

Agent-Based Services for Information Portals*

Ciarán Bryce & Michel
Pawlak
Centre Universitaire
d'Informatique
University of Geneva
Switzerland

Karsten Tolle & Peter
Werner & Roberto Zicari
Database Group
University of Frankfurt
Germany

ABSTRACT

Web portals have a huge importance on today's Internet. Their purpose is to serve as an entry point to information on the Web, by proposing search and classification related services. The usefulness of a Web portal depends directly on the quality of service it offers. This paper presents the SIMAT architecture which exploits a backbone of machines running a secure mobile agent platform to improve quality of service. The agents are programmed to reduce the *latency* of information retrieval and to support the personalization and security of information search requests.

Keywords

Web portal, mobile agent, security.

1. INTRODUCTION

A Web portal is a site that acts as a gateway to information on the Web. It can include a number of services, such as classification utilities, discussion forums, contact points, etc. An *infomediary* is a portal aimed at a specific community. Customers are attracted to infomediarities because of the specialization of its information and services. The customers are qualified and the information is highly relevant to their professional activities. The advantage of the infomediary model for users is that the critical mass of information found there does not exist elsewhere on the Web in a packaged form. An example of an infomediary is OTLAND which is located at www.otland.com and whose specialty is middleware technologies, e.g., XML, Linux, agents, Java and OMG activities.

*Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SAC 2003, Melbourne, Florida, USA Copyright 2003 ACM 1.58113.624.2/03/03 ...\$5.00.

As it stands today, the infomediary concept has shortcomings. First, to be economically viable, it clearly needs to attract as many customers as possible. Yet attracting more customers can mean including more wide-ranging information. This in turn can compromise the portal's value as a specialized information provider. Another shortcoming is that the model essentially runs over the Web, where information must be manually collected and copied to the portal. This implies that latency can exist from the moment that the information exists on the Web to the moment that the portal customer gains access to the information.

One of the goals of infomediary services is personalization: adapting a service to the requirements of each user. A current shortcoming with personalization is that a customer must release personal information to the portal, e.g., author preferences at a book store portal, or a credit card number. Sometimes, customers will not want to leave such information at the portal, even if they still want personalization.

This paper introduces the SIMAT system. The goal of SIMAT is to address some of the shortcomings of the infomediary model. In particular, we aim to permit customers to obtain up-to-date and personalized information in a minimum of delay without overloading the portal with information. The system also allows customers to control the release of information to the portal. Implementing these properties involves using a secure mobile agent platform called Lana [5].

The implementation of the SIMAT services uses a backbone of machines that are connected to the Internet, where each machine runs the Lana agent platform. We name this set of machines the *Agent Backbone*. An agent is an autonomous program that lives on the network and which can be delegated tasks on behalf of users. In SIMAT, the role of an agent on the backbone is to encapsulate a search request, to participate in the active search for information for customers and portal managers, and to inform them when the information becomes available. An agent can encapsulate a personalized request on behalf of a customer by storing ontology information for the user. Further, this personalized information is stored within the agent, and the security properties of the platform prevent this information from being released unless the agent has been explicitly programmed to release it.

This paper is organized as follows. Section 2 introduces the idea of information portals, and highlights the weak points

of this technology. Section 3 introduces the agent backbone and explains how we address the shortcomings mentioned. Section 4 presents the design of the value-added services in detail. Section 5 presents related work and Section 6 concludes.

2. INFORMATION PORTALS AND INFOMEDIARIES

An information portal is a Web service that acts a gateway to the Web. It usually contains a set of services to help users locate information. An *infomediary* is an information portal that specializes in some domain. In contrast to traditional portals, its clients are specialized in a domain and use the portal because it offers information and services that they could only collect after spending much time browsing the Web on their own.

2.1 The Basic Portal Model

There are several actors involved in the information portal or infomediary model:

- The *customers* are the portal end-users. They represent individuals who use the portal to find information. They can be large in number and represent a variety of interests.
- The *mediators* are the portal managers. In the current portal model, they are responsible for placing and updating information in the portal.
- The information *providers* are those companies and individuals whose information is placed in the portal.

