entities of primary concern to the infrastructure, since they are stored in a repository and made known via the registration of their handles. Intermediate entities, such as stored digital objects, are defined only because they may arise in implementations of repositories that provide access to registered digital objects.

However, their existence is not strictly necessary. For example, a repository may offer a service in which it deposits a digital object and registers the handle simultaneously, therefore creating a registered digital object without creating a prior stored, but not registered, digital object. (It is possible, of course, to create other useful classes of digital objects. For example, we may define a *proposed digital object* as a digital object whose handle field contains a string that has not yet been registered and whose uniqueness may not yet be known.)

Each repository contains a *properties record* for each of its stored digital objects. The properties record comprises all metadata for a digital object, including its key-metadata, but also, other metadata the repository may maintain for that digital object. Notionally, the key-metadata component is a subset of metadata, which is invariant for a digital object over repositories. No attempt is made in this paper to delineate how much of the metadata should be included in the key-metadata, other than requiring that it include the mandatory handle. Possible examples of repository-dependent metadata are the general terms and conditions for access and usage of the digital object, and the date and time of deposit.

A simple *repository access protocol* (*RAP*) is supported by each repository (and defined in Sect. 3.1). Only the minimal necessary aspects of the RAP are specified here. We anticipate that these aspects of the RAP, or the RAP itself, will be a subset of the interface protocol used by repositories, and require only the functions or operation of the RAP not be affected by any implemented supersets of the protocol. In particular, the RAP allows for accessing a stored digital object or its metadata by specifying its handle, a service request type, and additional parameters. If this request is complied with, the output of the service request is a termed a *dissemination*. A dissemination is the result of an access service request, along with additional data affixed to it, to be specified below.

An *originator* is an entity that authorizes or validates a set of digital objects; it is responsible for each such digital object including making it available in the System and defining terms and conditions for its use. Every digital object has an originator, which may be an individual or an organization (there may be a number of kinds of originators worth distinguishing, but we do not differentiate them here). Originators may deposit and access the digital objects they authorize or validate and may authorize others to do so (this also includes the right to withdraw or modify the objects), subject to the procedures established by individual repositories. Naming authorities have the right to insert handle entries for handles they generate into the handle server system and to authorize others to do so. The relationship of the originator to the naming authority is left unspecified here. An originator and/or a naming authority may also delegate this authorization ability to others (typically this would be to one or more repositories). Such delegation includes at least the right to authorize the further deposit of digital objects on behalf of the originator and insertion of designated groups of handles on behalf of the naming authority. Repositories may establish additional requirements of various kinds. The process by which an originator or a naming authority informs a repository of any such authorization is left unspecified here.

The initial repository used to deposit a registered digital object is designated the *repository of record* (*ROR*). The ROR is responsible for authorizing additional instances of the digital object at other repositories, and for making changes or withdrawals of such additional instances of the digital objects, usually upon the direction of the originator. Once designated, the ROR may subsequently be changed by an authorized party to another repository, but the method for achieving this is not specified here. The notion of ROR is not defined for stored digital objects that are not registered.

A handle is a globally unique string, produced by an authorized *handle generator*. It consists of two logical parts, concatenated with an intervening separator character. The two logical parts are: (1) name of a *local naming authority*, which controls the handle generation process, and (2) a locally unique string, which is assigned by (one of) its handle generator(s). An originator may ask a handle generator for a handle, or it may propose a local string to be used. The local handle generation process should insure that local strings are unique. Handles have no prescribed maximum length in principle, but there will be a default length in existence at any time, which can be adjusted upwards if necessary.

For handles to be unique, the names of local naming authorities are controlled by the global naming authority for the System. The global naming authority generates names for local naming authorities, and assigns these to local naming authorities for use by the handle generators they authorize. A prospective local naming authority may propose a name for itself to the global naming authority for validation and registration. A local naming authority, named, say, "X", may create additional, derived naming authorities of the name "X.Y", etc., each authorizing its own handle generator. (At this point, it is left unspecified whether the naming authority name spaces for repositories and for handle generators are distinct.)

