Agent services for users of digital libraries

J. Alfredo Sánchez* and John J. Leggett†

*Department of Computer Systems Engineering, Universidad de las Américas-Puebla, Cholula, Puebla, 72820 México, alfredo@udlapvms.pue.udlap.mx and †Center for the Study of Digital Libraries, Texas A&M University, College Station, TX 77843, USA, leggett@csdl.tamu.edu

This paper describes an architecture for digital libraries that introduces the notion of agent services. Agents are a powerful abstraction through which users may interact with a complex data universe by delegating tasks involving objects or events of interest. In the proposed architecture, termed AGS, providers of information define agent services to which patrons may subscribe. On the server side, classes of agents are specified that describe automated tasks to be carried out based on both the digital library information resources and the patrons' information needs and personal preferences. On the client side, patrons assign and control tasks by instantiating agents of available classes. We present a prototypical implementation of the agent services architecture that demonstrates the major concepts introduced by AGS. We also discuss the applications of AGS and the potential of agent-based user interfaces as an alternative style of interaction with digital libraries.

1. Introduction

In the complex and chaotic scenario presented by current information technologies, the emerging field of digital libraries promises the construction of well-organized enclaves that will support users (digital library patrons) in diverse information-intensive activities. Remote collaborative scholarly work, multiple conceptual mappings, and personalization of library resources are but a few of the possibilities that have been suggested as potential benefits of digital libraries. In the development of this vision, a pressing research issue is the construction of interfaces that bridge the gap between the user's information needs and the resources that digital libraries will make available. A great need exists to provide users with means for coping with the volume, dynamism and complexity of this data universe that is within their reach.

In this paper we present an architecture for digital libraries that introduces the notion of agent services. Agents have been identified as a prominent issue in the research agenda for digital libraries [1,2]. An agent (or user agent) in this context can be regarded as a semi-autonomous process that is perceived by the end user as performing a task on his or her behalf. Agents are a powerful abstraction through which users may interact with a complex data universe by delegating tasks involving objects or events of interest. In the proposed architecture, providers of information (publishers, librarians) define agent services to which patrons may subscribe. On the server side, classes of agents are specified which describe automated tasks to be carried out based on both the digital library information resources and the patrons' information needs. On the client side, patrons assign and control tasks by instantiating agents of available classes.
The paper is organized as follows. Section 2 briefly introduces the notions of user agent and agent-based interfaces. Section 3 describes the context and the components of the architecture for agent services, which we have termed AGS. A prototype that demonstrates the major concepts of AGS is presented in Section 4. Applications of the architecture are discussed in Section 5. Finally, the current status of the project and the work to be undertaken in the future are presented in Section 6.

2. Agent-based user interfaces

The notion of agent has become popular as an abstraction to describe complex systems or interactions among system components or participants. In fact, digital libraries have been proposed that are entirely specified in terms of agents (see, for example, [3] and [4]). However, no agreement has been reached on the definition agent or the characteristics an agent should exhibit. In this paper we follow the views of agency presented by Sánchez [5], according to which three major classes of agents can be distinguished: programmer agents, network agents and user agents.

Programmer agents are animistic abstractions (involving entities with intentions) used by software developers in conceptualizing, describing, analysing and predicting the operation of software systems (e.g. [6–9]). Network agents (also referred to as mobile agents) are abstractions used in modeling the communication among computers in a network, providing an alternative to the commonly used remote procedure calls. With network agents, communicating computers can supply the code of the procedures to be executed by remote machines. Each executing procedure is considered an autonomous agent that may decide to migrate to a different host in the network according to its task requirements (see [10–13]). Finally, user agents (or user interface agents) are animistic abstractions made available to the end user in the interface to computer systems. Users interact with their agents by delegating tasks that agents perform using knowledge on the user’s preferences and needs. User agents are considered one of the research frontiers in human–computer interaction [14]. Work related to user agents has been surveyed in [5] and includes [15–19]. Interfaces that allow users to view their interaction with computer systems as the delegation of tasks to and management of semi-autonomous processes will be referred to as agent-based user interfaces.

