```
   Constructor:  $WHO-PROTOCOL://$WHO-HOST/$PERS_NAME

   Component    Value   Default         Prompt                  Required?
   ---------    -----   -------         ------                  ---------
   WHO-PROTOCOL         whois++         Protocol for WHOIS++    Y
   WHO-HOST             bunyip.com      Host running WHOIS++    Y


   URL-ID:       a-netfind
   Abstract:     Generic NETFIND search
   Constructor:  $NET-PROTOCOL $NET-HOST $NET-SELECTOR

   Component    Value   Default                 Prompt                  Req'ed?
   ---------    -----   -------                 ------                  -------
   NET-PROTOCOL         telnet                  NETFIND protocol        Y
   NET-HOST             netfind.ee.mcgill.ca    NETFIND server host     Y
   NET-SELECTOR         netfind;;2/Srch;        NETFIND selector string Y
  $PERS_NAME $PERS_FIRST
  $ORG


WHAT-ELSE:  [optionally empty]

HOW:
                pers_res = run(a-whois);
                if empty(pers_res) then
                    pers_res = run(a-netfind);

WHAT-THEN:
                pers_res = last(pers_res, 10);
```

[**TCL**] Ousterhout, J. K. "Tcl and the Tk Toolkit", Addison Wesley, 1994.

[**TELE**] White, J. E., "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic White Paper, General Magic Inc., 1994.

[**TACO**] Johansen, D. van Renesse, R. Schneider, F. B., "An Introduction to the TACOMA Distributed System", Technical Report 95-23, Department of Computer Science, University of Tromso, Norway, June 1995.

10 Appendix – A Sample Search URA

This appendix contains a text-description of the contents of a sample search URA. This is an architecture-level description, for discussion purposes – this is not meant as an implementation proposal.

This example shows a person-search type URA containing two searching URL's; a "whois" and a "netfind" search. Thus this URA will attempt two different kinds of searches for the invoker. In this case, the URA supplies the same person's name to both URL's.

Although this example is not particularly exotic, it does demonstrate the URA's ability to specify a net-based activity: finding information about people. The user of this URA need not be troubled by specifics of URL construction (defaults are provided for all URL constructors), and can simply provide details about the information need (person name, etc) – the URA will take care of the protocol-specific details.

```
ID:

    URAname:    PersonSearch
    Abstract:   This URA  uses whois and netfind services to find
                information people with names matching the input.
    ScriptLang: tcl


WHAT:

   Component     Value  Default Prompt                  Required?
   ---------     -----  ------- ------                  ---------
   PERS_NAME     <b>    <b>     Last name of person     Y
   PERS_FIRST    <b>    <b>     First name of person    N
   ORG           <b>    <b>     Organization person     Y
                                    works for


WHERE:

   URL-ID:       a-whois
   Abstract:     Generic whois search
```

7 Conclusions

Most of the discussion above implicitly assumes that a human user will provide URA activation data ("what", and optionally "where"). However, there is nothing in the model that precludes the writing of programs that will interact with a URA system to select URAs and provide the necessary data for activation. This widens the scope of possible applications of URAs considerably. As more applications necessarily assume access to the Internet for functioning, URA activation may happen from functions buried deep within the program code – an application might make use of a URA to get periodic updates of documentation, etc.

The basic premise of the proposed URA scheme is that capturing the basics of composite net-activities leaves open the possibility of using URAs as the building blocks of more sophisticated systems.

8 Acknowledgements

The URA work described above grew out of initial suggestions by Peter Deutsch and continues to evolve through the discussions and work with Peter, as well as Christopher J. J. Alpaugh, Bill Heelan, and others in the Bunyip community.

9 References

[**RFC1034**] P.V. Mockapetris, "Domain Names - Concepts and Facilities", RFC 1034, November 1987.

[**RFC1035**] P.V. Mockapetris, "Domain Names - Implementation and Specification", RFC 1035, November 1987.

[**RFC1738**] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URL)", RFC 1038, December 1994.