One way to improve the quality of portal service is through *personalization*. Personalization is the individualization of a product or service into a unique specialized solution for a customer. The basis for personalization is the management of user profiles. A profile is a persistent collection of (relevant) information about a certain user or topic. The content of the profile depends on the potential use of it; a travel agency will store different information about its clients compared to a bookstore. Building profiles requires the collection and storage of the necessary data. On the Web, there are two categories of data collection: reactive and non-reactive. Reactive data collection is through the explicit input of personal information by the user. Non-reactive collection involves data generated by using the Web, e.g. server log-files, history etc.

The reason for information specialization on the infomediary is to attract a sophisticated and qualified community. The service offered by infomediaries should be sufficiently valuable for them to charge a fee to customers and providers. Nevertheless, despite the attractiveness of the infomediary idea, it has not yet shown itself to be valuable to the point of being profitable. An improved quality of service is perhaps necessary before a final judgment can be made on profitability. This improvement can come by addressing several shortcomings in the infomediary model.

2.2 Improving the Portal Service

The first shortcoming is that information access follows a pull model: a customer must connect to the portal to retrieve information, and a mediator must actively search the Web for information to integrate on the portal. This leads to a latency that degrades the value of the service. A push model is more appropriate in this case because the onus is on the services to work for the customer, rather than having the customer work the services. The second shortcoming is that it is too hard to concisely furnish the information that each customer requires. Personalization tools work to achieve this so that on the one hand, a customer's request is satisfied, and on the other, that the portal is not overloaded with information of little interest to most customers. However, personalization requires that customer information be placed at the portal site and this is offsetting to many customers.

Customer services

The first shortcoming of the portal model is that **customers need to be pro-active**: if a customer requires information, then he must take the time to go to the portal to get that information. While a single visit to the portal is of course necessary, multiple visits can be time-consuming and annoying.

Consider the example of a customer who queries the portal for a presentation about System S - a hypothetical middleware technology. If the presentation is available at the portal when he visits, then obviously the customer is satisfied. If the presentation is not available, then there can be three explanations. First, a presentation about System S simply does not exist, anywhere in the world. Second, the presentation exists on some Web site but has not been incorporated into the portal by the mediator. Third, the presentation does not yet exist, but soon will. In the latter two cases, the portal should automatically inform the customer as soon as the presentation becomes available. In this way, the customer only needs to visit the portal once to obtain the information or to signal his desire to obtain it.

Mediator Services

A second shortcoming of the portal model is that **mediators need to be pro-active**. They receive information from several providers to integrate into the portal, e.g., presentations, press releases, etc. Sometimes, the mediators must search the Web for the information themselves. This results in latency from the time that information is created to the time it appears in the portal. Further, the search for information can be time-consuming, and even futile since the information they seek might not yet be available.

Further, mediators must handle updates which can be a repetitive task. For instance, consider a portal that contains information about a conference which customers have expressed interest in. Each time that information about the conference changes, e.g., a new addition to the program, or a change of venue, the mediator must inform all interested customers about this change.

One solution to handling the shortcomings related to mediator services is to permit information providers to update the portal. However, this is a task that many information

providers do not want to do. Further, it requires giving and managing passwords for the portal that providers use when updating their content. For many portal mediators, this solution poses a big security risk. Another solution for reducing the delay in information acquisition is to place hypertext links in the portal pages to the providers' content. While this helps to a certain degree, it means that the mediator has less control over the presentation of the content. This is counter-productive since the portal's value partly comes from the control of content presentation that the mediator exercises.

Both of the shortcomings mentioned lead to a degree of *latency* in the information supply chain. It is clearly a value-added service for the portal if this latency can be eliminated. Fundamentally, the portal has to keep locally a continually up-to-date synthesis of information that exists elsewhere on the Web. Since this can only be achieved manually, latency arises.

Secure Personalization

A third important shortcoming of the portal model is that **increasing the customer base requires an increase in the amount of content**. While increasing the content is of course technically feasible, it makes it harder for the customer to locate the information he seeks and for the mediator to manage all of it. Customers use infomediaries because they are portals that specialize in some domain. Broadening the scope of the domain might dissuade existing customers who fear that the portal is becoming over-loaded with information of little interest to them.

The key solution to this problem is to permit the personalization of requests. This entails making the service "different" for each client so that each has the impression that he is using a service specifically built for him. Achieving this requires that a profile be built for each user who registers for the portal. In the context of the OTLAND portal, this profile specifies the type of technology that interests the user, e.g., XML or Java, Linux or Windows, whether he is oriented towards development or management, etc.