In addition to the first globally assigned component (e.g. "X"), each subsequent component field of a naming authority name (e.g. "Y", or "Z") must be nonnull and not contain the character ".". There may be other restrictions on the nonalphanumeric characters to be used in naming authority names. In particular, the default separator character is "/" (so, e.g., "X.Y/local-string" is a typical handle from the naming authority "X.Y") Other separator characters, and a syntax for defining other separator characters, (from a restricted class of nonalphanumeric characters) may be defined, and may entail other restrictions on the possible characters used in naming authority names, e.g., a conceivable syntax is to specify a nondefault separator by an initial nonalphanumeric character, so that "%X.Y%local-string" is a valid handle. We leave unspecified at this point how this might be accomplished, whether otherwise identical handles with different separators are identical or distinct, whether an *escape character* for restricted characters exists, and whether the separator characters are restricted (e.g., whether "a/b" is a possible naming authority name that can only be used with a nondefault separator). Initially, naming authority names will be issued conservatively, being restricted to alphanumeric characters.

The handle generator may be a person, an organization, or a fully-automated process running on some machine or a set of machines. An originator may control a naming authority, but there may be naming authorities that are not controlled by originators. The details of interaction with handle generators are left unspecified.

It is also unspecified what an originator must supply to a handle generator in order to receive a handle. An originator may propose handles to be assigned to its digital objects. Moreover, the handle generator need not assume any responsibility for insuring that a handle, which it generates, is associated with any particular digital object; that correspondence may be left to the originator.

A stored digital object may have associated with it in a repository a *transaction record*, which records transactions of that repository involving the digital object. The transaction record may contain entries such as the time and date of deposit of the object, the time and date of each request for retrieval of the object, the identity of the requesting party, the handle and service request for the object, and the applicable terms and conditions including amount and method of payment. Transaction records will only be made available to authorized parties. Repositories are not required to have transaction records persist for any period of time and it may store transaction records at various times and places as deemed necessary subject to administrative controls.

The data of each digital object is typed. Data types assumed to be in the System include *bit-sequence*, *digital-object*, and *handle*, and also *set-of-bit-sequences*, *set-of-digital-objects*, and *set-of-handles*. Other data types can be defined and made available to the System via the type construction operators *set-of* and *compose*; these types are then registered in a global type registry. The mechanism for this registration is currently unspecified. Note also that there is, at present, no (defined) registration of methods associated with types.

In contrast, one can create subtypes of digital objects by introducing new fields of metadata; these may be arranged hierarchically. For example, one might create a subtype of digital object called *computer-science-technical-report*, which has metadata for *author*, *institution*, *series*, and so forth.

We shall informally refer to digital objects whose data is a set, one of whose elements is of type *digital-object*, as *composite digital objects*. A digital object that is not composite is said to be *elemental*. (Note that this definition explicitly excludes the application of the adjective *composite* to a digital object whose data is another digital object, i.e., whose data is of type *digital-object*, as distinguished from a singleton set of this type. Nothing precludes the existence of such objects, however.)

The terms and conditions of a composite object may implicitly or explicitly be unioned with those of its constituent objects to arrive at the terms and conditions for those constituent objects. Terms and conditions may be explicitly imposed only on the composite object, in which case they would apply to each constituent object; or each constituent may have its own separate terms and conditions in addition. (Of course, creating composite digital objects may be subject to copyright and any other legal restrictions pertaining to its constituent objects.)

A digital object's data may incorporate information or material in which copyright, design patent, or other rights or interests are claimed. There may also be rights associated with the digital object itself. An author may have submitted a digital object for purposes of registering a claim to copyright in a work that may be incorporated in the object. Since the copyright pertains to the underlying work fixed in the form of the particular submitted representation, the rights would normally pertain to all representations of the work, including, but not limited to, those representations of the work that are contained in other digital objects.

While we intentionally avoid issues of content in the infrastructure, we note that the entities provided thus far give users a number of means to include digital objects that contain or may be interpreted to manifest the same or similar information or material. As an example, a literary work may be fixed in a number of different formats, e.g., LaTex, PostScript, and GIF page images. Each fixation may correspond to a distinct (elemental) digital object, each with its own unique handle, and other metadata). A composite digital object may then be created whose data is the set of these digital objects. Similarly, one could create a composite digital object whose constituent objects were the fixations of the literary works of Shakespeare in PostScript. The handle of this composite digital object, in effect, names the PostScript collection of Shakespeare's literary works.

