Agent Technology concepts in a heterogeneous distributed searching environment

by Valeda Frances Dent, Wendy Hall, Stephen Harris, Jessie Hey, Kirk Martinez

Hybrid libraries provide multiple ways to access information in various formats, normally within a common information framework. The eLib project MALIBU (MAnaging the Hybrid LIbrary for the Benefit of Users) focuses on the development of models, both prototypic and theoretic, for management and organisation of the hybrid library. This article describes the agent technology used for the MALIBU prototype search engine that allows for the search and retrieval of information from disparate resources.

An overview of the MALIBU project

MALIBU is a three-year Electronic Libraries Programme (eLib) project that has developed and implemented prototype hybrid libraries in each of three major partner institutions in the United Kingdom. In doing so, MALIBU has developed general institutional models that can be utilised throughout the higher education community. The project is led by King's College London with partners in Oxford University and the University of Southampton.

The MALIBU project focuses on the Humanities disciplines. Through co-operative resource-sharing it has provided innovative and cost-effective ways to meet the ever-increasing information requirements of staff and students. The term 'hybrid library' refers to providing effective access to both digital and non-digital resources within a common information framework. It can be defined as integrating access to all four different kinds of resources (legacy, transition, new, and future), using different technologies from the digital library world, and across different media. (Rusbridge, 1998).

The project has examined a range of issues, and developed models for organizing and managing the hybrid library, based on a service framework sufficiently flexible to allow non-disruptive introduction of future technical developments.

The development of the MALIBU search engine prototype and its use of agency form one component of the project. This article provides background on the decision to use agents and its impact on the development and functionality of the search engine.

Basic MALIBU search engine prototype functionality

The MALIBU search engine prototype is a search mechanism consisting of a single, web-based interface providing access to institutional and external resources, referred to as targets. This includes multiple types of digital data, (especially structured metadata), images, video, sound, large text bases available in a number of formats, and on-line public access catalogues (OPACS).

The prototype allows for asynchronous, prioritised synchronous and linked searches, and accesses both Z39.50 and non Z39.50-compliant catalogues. The search engine utilises an agent to communicate with each target, using the relevant protocol (HTTP or Z39.50) to search and retrieve information for the user. The goal is to allow the end-user the convenience of searching in a single search event on-line databases, web resources, internal and external resources. The search engine prototype allows users flexibility in managing their information environment. Its features include keyword, author, and title searches; profile management, which allows the user to determine and select those targets which might be most suited to their needs, and the facility to export results via email, HTML, plain text, RTF or RDF. A help feature is also available to tackle FAQ's. Figures 1,