To be really effective, personalization requires that the portal remember customer preferences or personal information (such as a credit card number). Given the risk of break-ins on servers and the possibility of misuse or theft of information, e.g. for SPAM or credit card number misuse, it is not always acceptable to clients that their information be logged at the server.

3. THE AGENT BACKBONE

The aim of the SIMAT system is to address the problems outlined concerning infomediaries using secure agent technology. This section explains that technology, and also the Lana agent platform that we are using.

3.1 Agents

The notion of agent is very common in the human world of business and commerce. It refers to an individual who is a specialist in some domain, and to whom people delegate tasks. A common example is that of a travel agent or estate agent. In the case of a travel agent, people delegate to him

the task of reserving a trip in exchange for a fee. The advantage of this model is twofold. First, the customer can be confident that the task will be done, since usually the agent is someone with the necessary competence for carrying out the task. Second, the customer is himself free to do other things - or to do nothing! - while the agent is handling his task.

There has been much effort over the past decade to apply the agent paradigm to the context of the Internet [10]. In this case, an agent is an autonomous software program that executes a task on behalf of a customer. A simple example of an agent is a web crawler, sometimes called a bot. This is a program that periodically searches sites for its collection of Web pages and which then builds an index of the pages that it finds. This is an example of the agent model since the bot searches the Web on behalf of customers.

A *mobile agent* is an agent program whose execution can be halted and its execution state packaged into an envelope. Execution of the agent only continues when the envelope is opened. The envelope can be seen as a snapshot of the agent's execution at the point in time that the envelope is created. The envelope itself is simply a blob of data, like a file, that can be exchanged between programs, perhaps over the network. When the envelope is opened on a different site to that on which it was created, the agent continues its execution on this new site. This creates the effect of a program (agent) having moved from one site to another. There are several reasons why agent mobility is a useful feature for applications.

Reducing network latency A dialogue between a customer and a server can involve the exchange of a number of messages over the network that connects them. Network connections can (still) be expensive, unreliable, and constitute a performance bottleneck for the application. In the mobile agent approach, a client encapsulates his logic within a mobile agent and sends this agent to execute on the server machine, or on a machine close to the server. In this way, the interactions between the client and server happen independently of the network, and do not suffer from problems with the network. The network is only necessary for sending the agent and for communicating the result back to the client. The approach just outlined is particularly useful when the client wishes that his machine remain **disconnected from the network** as often as possible. This is the case when the machine is a mobile telephone since the connection is an expensive and slow GSM link.

Deploying new services The fundamental feature of mobile agents is that programs can be dynamically deployed onto one machine from another using the network. This is useful for deploying services on machines that are not easily accessible - such as machines in a different location to where the owner is. This feature has been exploited for active networks [12] where network protocols and services are deployed on routers. The agent approach allows deployment to occur automatically, which is a significant improvement on manual remote shell tools.

Exploiting extra computing power A more recently discovered potential of the mobile agent approach is to ex-

exploit computing power that exists elsewhere in the network. The client simply deploys his computation on a machine that is currently available for this. This approach of exploiting extra computing power on the Internet is becoming increasingly common. The bottom line is that there are enough computers connected to the Internet that do nothing, enough of the time, for their power to be harnessed for other tasks. This has been seen in the peer-to-peer community [7] and the Grid [9]. SETI for instance is a peer-to-peer system that makes use of available PCs to compute functions that analyze radio signals from outer space¹. Entropia is another peer-to-peer system used for calculating the largest known prime number [6]. Though these platforms do not yet use particular agent systems, they illustrate the trend and benefits of deploying programs on foreign machines in order to get tasks done.

3.2 The Lana Mobile Agent Platform

Mobile agent computing does have the drawback that a new security risk is opened up. This comes from the fact that a user is able to deploy code on another user's machine. This code could contain a Trojan Horse or a virus. Currently, many agent platforms have serious problems in curtailing the effects of a malicious mobile agent [13].

This security concern was one of the reasons for designing and building the Lana platform [5]. Lana offers several security properties. One of these is that a Lana agent that executes on a machine cannot gain access to resources on that machine or to other agents' data, unless this has been explicitly permitted by the security policy of that machine.