Note that it is possible to construct objects with similar effects without using composite digital objects. For example, the single digital object intended to correspond to a work could have data of type *set-of-bit-sequences*, rather than of type *set-of-digital-objects*, and contain each of the forms of fixation therein. In this case, digital objects may not exist corresponding to the individual fixations. Another possibility is to have a digital object whose data is of type *set-of-handles*. In this case, the handles would name the individual fixations (which may not even be available from the same repository). Such a digital object may contain other data fields that further describe (or annotate) the handles. Yet another possibility is to create a markup language, which admits handles, plus other conventions for expressing how they relate to each other (for example, whether the individual handles are meant to be interpreted as different fixations of the same work, or a list of bibliographic citations, etc.) A digital object whose data comprise sentences in this markup language could serve to represent the same entities as do composite digital objects.

We use the informal term *meta-object* to refer to a digital object whose primary purpose is to provide references to other digital objects. Both digital objects whose data are of type *set-of-handles* and digital objects in a markup language that admits handles, would be instances of meta-objects.

A digital object may be *mutable* in that it may be changed after it is placed in a repository. Although none of the key-metadata may be changed, nor may any known digital object that it contains be changed (unless the original digital object is also changed), most other changes are permissible. Minor changes might be made to correct a misspelling or other such error; changes to the title of a mutable digital object may be permissible. A mutable composite digital object could be modified to add the representation of an underlying work in a new format. Mutability would also be a useful way to allow digital objects that are designed to change with time or are dynamically computed.

A digital object that cannot be changed is said to be *immutable*. If an object is immutable, then, once it is placed in a repository, the result of all subsequent requests to that repository that are functionally dependent on the data of the object must be identical. (However, it may be possible to remove an immutable object from a repository, or deny access to it at different points in time.) That a digital object is immutable may be reflected in its key-metadata. It is also possible that a given repository may preclude changing a stored object by an indication in its nonkey-metadata.

Once set, the mutability or immutability of a digital object cannot itself be changed. Users who wish to achieve a comparable effect would have to create a new digital object with similar data and altered metadata. The original digital object may then be withdrawn or not, as desired.

There is no requirement that a digital object be stored in a repository in any particular manner. Conceptually, the description of a digital object is strictly a logical one and is not intended to describe any particular implementation. In particular, it is possible that, in response to a request to access a particular digital object, a server runs a program that computes the digital object on the fly. It is possible for multiple digital objects to be embedded in a program (e.g., a data base manager or knowledge-based system) that emits them upon request. The program may itself be a digital object. Thus, accessing and depositing are virtual processes, and may or may not involve the actual depositing and retrieval of actual objects per se, although such actual storage and retrieval is likely to be prevalent.

3 Accessing digital objects

3.1 Repository access protocol (RAP)

Each repository must support a simple protocol to allow deposit and access of digital objects or information about digital objects from that repository. This is called *Repository Access Protocol*. RAP is meant to provide only the most basic capabilities and may evolve over time. Repositories may support other more powerful query languages that allow users to access objects that meet meaningful criteria. At present, the RAP includes deposit of digital objects, access to digital objects by handle, and related repository services. Each of these capabilities will produce different results, depending on the specific nature of the service request.

3.1.1 Access to a digital object (ACCESS_DO)

Access to a digital object will generally invoke a service program that performs stated operations on the digital object or its metadata depending on the parameters supplied with the service request. Defined service requests include *metadata*, *key-metadata*, and *digital object*; the first requests only the metadata, the second only the key-metadata, and the latter, the entire digital object (i.e., the key-metadata and the data). Other systems-level services may be defined. Possible examples of such additional services might be *encrypt*, i.e., return the digital object in some encrypted form, or *compress*, i.e. store a fewer set of bits than supplied with the property that the original bits can be regenerated, perhaps exactly. However, we do not define such additional requests, here.

In addition, it is possible that data-type-dependent service requests will be introduced. Possible examples of such data-type-dependent services requests might be *execute* (for digital objects a portion or all of whose data component is of type *program*), or *subpart* (which requests only a component of the data or metadata of the digital object, further specified by some parameter). We emphasize that such data-type-dependent service requests are not defined as part of the System infrastructure.

When a digital object is accessed via *ACCESS_DO*, the recipient receives a *dissemination*, that is, the result of the service request, along with information such as the key-metadata of the digital object, the identity of the repository, the service request that produced the result, the method of communication (if appropriate), and a transaction string corresponding to an entry in the transaction record. The transaction string is unique to the repository. In addition, the dissemination may contain an appropriately authenticated version of some portion of the properties record for that object, including the specific terms and conditions that apply to this use of the digital object and the materials contained therein.