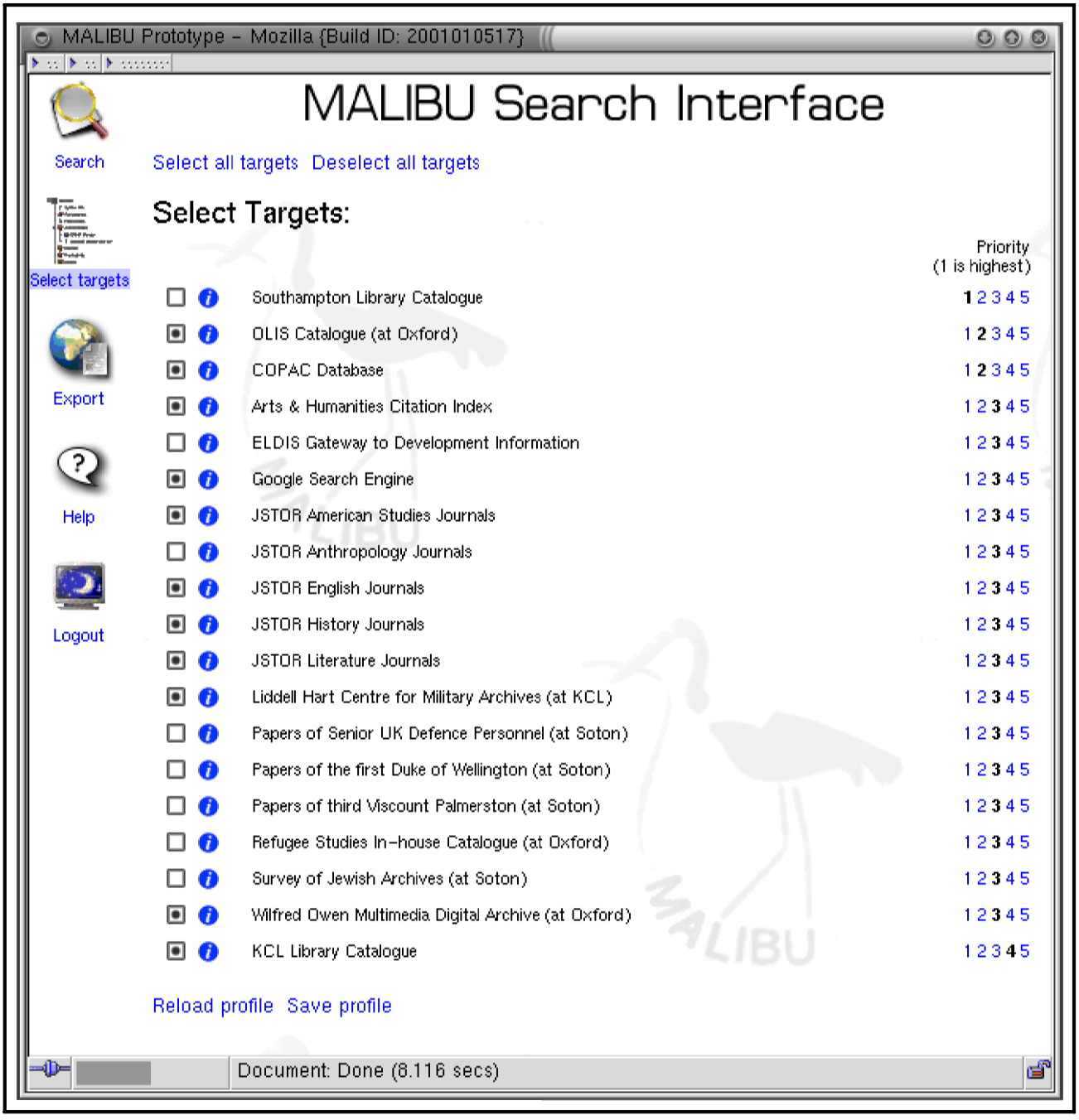

Figure 1 – The MALIBU search engine prototype interface

2 and 3 illustrate the simple interface that was designed to support user testing of the prototype.

In Figure 1, users specify the targets they want included in the search before beginning a search.

In Figure 2 users indicate the search parameters and how they would like their results returned.

In Figure 3 (overleaf) results of the search are returned as specified by the user. The user can follow links to access resources as well as export the results in a variety of different ways.

A general overview of Agent Technology

The concept of agent technology has wide-ranging application within engineering, computing and information science. An agent is an encapsulated computer system, situated in a particular environment, that is capable of flexible, autonomous action in that environment in order to meet its design objectives (Wooldridge, 1997). Many of the systems where agents are at work are extraordinarily complex, understanding what tasks are being undertaken and the systems' internal state can be