This security property has important implications. One is that a client agent that executes on a server machine cannot learn anything about the data in local files or inside of other client agents unless the local administrator and the other clients agree to allow that agent to gain access. Further, even the server software running on the machine cannot gain any access to information inside of the client agent.

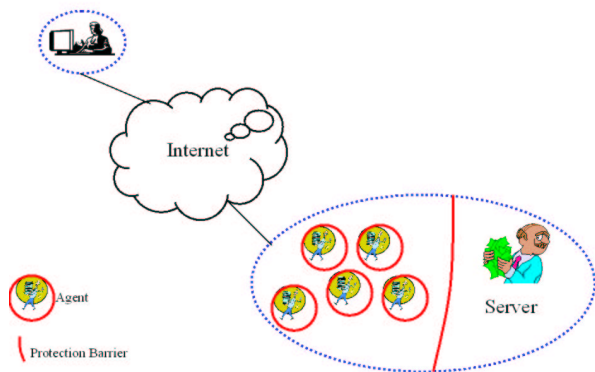


Figure 1: Agents are securely confined on a machine.

Figure 1 illustrates this property. The figure shows a client machine separated by the Internet from a server. The server

¹setiathome.ssl.berkeley.com

machine hosts a number of client agents. A thick line that denotes a protection barrier surrounds each agent. This line also separates the server software from the agents. This means that any information stored within an agent cannot be transferred outside of its protection barrier unless that agent is explicitly programmed to do this. Similarly, no information inside the server software or host system can be illicitly copied to an agent. Thus, even if the agent contains sensitive user information, like a credit card number, the server cannot see this unless the agent is programmed to release it.

The only way for this security guarantee to be violated, is if an attacker has tampered with the host operating system and Java virtual machine². However, loading a bogus version of an OS or Java environment is much more difficult than simply loading an agent containing a virus. Whereas the latter is enough to launch security attacks in other agent platforms, Lana prevents such viruses from causing damage.

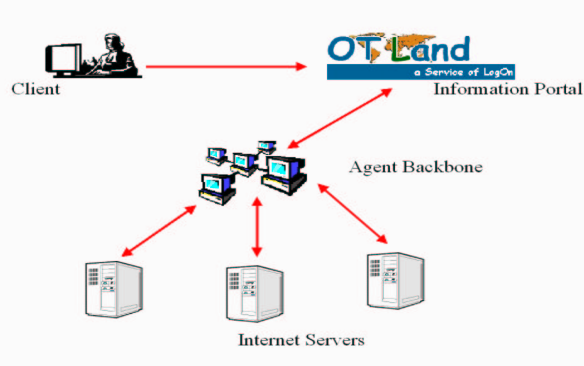


Figure 2: The structure of the agent backbone.

In SIMAT, a set of machines has the responsibility of running agents for the service, as is illustrated in Figure 2. These agents handle information search requests for the portal customers, providers and managers. A customer does not have to be connected to the portal while his agent is running.

The Lana system itself is an agent platform designed to run on standard user PCs connected to the Internet as well as on hand-held devices. Lana is modeled on Java because of its object-orientation, strong typing, security features, as well as its bytecode interpretation (and runtime compilation) approach that allows programs to run on heterogeneous platforms.

Like in Java, a Lana platform is structured from a basic hierarchy of classes. The core Lana classes are illustrated in Figure 3, where the arrows denote subclass relations.

LObject is the super-class of all classes in Lana. This class only contains a few basic methods, such as for printing its value to standard output and for calculating a hashcode. A Lana environment runs a set of concurrent *programs*. Programs may be mutually mistrusting so the Lana kernel pre-

²Most agent platforms run in a Java environment, and thus rely on the correct and secure functioning of this environment.

vents a program from gaining direct access to another program's objects. They can communicate using method calls through a security policy controls each of these calls. The security guarantees offered by Lana are crucial to be able to implement the SIMAT backbone.

In Lana, programs are mobile, meaning that they can be moved between machines. SIMAT agents are in fact implemented using Lana Program objects. Mobility is achieved by stopping a program's execution thread and then copying the program object - and the transitive closure of all referenced objects - to the new machine, where the thread is restarted. Migration is made possible by the fact that, primarily for security reasons, objects are not directly shared between programs.