[**URA-ID**] L. Daigle, P. Deutsch, W. Heelan, C. Alpaugh, and M. MacLachlan, "Uniform Resource Agents (URA's)", March 1995. Circulated as an Internet-Draft, now available from

`http://services.bunyip.com:8000/products/silk/silktree/uratree/uradraft.txt`

[**MIDWARE**] L. Daigle, Position Paper – URAs for MiddleWare. Prepared for participation at SigComm'95 Workshop on Middleware, August 1995. Now available from

`http://services.bunyip.com:8000/products/silk/silktree/uratree/mwposition.html`

[**JAVA**] "The Java Language: A White Paper" Available from

`http://java.sun.com/1.0alpha2/doc/overview/java/index.html`

- to cause that URA to be activated
- to find out what results were obtained by activating that URA (*e.g.*, error reports, results in a predefined format, etc).

  Thus, the invoker may require some or all of the following from the URA system:

- list of available URAs (of a given type)
- identifying (creator/author) information about a URA object
- list of content information required by the URA (*e.g.*, what to search for)
- list of Internet resources to be accessed by the URA
- the URA object itself (to save it locally, etc)
- the results of activating a URA (results are defined by URA type)
- URA activation status/error reports

6 Silk – A URA Application

In a project supported by the Canadian Government's CANARIE TD2 program, Bunyip Information Systems has developed a prototype application based on these URA principles. (See http://services.bunyip.com:8000/products/silk/).

Silk, an Internet client program, allows users to access and search for Internet information without having to know where to look or how to look. Silk's graphical user interface lists the various types of activities that are available to users(*e.g.*, "search for tech reports", "hotlist", etc). For each activity, users need only provide information required to complete the action (*e.g.*, which type of items to be found) and Silk then contacts existing Internet services, as needed, to carry out the activity. For example, Silk search objects include pointers to the information resources that are most likely to be useful for each type of search, and generates the appropriate query automatically. And, users can save out favourite instances of searches to avoid having to fill out the same form time and time again.

Silk uses Internet "resource handles" – objects that contain both a text string describing the resource and a pointer to its location. These handles are treated as real objects in the Silk application – they can be edited, moved around, compiled into collections, etc.

For example, a Silk search will return a collection of resource handles, and the user can pick out the ones that look most interesting. Silk displays the full resource, working in conjunction with a World Wide Web browser of preference (Mosaic, Netscape, Spyglass).

By treating both Internet activities and resources handles as objects that are on the desktop, Silk puts much of the power of shaping Internet interactions into the user's own hands.

Silk activities are embodied by a prototype version of URAs. Users can build new ones, adjust old ones, or add ones that found on the net.

WHAT: Specification of the data elements required to carry out the URA activity. For example, in the case of an Internet search for "people", this could include specification of fields for person name, organization, e-mail address, etc.

WHERE: Specification of the URL/URN's to be accessed to carry out the activity. A software search created in Canada will likely access a Canadian Archie server, and North American ftp sites. However, an invoker in Australia should not be obliged to edit the URA object in order to render it relevant in Australia. The creator, then, can use this section to specify the expected type of service, with variables for the parts that can be modified in context (*e.g.*, the host name for an Archie server, or a mirror ftp site).

WHAT-ELSE: (Instance information) At the very least, an invoker may wish to evaluate what URL/URN's are to be instantiated as part of the decision process for launching a URL.

HOW: If URAs were strictly data objects, specifying required data and URL/URN's would suffice to capture the essence of the composite net interaction. However, the variability of Internet resource accesses and the scope of what URAs could accomplish in the net environment seem to suggest the need to give the creator some means of organizing the instantiation of the component URL/URN's. Thus, the body of the URA should contain a scripting mechanism that minimally allows conditional instantiation of individual URL/URN's. These conditions could be based on which (content) data elements the user provided, or accessibility of one URL/URN, etc. It also provides a mechanism for suggesting scheduling of URL/URN instantiation.

WHAT-THEN: Once the URL/URN's have been instantiated, there will need for some post-processing in order to provide URA results in the format required by the URA type. For example, a URA built to deal with list-server mailing list subscriptions might belong to a class of net-services URAs that return simple success/failure messages.

The needs of post-processing are likely to be many and varied. The current proposal is to let creators specify a full-fledged scripting language (*e.g.*, Perl) to give them the maximum flexibility in creating tools to extract necessary information from the results of URL instantiations.

These URA objects fit into the context of a URAgency which applies basic methods (execute, identify individual components of a URA, etc). This structure is independent of the implementation of the URA itself.