As noted above, depending on the nature of the *ACCESS_DO* service request, the dissemination may not be stored as a digital object per se. It might instead include data that is not contained in any registered digital object, such as a portion of a digital object's data, the digital object data in a compressed format, or the result of executing the data of the digital object. In all cases, however, the key-metadata (including, of course, the handle) of the digital object is included.

From a copyright perspective, if the service request produced a dissemination that was derived from a particular digital object, the digital object may be *contained* in the dissemination, in the sense that the dissemination may be encumbered by the rights associated with the digital object. For example, if the data of a stored digital object represents an episode of a television program, and the dissemination contains the data corresponding only to the 2 min of this television program, the dissemination may be said to contain the digital object in a legal sense, even if it does not properly contain all of its data.

3.1.2 Deposit of a digital object (DEPOSIT_DO)

Several forms of DEPOSIT_DO are possible. For example, one form may take data, a handle, and perhaps other metadata as arguments, and produce a stored digital object and properties record from these arguments. Another possible form may take a digital object as argument, perhaps with additional metadata, and simply deposit it. Yet another form may take only data and certain nonkey-metadata, and automatically request a handle from a handle server, and then simultaneously store the object and register the handle.

The DEPOSIT_DO command may be used to replicate an existing digital object at additional repositories. The exact method of controlling such replication, if any, is unspecified here. A DEPOSIT_DO command may also be used to directly modify an existing mutable digital object. Alternatively, a modified version of an existing digital object may be stored as a new digital object rather than by modifying the existing one.

3.1.3 Access to reference services (ACCESS_REF)

This command provides a uniform and understood way to identify alternate means of accessing a specified repository and/or information about objects in that repository. Two possible responses are (i) *No information*, and (ii) a list of *servers*, *protocol-name* pairs, with the interpretation that each server, speaking the named protocol, will provide information about the contents of the repository. (i.e., we provide a means of allowing a repository to have its contents indexed, queried, or otherwise described. It is possible, for example, that a repository will be its own provider of information about its contents, and list only itself, and some protocol, as the information provider about its contents. However, it is not required that any accounting of the contents of a repository be available, or that it be available from any one service. This is because we do not require that repositories per se correspond to coherent collections, which may be distributed across independently operated repositories.)

The initial RAP has been purposely kept simple, and all the more complex transactions are assumed to be handled by other protocols, or by subsequent extensions of the RAP. In the first case, a primary use of the RAP for more sophisticated repositories is to have it present the other protocols that it supports (e.g., Z39.50, SQL3, ZQL, Dienst) as alternative access methods.

It may be desirable to extend the RAP in any number of ways, for example, to explicitly include, for example, a payment mechanism or a negotiation mechanism or a more sophisticated interactive model-based interaction mechanism.

Above we described the possibility that a user may construct a single digital object whose data is the set of all fixations (i.e., known formats) of a given work. If so, then there is as yet no formally defined method within the RAP to determine what formats are available, and then, to extract one of them. We expect a set of mechanisms to be developed which expand upon the internal structure of the objects in the infrastructure, but this level of description has intentionally been omitted here.

3.2 The handle server infrastructure

A highly reliable distributed system of *handle servers* is maintained as part of the infrastructure. These servers map handles to network resources at which the corresponding digital objects are available. Handle directory servers are also stipulated; these will be located at certain well-known locations and will maintain a table of network addresses of handle servers (generally, each handle server will contain such a directory). This table will generally be downloaded by each participating site frequently enough to be "acceptably " up-to-date at all times. Local handle servers may also exist. A local handle server could be run by an organization if it wishes to keep a store of pertinent handles locally. These local servers may access the global system of handle servers, but are not themselves necessarily accessible from the global system. Caching handle servers also may be run at local workstations on behalf of individual users to store location information for frequently used handles.

The handle server system is intended to be a means of universal basic access to registered digital objects. In the worst case, a user can present a handle to a handle server and be advised of some repository, which an authorized party has asserted contains the digital object designated by the handle. The handle server is not meant to be the only, or even primary, means, to locate repositories. Primary access may be provided locally and also by value-added service providers, likely in a variety of different and possibly incompatible ways. Users interacting with such services may not encounter handles; and such services may interact with repositories via RAP or via protocols that do not involve handles.