Program mobility is the key to service development and deployment using Lana. To deploy a new service to a Lana platform, one simply needs to code that service as a Lana agent (program), and have the program migrated to the host machine. This approach makes service deployment and modification very flexible, which is one reason for using an agent platform in SIMAT. Typical password protection is enough to prevent unknown users from deploying rogue services to a backbone machine. Each user can select the backbone machine on which he deploys his service.

Lana programs can communicate using asynchronous method calls. These calls are secure in that the calling program is given a fresh Key security object when the call is issued, and only this key can be used to interpret the method reply. This mechanism prevents malicious programs from intercepting or tampering with messages destined for others.

A second communication mechanism in Lana is the message board. Each Lana platform contains one message board. This is an associative object container in which programs can place and retrieve object copies. For security reasons, each object placed in the board is locked with a key; only a program possessing the key used to lock an object in the board may read that object. A second function of the message board is *resource discovery*: when a machine joins a network, its programs typically search the message boards of other devices in the network to see what resources are available. To support this, each machine publishes its services in its local message board using publicly known keys. Thus, when a machine joins an agent backbone - when it runs the Lana platform - it can query its neighborhood to locate other machines running Lana. The set of these machines constitute the Lana-based agent backbone. The message board is also used by agents that arrive on a platform

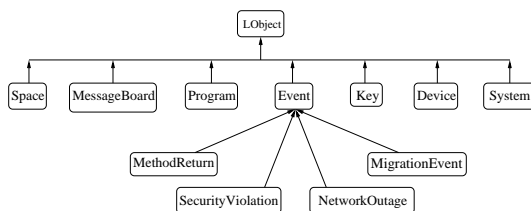


Figure 3: The hierarchy of principal Lana classes.

to discover the local resources.

The Event class represents a message that is asynchronously sent to programs. Pre-defined event types include those that indicate that a method has returned (MethodReturn), or that the method call has failed due to the invoked program having moved (MigrationEvent) or no access right having been granted (SecurityViolation).

4. THE SIMAT SERVICES

4.1 The Otland Portal

Created in September 2000, the OTLAND portal located at www.otland.com specializes in middleware, e.g., XML, Linux, agents, Java and OMG activities. The SIMAT system currently implements a search service for OTLAND. In this service, developers working on middleware projects can register information about their project. Other project managers can then gain access to this information. The OTLAND portal mediators have the following requirements for this and future services.

The deployment of new services should be fast and seamless By “fast” we mean that the time needed to obtain a running service should be short. This is because the number of users is increasing rapidly and one cannot foresee the types of service that one might need to add in the future. It should be easy to remove services that are not attracting clients. By “seamless” we mean that a service can be added without having to bring the portal down. The Otland portal currently gets nearly 3.000 hits a month, and any down-time is clearly undesirable as all customers come through the Web site. This requirement is supported in Lana by having services deployed on a backbone of machines, independent of the main Web site. A service is encapsulated in a set of agents whose life span is independent of other agent services and which run on machines that are independent of that used for the Web server.

Service development must be out-sourced The mediator company for OTLAND is an IT company, but does not do development. Whenever a new service needs to be developed, this task will be out-sourced to an independent developer. Integration of this new service with the portal is facilitated using the backbone model.

Requests contain sensitive client information In the current SIMAT service, an agent contains a user's e-mail address or other personalized information. Some of this information, such as the e-mail address, is never disclosed to the server. The security guarantees of the Lana platform assure users that their e-mail address will never be disclosed to a source that could exploit the address for sending SPAM. The rules governing the release of information are programmed into the agent.

4.2 SIMAT Services

The SIMAT system uses a backbone of machines to serve the information portal. There are currently three machines on the backbone, two in Geneva and one in Frankfurt University. Each machine runs the Lana platform, and thus all agents are Lana agents.

Currently, a client will choose a backbone machine based

on the trust he possesses in the provider of that machine. He may also choose based on the quality of the connection between the backbone machine and the Web server. He may even choose to make his own machine part of the backbone if he has little trust in the current set of backbone machine providers. This flexibility is a key advantage of SIMAT.

A portal customer uses the service via a browser front-end that connects to the information portal's Web page. He specifies his request using a simple applet. This request data is then fed to the agent backbone where an agent is created that encapsulates and executes the request. At this point, the client can disconnect himself from the portal. The agent is running in the background and will send the request result to the client via e-mail as soon as it is available.