The main interest of the work presented in this paper is the potential of agents in assisting users in taking advantage of the rich, yet complex and dynamic, information universe enabled by digital libraries. User agents promise to contribute to a more lively, personalized and cooperative work space in the digital library. The tasks that user agents will carry out in a digital library will be varied. Agents will notify their users when library items of interest are added or updated, filter retrieved information according to the user’s needs or preferences, or handle routine administrative procedures in the library (such as copyright and billing procedures). Agents will guide new patrons around the library, providing hints based on their knowledge of the library, observed usage by other users, or by contacting other users (or their agents) for assistance. In order to realize these scenarios, however, much work needs to be done. Among the major research issues in agent-based user interfaces that have been identified [20,21] are: the representation of agents in the user interface, the need to provide mechanisms to guarantee the control of agents by the user; the use of existing active features in current systems to build agent functionality; programmer and end-user extensibility;
and the relation of agency with other user interaction styles. The architecture for agent services presented in the following sections represents an effort aimed at exploring these issues and facilitating the investigation of agent-based user interfaces for digital libraries.

3. An architecture for agent services

Though diverse architectures have been proposed, there seems to be a tacit consensus regarding many of the general characteristics digital libraries should exhibit [3,22–26]. It is clear that the main purpose of a digital library is to provide information services to patrons in highly distributed environments. Figure 1 shows a simplified representation of a general architecture for a digital library. Components include server nodes on which publishers and librarians maintain library materials and service programs, client nodes from which librarians have access to functions available for resource administration, and client nodes from which patrons access library services. Repositories on the server side will typically rely on advanced database management systems (DBMSs) for storing, indexing, and selectively retrieving library objects. A number of translation layers are possible between the basic storage substrate and the abstractions handled by the various library services.
Based on this general architecture for digital libraries, we have designed AGS, an *Agent Services Architecture*, which enhances current views on digital libraries by making agents available to patrons along with existing library services. The major components of AGS (Fig. 2) are briefly discussed in the following subsections (a detailed specification of the architecture is available in [5]).

### 3.1 AGS abstractions

In order to provide digital library users with the notion of *agent*, AGS introduces five abstractions into the base architecture: *agent classes*, *agent instances*, *agent actions*, *user profiles*, and *agencies*. An *agent class* defines behaviour for user agents, whereas an *agent instance* is associated with an agent class every time a user delegates a specific task to a given agent. *Actions* are objects used as building blocks in the construction of agent functionality. A *user profile* is associated with each AGS user and contains arbitrary attributes that describe the user’s interests and preferences. *Agencies* refer to other AGS-enabled digital libraries known to a given AGS server.

In AGS, each of these abstractions can be regarded as objects with specific attributes that are used by the various AGS subsystems to facilitate the assignment and control of tasks to user agents in the digital library. For example, attributes for an agent class include a class name, a textual description of the general behaviour it implements, references to actions implementing the agent functionality, actions implementing specific...
user-agent interaction and, if applicable, actions that undo agent operations. The language in which the agents’ actions are defined and the parameters required by each of the actions are also agent class attributes. Agent instances are generated on server or client nodes based on corresponding agent class attributes. Attributes for an agent instance thus include an instance name, the name of the agent class to which it belongs, the identifier for its owner (the user delegating a task), a host machine identifier, a process identifier, and optional actions that will be performed in case of emergency (e.g. send e-mail to the user upon connection loss or recovery from system failure).

3.2 AGS components

The server side of the base digital library architecture is augmented by a new service module called the Active Library Service (ALiS), which utilizes existing digital library infrastructure to provide functionality for managing the five AGS abstractions. Two other new components interact directly with ALiS: the User Agent Manager (UAM), which provides functionality for the administration and tuning of AGS resources; and the User Agent Director (UAD), through which end users define and control agents in the digital library. Although Fig. 2 shows only one library server along with a patron client and an administration client, in AGS it is possible for multiple users to access the library from a single client, each of them having the capability to use AGS services by starting a UAD process. A single UAD may also contact multiple library servers and their corresponding Active Library Services. Finally, user agents performing a task on the server or on a client node may also utilize ALiS functionality (e.g. to obtain or update user profiles) and may communicate among themselves. This slightly more complex scenario is illustrated in Fig. 3, in which, for simplicity, only AGS server and
patron components are displayed in addition to user agents and object repositories.

In order to delegate a task to an agent in the digital library, a patron selects one of the available agent classes via the UAD, which sends an ‘instantiation’ request to ALiS. ALiS retrieves the code for the actions implementing the corresponding agent class and, depending on the class attributes, allows the user to provide required parameters and starts execution on the server or the client. This execution may involve the creation of a new process, the definition of rules in the underlying database, or passing the code to an available interpreter, since the agent’s actions may be implemented in any language known to the AGS servers or clients. Once missions have been delegated, users can interact with their agents via the UAD to suspend or resume tasks, inspect their current status, or permanently dismiss any of the agents. The UAD contacts ALiS with the appropriate request and ALiS in turn sends a message (sleep, awake, terminate) to the involved instances and activates or deactivates associated rules in the database.