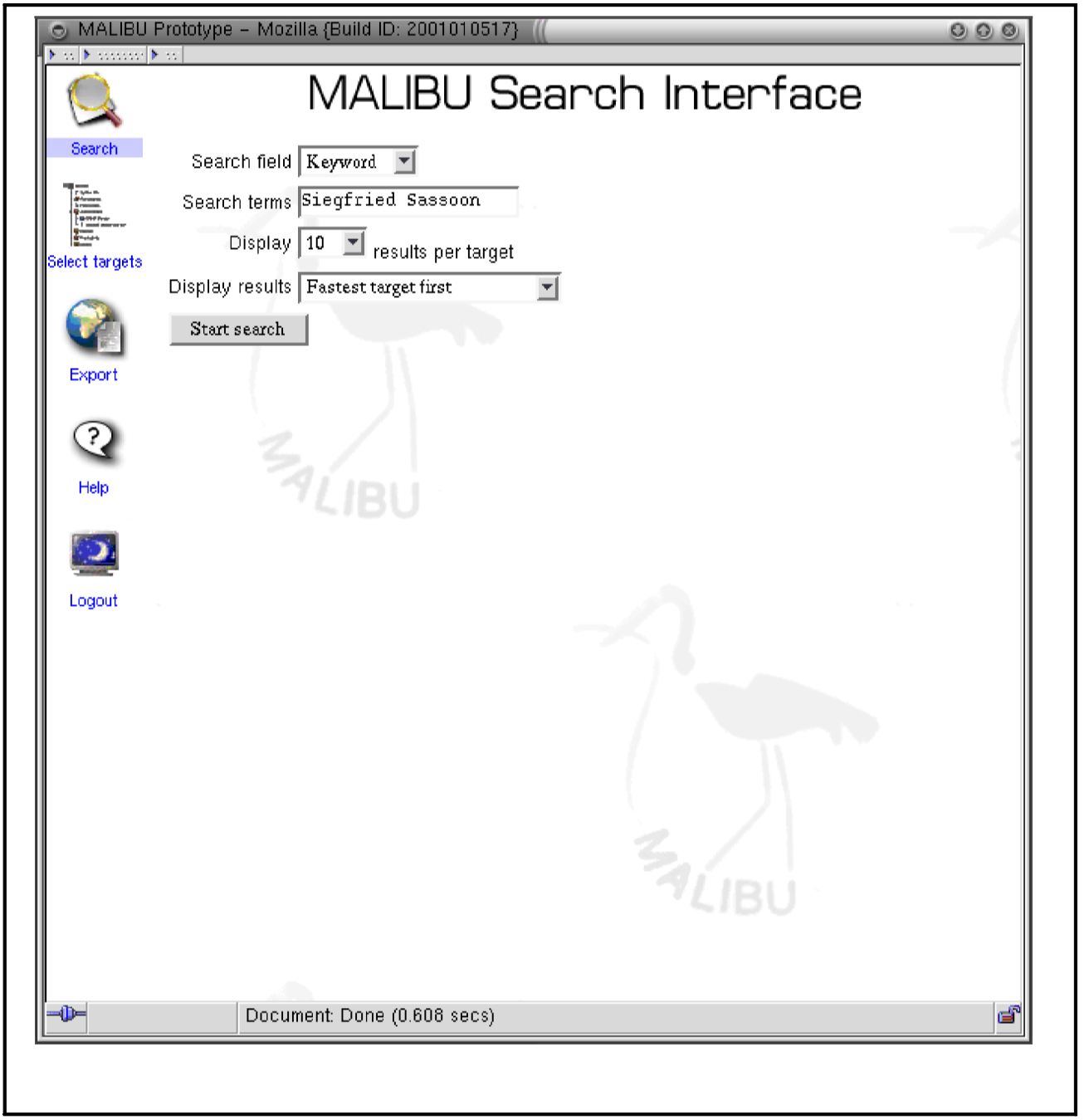

Figure 2 – The MALIBU search interface parameters

difficult. One of the benefits is that it provides a helpful, abstract way of talking about complicated distributed systems using naive psychological terminology.

Agents can perform a variety of tasks, and this flexibility is one of the many strengths of the architecture. Generally, agent architecture may be responsible for the four functions of observation, recognition, planning and/or inference, and action or execution (Miller, 1997). For the complex agency system, personalising agents is common practice. For example, mediator agents, communication agents and query-planning agents (Birmingham, 1995) are easily recognisable term-descriptions for a complex set of actions and processes. Agents can translate, communicate and publish information, as well as to guide the search process by taking the users' query from the interface to the appropriate target. Agents can also negotiate for access to, and the exchange of, information with other agents. This exchange is akin to a conversation that allows agents to determine which tasks will be performed within the context of pre-determined goals. The specialised properties that agents possess allow for the effective management of the users' information environment.