The information provider has a simple GUI front-end to the system. A snapshot of this is illustrated in Figure 4. The information he enters is tailored to the SIMAT system. In particular, it is information stored on the backbone for the agents to search. As such, the information might be more than what is available on the provider's public Web page, or even more specialized than the information that the provider has made available to the portal's main site. This is a policy decision of the provider. This explains how the information available in SIMAT can be tailored to individual interest groups without overloading the main Web service with information of little interest to most customers.

The primary agent scenario is the agent search request. In this case, a customer logs onto the portal to specify his request. This information is fed to a daemon agent on the backbone which creates a new *request agent* that encapsulates the request and which is completely responsible for it. The architecture is illustrated in Figure 5.

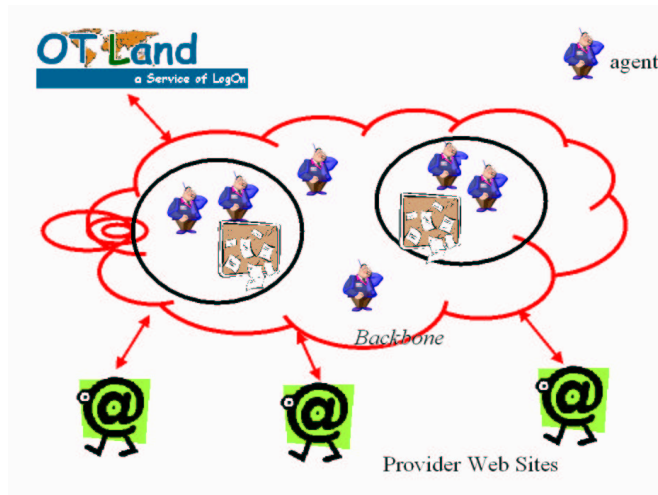


Figure 5: The structure of the agent services.

The set of backbone machines is used as a distributed cache. For the OTLAND services, the Lana message board of a backbone machine is currently used to store project profile information. When a request agent arrives on a backbone, it queries the project profiles for a match. If a match is found, then the project information is extracted and composed into a mail that is sent to the customer. If a match is not found,

then the agent goes to sleep until new project information is added to the backbone cache.

When a provider enters a project profile, or update to an existing profile, the information is immediately inserted into the cache by a *provider agent*. The addition of provider information wakes up all request agents. The request agents look to see if the new information is relevant to the information they seek, and if this is the case, they send new e-mails to the customers indicating this new information.

Each request agent has a validity date that indicates when the agent should expire - the point at which the customer is no longer interested in the information. While a request agent is active, the customer that owns the agent will receive the requested information, and any subsequent updates to that information, until the request agent expires. Thus, the customer is able to follow the evolution of the requested information, which is a value-added service of the SIMAT system. This distinguishes it from typical search tools since each customer is informed of modifications to the information that he expressed interest in.

There are also `http` agents on the backbone whose role is to interact with the main portal site as well as other web sites using the `http` protocol. These agents can be used to have public information from the main portal site appended to request data. `Http` agents can also be programmed to talk with well-known sites such as `google` or a major middleware provider, so that information from these sites can be transferred to the user.

4.3 Profiles

Personalization is based on user profiles. A profile is a persistent collection of relevant information about a certain user or topic. The content of the profile depends on its potential use; a travel agency will store different information about its clients than an infomediary.

The role of profiles in SIMAT is to efficiently match requests to technical items on behalf of customers. This means that customer profiles have an influence on the result sets and on the way that results are presented to users. The profiles are currently stored on the backbone of SIMAT, and describe customers, requests and technical items.

- A customer profile is created when the customer registers with the SIMAT service. The customer's profile contains his technology preferences, his occupation (e.g., programmer, manager, etc.) and the main tools that he uses.
- A technical item is something which is often searched for via the portal. The reason for associating profiles with technical items is that each item has a specific meaning in the context of the portal and it is vital to know this meaning to locate relevant information on the customer's behalf. In OTLAND for instance, the technical item named "conference" clearly relates to a conference organized on Java, XML or OMG. Thus, a search for "conferences" via the portal can thus disregard information about conferences in other domains. Further, a user with the profile "Java programmer"