Using URAs

The basic functionality an invoker can expect from a URA system is as follows:

- to be able to pick a URA to be activated (*i.e.*, identify an activity to be carried out)
- to specify whatever run-time information is needed by the URA

sound: it is the invoker that has an inherent interest in promoting the successful completion of the agent's work, therefore it is the invoker that provides that environment and the computing resources.

By accessing Internet resources through standard, published protocols, the URA approach means that information systems development is appropriately partitioned – new information systems can be developed with URAs without requiring the creation of a dedicated protocol for each type of system.

The fact that the agent itself does not roam is not limiting to its ability to compile information from multiple remote providers. Instead of having to move itself from one site to another, potentially encountering differences in environments, the agent calls on multiple remote sites to access their resources directly.

Discussion of agents is often linked with the issue of security. This is primarily of concern when the agent is being executed at a site or by a user other than that of the original creator of the agent, but it can ultimately affect any agent. The usual proposal to deal with security aspects is to provide the agent with an agency – an environment in which to operate. Although this does not solve all potential security issues, it does provide more control over the extent to which damage can be done (intentionally or inadvertently, by accessing inappropriate information, by perturbing the computing environment, etc).

In the model presented by this work, the agency is the environment that the invoker provides for the use of the agent. In addition to providing computing resources for the agent, it also means that any security violations will affect the invoker's environment, not the resource provider's. Again, it is a question of economics, and situating the cost of executing the agent as closely as possible to the invoker.

5 An Overview of URA Object Structure

This section outlines the necessary components of agent objects in order to encapsulate, maintain, and act on knowledge to carry out composite Internet information tasks. Tasks are described as "composite" if they require the interactions with one or more information resources, and/or if they require transformation of this information.

The purpose of the Uniform Resource Agent is to allow the encapsulation of network resource particulars, so that users (and user programs) can specify requests at a high level. These are specified as objects, with a script component, to provide a powerful yet manipulable construct. The proposed objects are "agents" because they are delegated to perform some action on behalf of an invoker.

URA structure

The proposed object structure to support this functionality is based on 6 parts:

ID: Identification of the URA object, including a URA name, type and abstract, creator name, resources required by the URA, etc.

4 Underlying Principles of URAs

Interaction Model

The interaction model supported by the URA architecture has three primary components: an information need, an information processing task or activity, and information resources. That is, a client's information need can be satisfied by some processing task which will draw on specific resources. In the current Internet environment, the bulk of the processing associated with satisfying a particular need is embedded in software applications (such as WWW browsers), and these applications access information resources directly through a fixed set of communications protocol standards. In moving towards more complex information environments, the temptation is to focus on developing distributed information resources, accessible by protocols of greater intricacy than those existing today.

Instead, through the use of URAs as client agents, the client application is responsible for identifying the information activity, the Internet resource servers are responsible for optimizing responses to specific queries, and the agents form a middle layer of activity that supports high-level information tasks by processing the results of specific queries to address the information need. This situates activity expertise with the activity itself.

This model is evolving in the experimental work done with URAs. A URA object is constructed to embody a particular Internet resource activity. To satisfy an information need, a client invokes a URA instance, supplying it with details to characterize the particular information need. Embedded within the URA is the necessary specification of Internet resources to access to satisfy the request. The important step is that the client provides information about its *need*, and delegates the processing to the URA. It is as irrelevant to the URA how the client software obtains this information (querying a human user, extractions from a user model, reading the phase of the moon, the state of the network, or any combination of the above) as it is to the client how the URA carries out the task.

URA technology is a first step in exploring the concept of keeping client information processing distinct from information resource processes, while providing an interaction model to satisfy increasingly complex information needs.

Situating Resources

As agents, URAs are tasked with carrying out activities on Internet information resources, in order to provide a higher-level abstraction of available resources. An underlying assumption, then, is that providers of resources (documents, services, etc), will make them public through standard Internet protocol mechanisms (*i.e.*, accessible by URLs, published as services, etc). Thus, a client should not need to deposit a process on the provider's machine in order to access any aspect of the resource that the provider intends to make public. (Note, too, that "public" is a relative term – it may mean the world at large, or a set of subscribed users, etc).