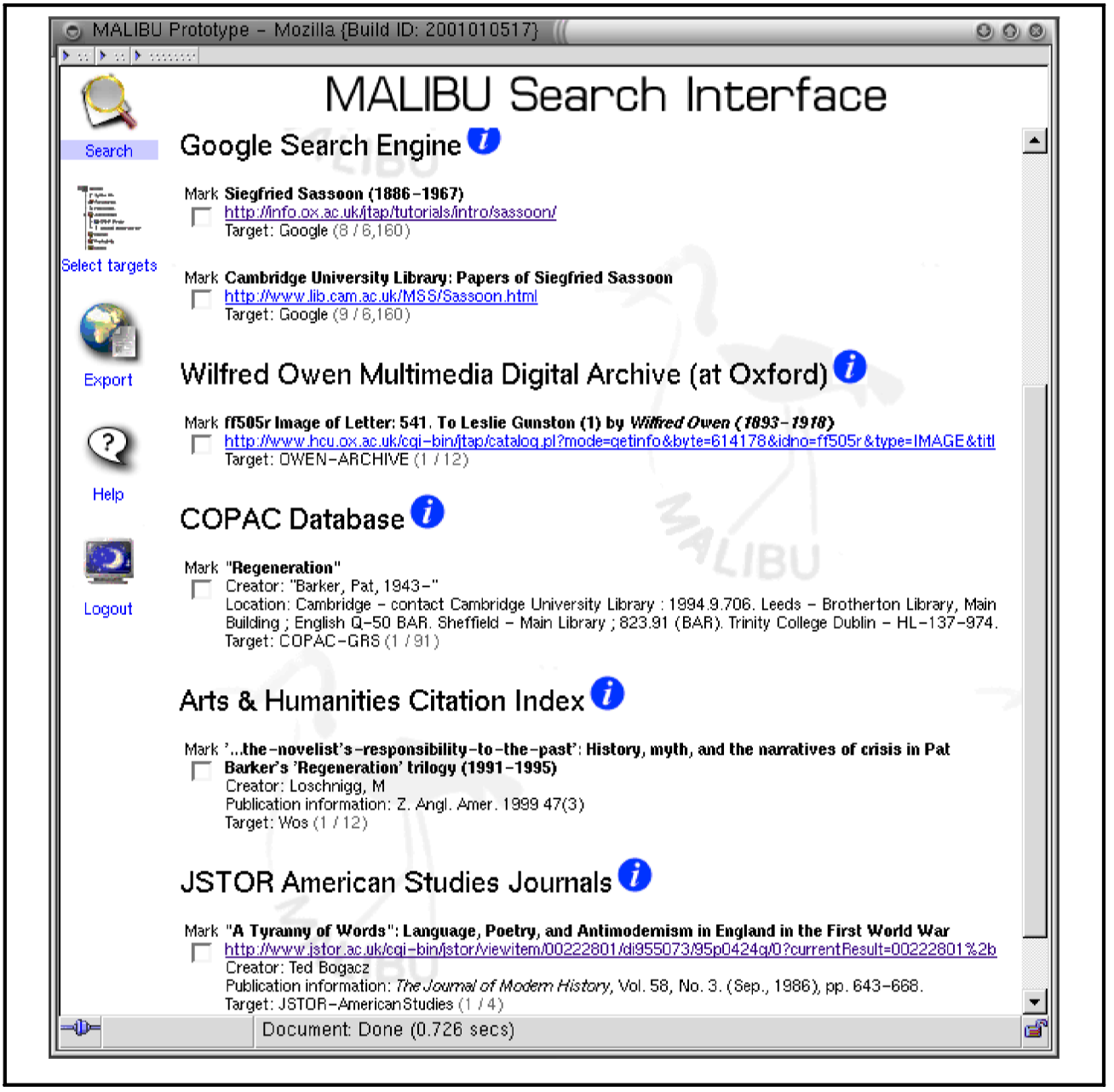

Figure 3 – The MALIBU search interface results

Benefits of Agent Architecture for MALIBU

The MALIBU search engine prototype, GIGA (Global Information Gathering Architecture) uses query agents to communicate with each target, using the relevant protocol to search and retrieve information for the user.

Figure 4 illustrates the graphical representation of the data flow and logical clustering for the GIGA system.