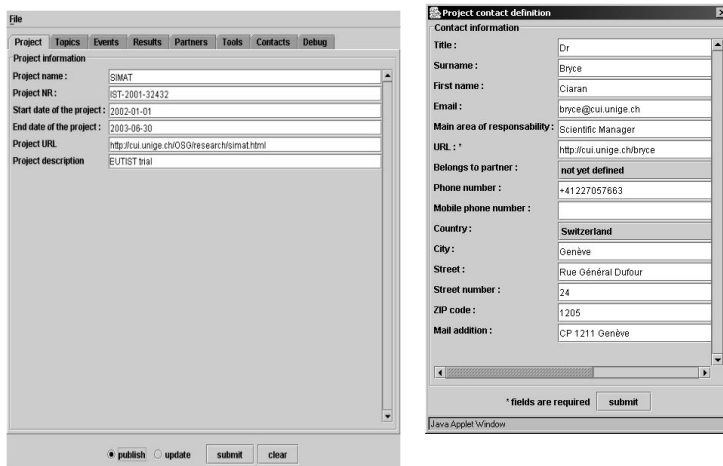


Figure 4: The provider interface for the information portal services.

may not need to be given information on Linux conferences, if Linux does not form part of his profile.

- Finally, requests are also stored in a profile. They are entered by the user filling out the request form and stay in the system until the request expires.

At the moment our system works with a single name domain, that is, keywords have a unique semantics. As a starting point, we provide simple pattern matching to match requests. This is suitable for our current closed world application. We are nevertheless beginning to extend this, as described in the next section.

4.4 Improvements using RDF

In the future, producer information will be stored in XML format, using the Resource Description Framework [11]. RDF is an information description format that will be used in the Semantic Web [3].

One of the main benefits of RDF compared with XML is its default mapping to the semantic tree. If pure XML was used, then the agents would encounter two situations: either they know the XML schema or DTD associated with a data item, or they do not. The latter situation is undesirable. However, in the context of an environment as heterogeneous as the Web, one cannot enforce the use of a particular DTD. RDF gives a more flexible approach. Even without knowing the complete set of the used vocabulary the agents could generate the semantic tree, find parts they understand and use this information to support the user. Additionally RDF is designed in an object oriented way to represent relations of ontology. This enables the agents to infer or derive information from the knowledge they have. Several systems now exist that illustrate the power of RDF, e.g., the “Closed world machine” [2] or RDFSuite [1] with its specialized query language for RDF called RQL.

We have already generated RDF Schemas (vocabularies) to describe the profiles needed for SIMAT. Elements of profiles can be uniquely distinguished since the name space of the vocabulary employs URIs. Thus, the problem of homonyms are avoided. We can thus ensure that the result returned by the agent to the user really fits his needs.

Since our system knows the semantics of the vocabularies, further improvements are attainable. For instance, assume we have a user who is interested in 'topics:XML' and 'topics:RDF', and this is stated in his customer profile³. This user now issues a request for 'simat:Conference'. Currently, the result will contain all conferences with 'simat:Conference' in their profile, as well as 'topics:XML' and 'topics:RDF'. In addition to this, in SIMAT we distinguish between conferences and workshops. Normally, events with 'simat:Workshop' in their profile do not match conferences since not all workshops are affiliated with some conference. In cases where workshops are affiliated, we can define the relation *affiliated*, which maps workshops to conferences. This is sufficient in RDF to enable an inference engine to have information about affiliated workshops included in the query results.

5. RELATED WORK

The notion of agent - a program that runs independently of the client and that does work for him - is of course not new to the Web. A web crawler or spider is one example. Portals now exist that allow users to create agents that inform the users if some other Web site is changed, or to surf the Web for a cheap airfare. In the Web community, these agents are often called *bots* (short for “robots”) [4].

In comparison to SIMAT, these bots are fixed and do not allow the dynamic deployment of new services. This dynamics is an important goal of SIMAT since the services that the portal offers can vary as the requirements of the portal

³We are using the common abbreviated form for resource URIs.

community evolve. Even going beyond SIMAT, the ability to define personalized agents is important to support the complete functionality of the Semantic Web. Further, the SIMAT service allows requests to be followed up on, so that any change to the information is propagated to the customer.

A system that has similar goals to ours is Opelix [8]. This system uses agents for refined searches. Since the information sought by a user is often on several sites, agents are programmed to co-ordinate the search results from individual sites. This idea is useful and we could adopt the technique in SIMAT in the near future. The aim of SIMAT is slightly different. Our goal is to develop an architecture that can support personalization of requests and to reduce information acquisition latency. To obtain this, we employ a backbone of machines. This allows us to integrate services that do not require changes to the existing Web services. We prioritize service deployment and availability.