Handle servers provide a number of services, three of which are RESOLVE, INSERT, and DELETE. A party that is authorized to insert, delete, and otherwise change handle entries for a particular naming authority is called a handle administrator. A naming authority will generally designate one or more repositories to act as handle administrators on its behalf. This designation will be made known by the naming authority to the handle server system.

(i) RESOLVE: A handle is sent to a handle server to locate network addresses of repositories containing that object. The handle is first mapped to locate the handle server from the handle directory server table but is not otherwise interpreted. One can also supply a handle to a separate system, which invokes the above procedures to find the stated object. Local handle servers may use any technique to do the mapping. The handle servers maintained as part of the infrastructure map the handles by hashing them.

No guarantee is made that the identified repositories will provide the designated object. Rather, the user is assured only that the specified repositories are where authorized maintainers of repository services have indicated particular digital objects reside.

Since a handle is just a unique string, it can be mapped to an actual repository by any of several mechanisms, including a mechanism that attempts to interpret the string. Repository names are not actual network addresses; they must first be mapped to network locations. The method for accomplishing these mappings is not specified. The handle service is one available means for both kinds of mappings; it would specify at least the location of the interface that supports the RAP protocol for a given repository. There may also be a need to explicitly provide a country identifier for repositories, naming authorities, and/or originators. For the present, however, country identifiers are to be omitted.

When a repository is identified by a handle server, it will be most efficient to map the handle directly into the network address (or addresses) of the repository. This mapping avoids having to do a double lookup from repository name to repository location. However, if the location of the repository were to change, the handle server would have to be notified so it could make the corresponding changes. It is possible that certain repository names may resolve to broadcast addresses to locate specific machines. This might be the case where a single repository consists of multiple machines on a local area network at a given site. The handle administrator may determine whether to store IP addresses or domain names or other information in the handle server. The entries are typed and therefore one or more of the above information types may be provided by the administrator for retention in the handle server.

(ii) INSERT (DELETE): Information associating handles with network services are inserted into (deleted from) the handle server system by the handle administrator or other parties authorized by it. Such authorized parties include repositories of record. The repository of record is presumed to make known to the handle server system that it contains (or no longer contains) a particular digital object some reasonable time after the digital object is deposited in (withdrawn from) it. Similarly, the repository of record would make known to the handle server system the identity of other repositories, which it authorizes to store a given digital object. The handle server system may perform certain administrative functions upon receipt of unauthorized requests. In addition, some form of reporting may be desirable to insure that entities that misbehave can be detected.

3.3 Value-added reference services

The handle server system is intended as a *safety net* of information about where digital objects reside. There will no doubt be other, valuable services that provide information to users about the location of digital objects in repositories. However, we do not consider these services per se to be part of the infrastructure of the System. Instead, they comprise value-added services whose nature we do not see as appropriate to constrain.

In addition, as mentioned above, we do not require repositories to provide a description of their contents. Repositories may not house coherent collections, and hence, querying or searching a repository may be a service appropriate only to the repository administrator, not to a user. Presumably, such capabilities will exist in the form of value-added services. It is such services, rather than repositories per se, that users would interrogate to identify digital objects of a certain nature. Such services may, of course, be offered by repositories themselves, especially in the case when one is intended to house a coherent collection. However, such a server is not a requirement of a well-behaved repository.

4 Imposing semantics on handles

As discussed above, a handle is presumed to have two logical components, a local naming authority name, and an identifier unique to that naming authority. These naming authorities will be assigned in a manner. For example, there may be a "naming authority" named "berkeley", which will authorize other naming authorities within the "berkeley" domain. Within the "berkeley" domain, names are locally assigned to other naming authorities. Thus, the name "berkeley.cs" might be assigned to the authority responsible for naming the UCB Computer Science technical report series (or to several such series). Note that this particular naming authority will not generally correspond to a valid Internet address, even though it may follow similar syntactic conventions.

Particular naming authorities may follow their own conventions for assigning semantic or nonsemantic strings to their objects. For example, "berkeley.cs" may follow a proposed convention for its technical reports, and give each of the corresponding digital objects (whether composite objects or meta-objects) a local handle, e.g., "csd- 93-712". (The "csd"—for "Computer Science Division" is perhaps redundant; however, we use it here to indicate the possibility of a single naming authority issuing several distinct series.)