The meta-agent is the central communication point and acts as an information broker to facilitate communication between agents. It optimises this communication by routing the messages to the agents to which it is immediately relevant. This is done with the use of "clouds" of agents and could in theory narrow the communication to the agent level.

There are several intrinsic and highly specialized features that agents possess which proved beneficial in terms of the MALIBU search engine prototype. They were:

- Human ability to reason about system;
- Agent ability to interact with each other within the system;

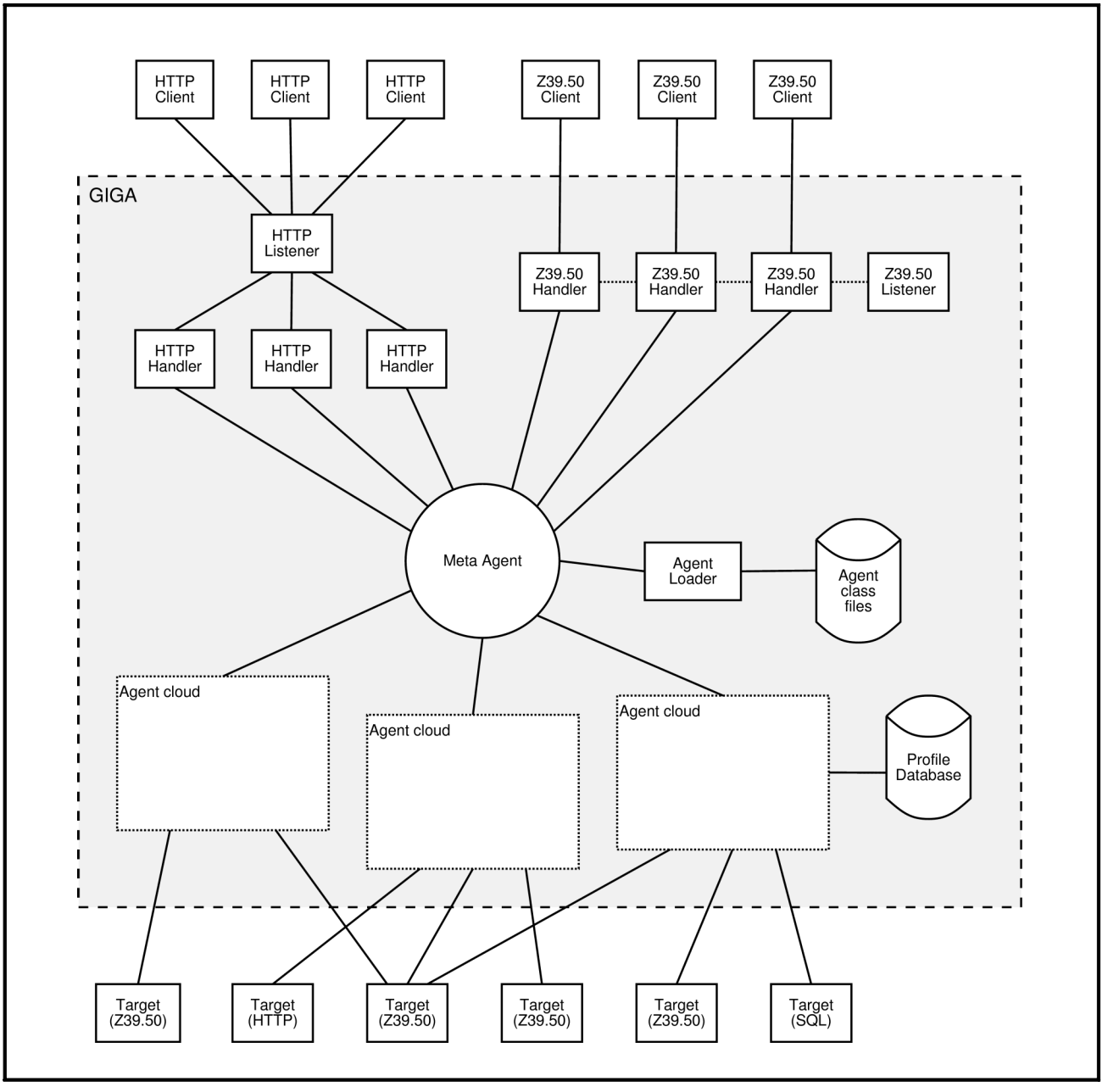

Figure 4 – Diagram showing data flow and logical clustering

- Ease of distributed development;
- Ease of system maintenance;
- Internal feedback and human agency.