3.3 TAGS: a toolkit for AGS

In order to provide an extensible agent environment, and to facilitate the development of new agents in the AGS digital library, we have designed a toolkit for agent construction called TAGS. TAGS allows the agent developer to specify agent functionality using high-level abstractions involving AGS actions, agent instances, agencies, user profiles, digital library objects and DBMS features such as rules and event generators. For example, using agency management functions, agent instances can obtain information such as addresses and attributes associated with other sites offering AGS services. Similarly, agents can get contact information for other instances (local or otherwise) by utilizing instance management functions. By using TAGS, the programmer does not need to know specific details of the DBMS underlying the digital library or the communication mechanisms being employed among the system components to implement agent functionality.

4. Prototypical implementation

In order to assess feasibility and applicability, we have developed a prototypical instance of our architecture for agent services. This prototype, which we have termed AGS-1, implements the major features and components of the AGS architecture, as well as some experimental agents. To facilitate the construction of agents we have also implemented the main primitives included in the design of TAGS. We have called this version of the agency toolkit $TAGS-1$. The base digital library architecture for AGS-1 is provided by ongoing research at Texas A&M University’s Center for the Study of Digital Libraries (CSDL). For example, one of the prototypical libraries under construction at CSDL is the Bush Presidential Digital Library. We describe the context and features of AGS-1 and TAGS-1 below.

4.1 The context of AGS-1 and TAGS-1

Figure 4 shows the context in which AGS-1 is embedded. Prototypical digital libraries at CSDL utilize Illustra, an ‘object-relational’ DBMS [27], as a basic object storage substrate. The abstractions handled by the various digital library services are all
implemented in terms of simple and composite objects with arbitrary attributes. The mapping between the abstractions handled by the library services and the DBMS’s objects and relations is performed by a locally developed server called the Object Manager (OM) [28]. Services other than user agents and their abstraction mapping requirements are documented elsewhere [29,30]. One of the access means to CSDL digital library services is the popular HTTP protocol, which is enabled by a server that also acts as an interface between client application and other services. By using HTTP and the Common Gateway Interface (CGI) [31], AGS-1 resources can be referenced using the Uniform Resource Locator (URL) scheme [32], and widely available WWW browsers and HTML forms can act as client interfaces. *H*, a general toolkit for inter-process communication [33,34] used throughout the CSDL libraries, can be used to specify arbitrary protocols for inter-agent communication.

We have used C with embedded SQL code to implement server-side functionality and access to Illustra databases in AGS-1 and TAGS-1. This functionality can be accessed by librarians and patrons via widely available web browsers. Figure 5 illustrates the interface to the UAM functions as rendered by one such browser, whereas Fig. 6 shows the form used to add a new agent class after selecting that option in the main UAM menu.

While browsing a library, patrons will find links to potentially useful AGS services. Alternatively, users can invoke the UAD directly. In either case, after providing a user identifier and a required password, the UAD presents the user with a list including any agents that have been assigned tasks previously and a list of available classes of agents

---

**Figure 4.** The AGS-1 instantiation of the architecture for agent services.
to which the user may delegate new tasks. From this interface, illustrated in Fig. 7, end users can exercise several control options over any of their agents in the library. In this example, the agents displayed have been assigned tasks in the context of the Bush Presidential Digital Library. ‘Bush Home Watcher’ is an active agent of class *Page Change Monitor*, the definition of which was shown in Fig. 6. Two other agents, of class *Topic Expert*, have also been assigned tasks, and one of them has been temporarily suspended. In order to delegate a task to a new agent, the user clicks on one of the available agent classes and the UAD presents a form such as the one displayed in Fig. 8, which illustrates the definition of the ‘Bush Home Watcher’ agent instance. When in action, this agent will inform the user of any changes to the library’s home page and will provide a link to the corresponding URL for the user to examine.