The full unique handle for this digital object would be

```
berkeley.cs/csd-93-712
```

where the "/" separates the naming authority name from the string unique to that authority.

In addition, digital objects may exist for this work in each of a number of fixations (formats). The handles for these fixations may also be semantically interpretable, e.g., the string "csd-93-712/all.ps" might be the unique local part of the handle for the digital object corresponding to the PostScript version of this work; "csd-93- 712/all.tif " the handle for the tiff representation. (Note that the character"/" is allowed in the local name. It may also be desirable to distinguish other characters, but this is not discussed further in this paper.)

Other schemes may be used to generate handles in other ways. For example, the local portion of a handle might correspond to a date-time format, so that the digital object above might instead have the handle

```
berkeley.cs/1994.12.05.23.42.12;7
```

These handle forms can be embedded within various syntactic wrappers to distinguish them in various contexts from other notations. For example, the handle might be expressed in URN syntax as follows:

```
<URN:ASCII:ELIB-v.2.0:berkeley.cs/
  csd-93-712>
```

Here "ELIB-v.2.0" is supposed to suggest (via "ELIB") that this is a URN for electronic library material, and also, (via "-v.2.0") that some particular naming convention is used by the naming authority. Another possibility is the notation used by Grass and Arms (GA1994), which resembles that for URLs, and proceeds that handle with the prefix "hdl://" (to denote that a handle follows), or just "//" (if it is important to distinguish a global root for the handle), e.g.:

```
hdl://berkeley.cs/csd-93-712
//berkeley.cs/1994.12.05.23.42.12;7
```

The user of this notation is cautioned to avoid confusion with URLs, which name services, while handles name digital objects, not network services.

Various services might exploit semantic conventions to locate an object given its handle, without consulting a handle server. For example, a naming authority may have its own repository and reference server associated with it; the latter might be looked up (perhaps via an additional service), and queried for the location(s) of this particular report.

Users may, of course, attempt to incorporate all manners of semantic or system content in handles. Also, it is plausible that imposing any content in handles per se could be troublesome. Instead, handles per se could be declared to be uninterpreted, and an additional level of indirection be introduced to interpret them. Additional name services could be created to translate user-oriented nicknames to system-oriented handles, as are done for file systems today. We stop short of advocating such a system here, however, assuming that a semantically-motivated convention, such as that which has served for URLs, will continue to be useful at some level, and does not require an additional level of mediation.

5 Conclusion and summary

This paper provides a method for naming, identifying, and/or invoking digital objects in a system of distributed repositories that provides great flexibility and is well-suited to a national-level enterprise. It allows the possibility of locating digital objects without making any presumptions about the object or its locations(s). It also admits value-added conventions that various users may use to their own advantage. For example, a reference server might internally refer to an object by its global handle, and, additionally, keep track of repositories in which this object is believed or known to reside. If a user requests this object, the reference server might look up the repository name or address, determine the repository service, and ask that repository to deliver a version of the object to the user. Alternatively, the server might instead use the object's handle at run time to query syntactically a handle server for the name of repositories or services that house the object.

This system also allows for *public* and *private* naming authorities. Many naming authorities will be private, and only assign identifiers to their chosen clientele (e.g., department members eligible to produce technical reports); however, public naming authorities could provide a service whereby they generate an identifier to anyone who requests one. Individual citizens not associated with any official body might use a public naming authority to generate identifiers for objects they wish to store for private purposes or for public dissemination on their own (this is an example of a situation in which the originator does not control the naming authority.)

In the CS-TR project, CNRI is providing the global naming authority plus a handle management service that accepts handles with and without semantics. This service does not make use of handle semantics; however participants are able to take advantage of handle semantics, if any, to access objects directly. Each participating institution would be free to propose or request names of its own choice. Each of these names may also have associated with them a nonsemantic identifier (such as a date-time-stamp), which is not otherwise specified in this document.

Acknowledgements This research was supported by the Advanced Research Projects Agency under Grant No. MDA-972-92-J-1029 with the Office of Naval Research. We would like to thank Jerry Saltzer, Michael Stonebraker, Jim Davis, Carl Lagoze, Bill Arms, Hector Garcia-Molina, Jim Gray, Patrice Lyons, David Ely, Judy Grass, Barry Leiner, John Garrett, and all the members of the CS-TR project for their many helpful comments on and insights into this work.