Human ability to reason about system

The use of agency provides a reasonable framework that maps well with our mental model of the world. The use of the agent as a metaphor allows programmers to understand and work with otherwise complex concepts and to describe internal tasks and processes in a way that is easier to comprehend.

Agent ability to interact with each other within the system

The MALIBU agents interact with each other, not just with the central system. If an agent decides to process a query, it may interact with another agent to achieve a common goal.

Ease of distributed development

The MALIBU search engine prototype demands communication between agents. This communication takes place by means of a common vocabulary and agreed upon meanings to describe a subject domain, or ontology. This ontology defines the

communications of the agent in a machine-readable format, and it also allows for a greater degree of flexibility than a traditional application programmer interface (API)[1]. An agent that wishes to know whether a given MALIBU search engine prototype user is allowed access to a given resource can express what it knows about both the user and the resource, and any agent that knows the answer can respond. As well, the agent responding to the query can utilize the information provided without affecting the code of the agent originally asking the question. This is a feature that allows each agent to function according to its own ability.

Ease of system maintenance

It was very important that the architecture of the MALIBU search engine prototype be dynamic, modular and scaleable. The use of this architecture allows agents to be introduced into the system at any time, without disruption to other processess or agents. This, in turn, means that those maintaining the system can increase its capabilities with minimum effort. It also limits the amount of disruption to the user. Wiederhold indicated the maintenance effort required to update one part of the code in response to an external change is in direct proportion to the square of the number of modules in the system (Wiederhold, 1995). For the MALIBU search engine prototype system, that number would be between 10 and 20 modules if it were a conventional system. As it is an agent-based system, the effort required is proportional to the data path length, which in this case is 4 — a noticeable difference in the amount of maintenance effort required (Figure 5).

Internal feedback and human agency

People, or 'human agents', responsible for maintaining the MALIBU software are contacted by email when agents suspect there may be a problem, and the resulting bug fixes become a part of the system.

This enhances the durability and flexibility of the system, whilst lessening the need for regular observation of system performance by the maintainers. This is especially helpful with HTTP targets that are prone to interface changes without notice, thus interfering with the correct operation of search mediators.

Implementation

Language and operating environment

The project selected Java as the development language for GIGA. Java provides a great deal of platform independence, which is important as the prototype must run on various UNIX implementations and Windows NT.

The object-oriented nature of Java was also essential in constructing a system with a reasonable amount of effort. The performance and ease of development of Java was a great advantage. Other