5. Applications

The Agent Services Architecture and its AGS-1 prototypical implementation offer an open environment that can function as a testbed for exploring agent-based user interfaces to digital libraries. By making the *agent* abstraction available to both librarians and patrons of digital libraries, AGS fosters the development of a cooperative, highly synergistic system. As access facilitators, librarians can use their expertise and knowledge about the library offerings in providing helpful agent classes and suggesting potential
applications for agents. Patrons, on the consumer’s side, can utilize the library resources using a variety of available interaction styles, including the possibility of delegating tasks to be performed by agents. Librarians may suggest potential applications for the agents specified using the UAM, or patrons may define agents instances by directly using the UAD. Useful agents are thus the result of a distributed, evolving collaboration between librarians and patrons, which is mediated by the AGS active environment that provides services for agent construction, customization and operation.

Although AGS provides a set of primitives to facilitate the utilization of agent-related abstractions in the construction of new agents, the proposed architecture does not impose any specific structural or behavioural requirements upon user agents. AGS agents do need to comply with a very simple communication protocol to interact with the ALiS server (by implementing meaningful behaviours in response to user control messages and notifying ALiS of status or host changes), but developers are free to employ any tools or algorithms in the implementation of user agents to be incorporated into the AGS environment. Machine learning, information retrieval, programmer or network agents, and programming by demonstration are but a few of the possible areas
from which developers can draw techniques to construct agents with diverse functionality and applications.

The AGS-1 prototype shows the applicability of the proposed architecture by demonstrating that agent-based user interfaces are feasible with currently available technologies. Existing active features offer great potential for enhancing agent capabilities and transforming a passive environment into a personalized, cooperative system. Existing active features such as database rules and alerters, typically used to support programmer tasks (e.g. enforcing referential integrity), are made available by AGS for direct utilization in end-user applications. More conventional database functionality, such as transaction management and crash recovery, are used in AGS to guarantee consistent manipulation of persistent objects and resilient agent operation.

In the design of the AGS architecture we have placed particular emphasis on including provisions to promote control of agent tasks by the end user. Because of the semi-autonomous nature of agents, this issue has paramount importance and has been the subject of much concern and debate among researchers in human–computer interaction [35,36]. In AGS, an active agent can be summoned at any point in time by the user and its status can be inspected to verify mission progress or to enquire about the rationale or precision of particular actions or suggestions. Agents can also be permanently dismissed, temporarily suspended or their tasks resumed at the user’s will.
6. Future work

The agents that have been implemented to demonstrate the concepts introduced by AGS are still quite simple. Much effort is now being focused on producing more sophisticated and more useful agents with diverse functionality. For example, an area that requires particular attention is the provision of agents to assist users in the construction of a digital library. Populating repositories with millions of objects is a task of enormous proportions. A project to develop an adaptive class of agents that will use domain-specific knowledge and existing repository objects to save data entry work is now under way. Another class of agents is under development that will offer alternative views on library objects; this is an essential requirement for patrons of libraries in areas such as systematics and taxonomy. As digital libraries evolve and users begin to explore their facilities, we expect to be able to detect new requirements and potential applications for agents.

Indeed, work will continue to produce a more complete implementation of the AGS architecture, particularly addressing the shortcomings of the current AGS-1 implementation. We will support more fully the definition of agent functionality in multiple languages. AGS-1 supports only dynamically loadable objects and takes advantage of Java-aware web browsers to allow for agent definitions in this language. However, due to security provisions, user agents written in Java and executing on the
client side are not allowed to access non-Java functions. This makes it impossible for the developer to take advantage of the TAGS agent construction toolkit, since the current version (TAGS-1) is written in C and is available only as an object library. Another language we intend to support is Telescript [13], which will facilitate the utilization of network agent technology in the construction of user agents. Finally, we will also be working on extensions to the current prototype to take agent extensibility one step further and allow end users to build their own agents using pre-defined, re-usable actions. These extensions will build on previous work on a related project [21].

One of the major motivations driving the design of the Agent Services Architecture has been the belief that user interface considerations are as crucial as other aspects of the development of digital libraries such as storage, retrieval and validation of library elements. We believe that agent-based user interfaces will play a key role as an alternative interaction style with digital libraries. By making user agent provisions part of digital libraries from an early stage in their conceptualization and construction, we expect that user needs and technological advances will drive the development of more useful and viable agent-based user interfaces for digital libraries.

References


33. P. Nünnberg 1994. Implications of an open, extensible, and distributed hypermedia information system architecture for inter-process communication subsystem design. M.S. Thesis, Department of Computer Science, Texas A&M University, College Station, Tex.
34. P. Nünnberg and J. Leggett 1996. The HOSS inter-process communication facilities. Submitted to *Journal of Network and Computer Applications* (Available from the authors: pnuern@csdl.tamu.edu).