The products Informant and Monitor Information System (MIS) by the Swedish company GetUpdated⁴ are in some points comparable with SIMAT. The Informant monitors a number of user specified web sites and informs users about updates. In SIMAT, the infomediary specifies the web sites for the users and adds or removes them if needed.

Autonomy (www.autonomy.com) offers similar services to SIMAT. These allow to build up profiles and offer personalized services. The main focus of Autonomy is on intelligent filtering algorithms. Though such algorithms are important in a context like SIMAT, this system does not provide an infrastructure for security or for dynamic service provision.

6. CONCLUSIONS AND FUTURE WORK

This paper has presented the design of agent-based services that use the Lana agent platform. The services aim to improve the quality of an infomediary information portal by reducing the latency of customer and mediator information acquisition and by permitting customers to introduce personalized requests with security. The services are built on the principle that information services can be cheaply, quickly and securely deployed using a lightweight agent platform with in-built security features.

Future work includes the deployment of other services. An example is a statistics service, where agents monitor the requests being made. This service is useful to portal managers so that they can understand the evolution of the portal community's interests. Another service being tested is the possibility for agents to search the caches of the machines of the backbone for the information they require, in a peer-to-peer fashion. Thus, even if a Web site is unavailable, an agent can search the caches of the backbone machines for copies of the requested information.

The service described in this paper is being designed for the OTLAND portal, located at www.otland.com. The service is being tested by a cluster of users from European Union projects in the area of agents and middleware.

Acknowledgments The SIMAT project is co-sponsored by

⁴www.getUpdated.com.

the European Union as a trial project as part of the EUTIST cluster on *Agent Middleware Infrastructure Technologies* under grant number IST-2001-32432 and by the Swiss Department of Scientific Research (OFES) under grant number 01.0211-1.

7. REFERENCES

- [1] S. Alexaki, N. Athanasis, V. Christophides, G. Karvounarakis, A. Maganaraki, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: High-level Scalable Tools for the Semantic Web. Technical report, ERCIM News No. 51, October 2002 (Special Theme: Semantic Web), Oct. 1995.
- [2] T. Berners-Lee. Cwm - Closed World Machine. <http://www.w3.org/2000/10/swap/doc/cwm.html>.
- [3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [4] G. Beuster, B. Thomas, and C. Wolff. MIA - A Ubiquitous Multi-Agent Web Information System. In *Proceedings of International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, Dec. 2000.
- [5] C. Bryce, C. Razafimahefa, and M. Pawlak. Lana: An Approach to Programming Autonomous Systems. *ECOOP European Conference on Object Oriented Programming*, LNCS, 2002.
- [6] A. Chien. Parallel Programming Challenges for Internet-scale Computing (Entropia). *ACM SIGPLAN Notices*, 36(7):72–82, 2001.
- [7] D. Clark. Industry Trends: Face-to-Face with Peer-to-Peer Networking. *Computer*, 34(1):18–21, Jan. 2001.
- [8] E. D. N. et al. Using agents in performing multi-site queries. In M. Klush and F. Zambonelli, editors, *Cooperative Information Agents V, Proceedings of the 5th International Workshop CIA 2001*, number 2182 in LNAI, pages 100–105. Springer-Verlag: Heidelberg, Germany, Sept. 2001.
- [9] G. Fox and D. Gannon. Grid computing: Computational grids. *Computing in Science and Engineering*, 3(4):74–81, July/Aug. 2001.
- [10] N. Jennings, K. Sycara, and M. Woolbridge. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-agent Systems*, 1(1):7–38, Mar. 1998.
- [11] J. Mason. XML and RDF: the promise and the reality of new Web architectures. *Computer Networks and ISDN Systems*, 30(1–7):763–765, Apr. 1998.
- [12] P. Tullmann, M. Hibler, and J. Lepreau. Janos: A Java-Oriented OS for Active Network Nodes. *IEEE Selected Areas of Communication*, 2001.
- [13] J. Vitek and C. Bryce. The JavaSeal Mobile Agent Platform. *Autonomous Agents and Multi-agent Systems*, 3(2):7–38, Mar. 2001.