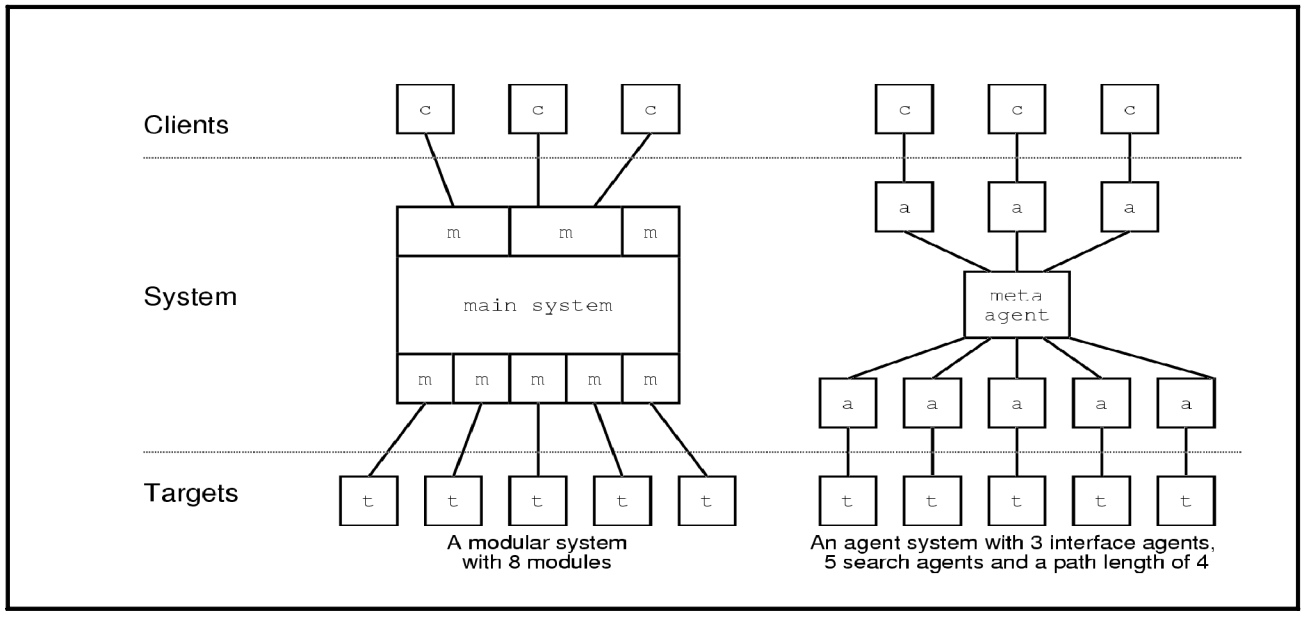

Figure 5 – Comparison of maintenance effort required

languages considered were Prolog and Lisp, but they could not provide Z39.50 and HTTP protocol libraries.

Although C or C++ might provide greater performance and equal protocol, the differences in binary format between UNIX and NT was a hindrance.

Architecture

The architecture chosen for the MALIBU search engine prototype is equivalent to the one that Wiederhold (1991) describes, but with the additional brokerage layer implemented. The additional layer, the meta-agent, facilitates resource control and intercommunication between the agents and provides a central point of control.

Ontologies and sub-ontologies

It was advantageous to use an agreed upon sub-ontology, or set vocabulary, in each domain discussed by the agents, with one overarching ontology representing the system as a whole. The sub-ontology for a given conversation is chosen so that it is the common subset of domains of all the agents involved in the conversation. This reduces interaction to a minimum and grants as much autonomy as possible (Wiederhold, 1995). To facilitate this, the ontology was implemented as a set of key-value pairs[2], with each key representing a meta-data element. This form of ontology was chosen to mirror classic meta-data representation, as most of the information in the ontology is meta-data. The remaining information (such as internal events, etc.) can easily be represented in that form.

Generally, ontologies are expressed using a knowledge representation language, often KQML (Finin, 1994). Much like Dublin Core[3], KQML defines the basic semantics, but not the syntax for ontology. An example using Lisp syntax is shown in Figure 6, it tells the agents called soton-opac and kcl-opac to search for the keyword "Eric Satie"and return the results in the Dublin Core format. The equivalent GIGA message would be:

```
Query {
     DC.Subject: Eric Satie
     Target.Name: soton-opac || kcl-opac
     Target.MaxReturnedRecords: 10
}
```

The key-value representation was chosen over more typical ontologies for the following reasons: overhead, ease of development, and simplicity

Overhead

Although the system resources were not a large concern, MALIBU team members anticipate that there will be a large number of agents within the system. These agents must respond rapidly to external and internal questions. Choosing a low level representation such as object hierarchy allowed for greater processing speed and construction of ontological representations of queries.

Ease of development

Many of the MALIBU project team members involved in the development of the GIGA were new to the application of agent technology. The key-value approach reduced the learning curve by

```
(stream - all
     :sender html-interface
     :receiver soton-opac
     :reply-with id1
     :language Prolog
     :ontology dublin-core
     :content "subject-matches ("Eric Satie", Results, X), X<11")

(stream-all
     :sender html-interface
     :receiver kcl-opac
     :reply-with id2
     :language Prolog
     :ontology dublin-core
     :content "subject-matches ("Eric Satie", Results, X), X<11")
```

Figure 6 – KQML example using Lisp syntax

using familiar software engineering data types and methodology.