Therefore, URAs are meant to be executed locally to the invoker, not the information provider. The URA does not need to roam throughout the course of its execution. Instead, it accesses remote resources through published protocols, processes information within its own computing environment, and generates results for its invoker. This approach is economically

As a proposed solution, this paper describes Uniform Resource Agents (URAs) [URA-ID, MIDWARE], agent objects which encapsulate Internet resource activities. A fully-instantiated URA carries out a task delegated by an invoker (human or otherwise). The nature of the task is determined by the agent that the invoker instantiates; that originating object encapsulates some knowledge of existence of relevant Internet resources and information required in order to access them. In this way, URAs can be used to allow invokers to carry out high-level Internet resource activities while insulating them from the details of Internet protocols, etc. Also, by formally specifying a high-level Internet activity in an agent, the activity can be repeated by the same or another invoker at a later date, the activity can be incorporated into a still higher-level activity, the agent object can be modified to carry out a related task, etc.

As a very simple example, consider the client task of subscribing to a mailing list. There are many mechanisms for providing the user information necessary to complete a subscription. Currently, all applications which provide the ability to subscribe to mailing lists must contain net-aware code to carry out the task once the requisite personal data has been solicited from the user. Furthermore, any application program that embeds the ability to subscribe in its code necessarily limits the set of mailing lists to which a client can subscribe (*i.e.*, to those types foreseen by the software's creators). If, instead, there is an agent to which this task can be delegated, all applications can make use of the agent, and that agent becomes responsible for carrying out the necessary interactions to complete the subscription. Furthermore, that agent may be a client to other agents which can supply particular information about how to subscribe to new types of mail servers, etc.

From a resource provider's perspective, two different library service providers can construct and ship these agent objects such that clients are unaware that the library software is completely different at each site. Thus, a single library-access software client can access any number of different library systems by using the systems' individual agents. Furthermore, the client software in this case may not be interacting with a human user. It could, in fact, be responding to the request of another agent. In this way, more complex information systems can be constructed from the basic agent object.

3 Other Internet-based Agents

A number of Internet-aware agent and transportable code systems have been released recently – Java [JAVA], TCL [TCL] and Safe-TCL, Telescript [TELE], and the TACOMA system [TACO], to name a few of them. Some of these systems, like Java, focus on providing mechanisms for creating and distributing (inter)active documents in the World Wide Web. Others, like TACOMA, have more general intentions of providing environments for mobile, interacting processes.

While each of these systems makes its individual contribution to solving the transportation, communication, and security issues normally associated with agent systems, they are just more objects that exist within the Internet information space. That is, while they may permit individual users to have a more sophisticated interaction with a particular information resource, they do not address the more general Internet problems of naming, identifying, and (re)locating resources. It is this set of problems that URAs specifically set out to address.

Agents for Internet Information Clients

Leslie L. Daigle
Peter Deutsch
Bunyip Information Systems, Inc.

September 22, 1995

1 Abstract

This paper presents exploratory work on the use of agent-oriented techniques to provide sophisticated Internet information access and management. While related to general agent-based approaches, the project's design decisions have respected the nature of Internet information resources and interactions. The proposed agent technology focuses on the necessary information structures to encapsulate Internet activities into objects that can be activated, transformed, and combined into larger structured activities. The paper introduces the general problem area and the proposed solution, discusses relevant related agent systems, and outlines particulars of the proposed Internet Uniform Resource Agent architecture.

2 Introduction

The evolution of Internet information systems has been characterized by building upon successive layers of encapsulated technologies. Machine address numbers were devised, and then encapsulated in advertised machine names, which has allowed the evolution of the Domain Name System (DNS) [RFC1034, RFC1035]. Protocols were developed for accessing Internet resources of various descriptions, and then uniform mechanisms for specifying resource locations, standardized across protocol types, were developed (URLs) [RFC1738]. Each layer of Internet information primitives has served as the building blocks for the next level of abstraction and sophistication of information access, location, discovery and management.

The work described in this paper aims to take the current level of Internet information abstraction and make the next step in encapsulation. While TCP/IP protocols for routing, addressing, etc, have permitted the connection and accessibility of a plethora of information services on the Internet, these must yet be considered a diverse collection of heterogeneous resources. The World Wide Web effort is the most successful to date in attempting to knit these resources into a cohesive whole. However, the activity best-supported by this structure is (human) browsing of these resources as documents. It does not adequately address more abstract activities, such as information management, information representation, or other processing of raw information.