Simplicity

It was determined that for the MALIBU system, with its relatively simple information requirements, typical universal ontologies were overcomplicated and made it difficult to represent meta-data. The key-value approach provided a manageable and straightforward solution.

Conclusion

The development and testing of the MALIBU prototype search engine concluded in January 2001. The prototype was well received across the three partner sites, and initial user feedback indicates the usefulness of this type of application. One conclusion is that although architecture is often transparent to the user, the impact of its practicality is undeniable.

The use of agency for the MALIBU project was both appropriate and efficient. Scalability and flexibility have allowed for ongoing development of the search engine prototype searching capabilities by a team of technical staff, located on three separate campuses. In many ways, the use of agent technology has allowed for maximum control by the user over their information environment. Agents interacting to guide a search from beginning to end are a basic example of the high levels of both inter- and independence and interoperability that agents possess. Potential developments in this area include the possibility of an international standard ontology and knowledge representation language. This would allow interoperability between agents from any compliant search system. It would also allow database vendors to develop and distribute their own agents optimized to exploit their resource.

There are many possibilities for the use of agent technology within the hybrid library environment, and just as many implications. Further research and investigation are needed to determine what might be most fiscally sound, practical, useful and beneficial to the end-user.

Acknowledgement

Special thanks to Professor Maurita Holland, University of Michigan School of Information for her help.

Notes

1. API is a tightly defined software interface which is normally used to communicate with another programmer's code

2. A key-value pair is a pair of data items, one the indexed term, and the other the contents. For example, a database of ISBN numbers against book titles.

3. Dublin Core is a set of metadata elements defined to facilitate the discovery of document-like items in a networked environment such as the Internet.

References

Birmingham, William (1995) An agent-based architecture for digital libraries. *D-Lib Magazine.*
Finin, T., Fritzon, R., McKay, D. and McEntire, R. (1994) KQML as an agent communication language. *Proceedings of the Third International Conference on Information and Knowledge Management,* ACM Press.

Miller, Marc. (1997) Software Agents. *CHI 1997 Electronic Publications.* http://www.acm.org/sigchi/chi97/proceedings/tutorial/mm.htm.
Rusbridge, Chris (1998) Towards the Hybrid Library. *D-Lib Magazine.*

Wiederhold, G. (1991). Mediators in the architecture of future information systems. *IEEE Computer Magazine* (25) 3: 38 - 49.

Wiederhold, G. (1995). Mediation and Software Maintenance. *Stanford CSD Technical Note STAN-CS-TN-95-26.*

Woolride, M. and Jennings, N. (1994). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* (10) 2: 115 - 152.

Contact details
Valeda Frances Dent MILS, MSW
Director, Residence Hall Libraries
Adjunct Lecturer,
School of Information
University of Michigan
USA
Tel: +1 (734) 764 1152
Fax: +1 (734) 764 1153
Email: ballet@umich.edu

Wendy Hall
University of Southampton
Highfield, Southampton
SO17 1BJ.
United Kingdom
Tel: +44 (0) 23 8059 2388
Fax: +44 (0) 23 8059 2865
Email: w.hall@ecs.soton.ac.uk

Jessie M. N. Hey
University of Southampton
IAM Group, ECS, Bldg 59
Highfield, Southampton
SO17 1SQ
Tel: +44 (0) 23 8059 3256
Fax: +44 (0) 23 8059 5791
Email: jmnh@ecs.soton.ac.uk

Stephen Harris
Department of Electronics and Computer Science
University of Southampton
Highfield, Southampton
SO17 1BJ
United Kingdom
Tel. +44 (0) 23 8059 6000
Fax +44 (0) 23 8059 5791
Email: s.w.harris@ecs.soton.ac.uk

Kirk Martinez
Department of Electronics and Computer Science
University of Southampton
Highfield, Southampton
SO17 1BJ
United Kingdom
Tel: +44 (0) 23 8059 4491
Fax: +44 (0) 23 8059 5791
Email: km@ecs.soton.ac.uk