An Active Ontology-based Blackboard Architecture for Web Service Interoperability

George Lepouras[1], Costas Vassilakis[1], Anya Sotiropoulou[1], Dimitrios Theotokis[1], Akrivi Katifori[2]

[1]Department of Computer Science and Technology, University of Peloponnese
{gl, costas, anya, dtheo}@uop.gr
[2]Department of Informatics and Telecommunications, University of Athens
vivi@mm.di.uoa.gr

Abstract—**Web services are functional, independent components that can be called over the web to perform a task. Web services are provided by organizations to enable others to perform tasks the organization offers online. However, with an ever increasing number of web services, finding the web service that performs a certain task is not always easy. Furthermore, adopting an end-user point of view what is needed is the actual result and not the service per se. It is often the case that more than one service have to be combined to produce the anticipated outcome, e.g. in the case of life-events. To this end, we propose an active, ontology-based blackboard architecture that aims at tackling the problems inherent in dynamic synthesis of composite web services and at facilitating user interaction with complex e-government transactions.**

Keywords: Web service synthesis, active blackboard architecture; ontology-based architecture

1. INTRODUCTION

In the early days of the Web the new medium was mainly used for disseminating information to users. Soon it was realized that the new medium could be used for selling products and services to users. Starting with books and cds users can nowadays find and buy almost anything. The private sector was the first to use the Web for reaching new potential customers and the public sector followed.

Electronic government employs the Web as a medium that can help citizens perform actions online without the need for their physical presence. Electronic services can be used for this purpose, with citizens accessing them from the ease of their home. Although at first governments aimed at implementing and getting online as many services as possible, it was soon realized that this would not be enough if electronic services were to be used as an effective means for fighting bureaucracy. The reason for this was twofold:

- Citizens were often unaware of the availability of an electronic service reverting to the traditional methods.
- More than one service had to be invoked in order to complete a citizen's task, invalidating the advantages of performing actions online.

The first problem could be solved by creating central locations (portals) for publicizing available electronic services. The second problem can ideally be solved by creating a new, 'meta'-service that will invoke all necessary services on behalf of the user. In such a case the intermediate services that are being invoked have to have a known method of interaction. This is the notion of a web service.

Although a web service is an electronic service the inverse is not necessarily true. A Web service denotes a functional component that has a published interface for invoking it in a standardized manner. Web services do not have a graphical user interface for users to interact with. A web service uses a standardized interface to offer data and processes through the web. If a user needs to access a web service a developer can write a web page or a program that calls that service and provides the front end.

A web service can be called from different applications, each written in different programming languages and residing in different operating environments. Basic communication is established with protocols based on XML as explained in the next section.

In order to be able to find a particular web service, developers have to know its location. Although browsing tools exist, finding the appropriate web service is not easy, especially when the same service is provided by more than one administration.

To this end we propose an architecture that enables through the use of an active, ontology-based blackboard the registration and interoperability between web services; the architecture also provides facilities for the users to locate and invoke these web services either individually or through their dynamic composition.

The rest of the paper is structured as follows: the next section describes web service composition issues, the third section offers an overview of the proposed approach while the fourth section elaborates on technical issues. The final two sections illustrate the architecture with a usage scenario and conclude with general considerations and future research topics.

2. WEB SERVICE SYNTHESIS

In order to assemble a new service from other web services the developer has to be able to find the appropriate web services and to know how to invoke them. To this end the UDDI [1] initiative was formed. UDDI is an open industry initiative (sponsored by OASIS) enabling web service providers to find each other and define how they interact over the Internet. UDDI stands for Universal

Description, Discovery, and Integration. UDDI describes a registry where service providers can register themselves. UDDI is XML-based, platform-independent standard for web services and it is designed to enable SOAP messages to request information from it.

SOAP [1] is a XML-based protocol originally designed by IBM and Microsoft, which allows the exchange of messages between software components. SOAP is an acronym of Simple Object Access Protocol, one of the basic protocols for web services. SOAP is extensible and although is usually executed over HTTP, it can be run on top of all the Internet Protocols.

Apart from being able to respond to SOAP messages, UDDI provides access to WSDL documents. WSDL [1] stands for Web Services Description Language, an XML format describing Web services. WSDL is used to describe the binding and the message format a web service can understand.

When a program needs to interact with a Web Service it can first communicate with UDDI to find the web service and the proper way to interact with it. With the help of these protocols and standards web services can be called by other programs synthesizing a new service for end-users.

Although web services can be synthesized together in any order and combination to produce a new service this is not done automatically. In eGov [2], an IST-funded EU project, simple services can be combined to form new composite electronic services. This task is carried out by the developer who has to find the required web services, get the set of all input necessary for the execution of the services, arrange for the flow of execution and of alternative execution patterns depending on the output of the services, the linking between the output and input of services to create the composite service. Similar approaches are taken by technological frameworks, such as [3].

In [4] the strictness in the specification of the execution path is relaxed allowing the predetermined execution path to be altered in run time. This work additionally introduces consistency rules, which are able to verify that changes that have occurred either in the composite service schema or in some constituent service have rendered the composite service to be invalid; however still human intervention is required to remedy these cases.

Sirin et al [5] describe a prototype which employs a semi-automatic process to guide developers in the composition of a new service. The prototype allows the developer to choose from possible matches at each step of the composition process, taking advantage of the semantic information available.

All the approaches previously described do not tackle efficiently the problem of dynamically composing a web service. To be able to dynamically synthesize a composite service from existing web services it is of the essence to facilitate the process of finding a web service among many possible alternatives. Klein and Bernstein [6] propose the use of a process taxonomy approach, in contrast to 'frame-based' approaches for 'matchmaking' between tasks and on-line services. According to this, service retrieval can be improved without sacrificing precision and increasing the computational complexity. However, this approach helps retrieve potentially 'right' web services it still cannot be employed in the automatic composition of services from the web services retrieved.

In this paper, we propose an ontology-based approach employing an active blackboard which can facilitate the automatic composition of services, based on the input and output parameters of the web services. Ontologies offer on the one hand high-level semantics, allowing humans to easily search and manipulate the modeled information, and on the other hand formalism and standardization, which facilitate mechanical processing. In this respect, ontologies can serve well all tasks related to automatic synthesis of composite services.

3. OVERVIEW OF THE ACTIVE BLACKBOARD ARCHITECTURE

The notion of a blackboard [7] [8] is similar to that of a real world blackboard. On a blackboard everyone can write something and other people will be able to access that information. Similarly on an electronic service blackboard service providers can register their services. Once a service is registered it can be made available to service consumers. Data passed to services as input parameters or results produced by the services are also written on the blackboard for future reference and use. In contrast to a real blackboard its electronic counterpart can be active. In a real-world blackboard when a user writes something other users have to look at it (possibly by polling) to view that new information is available. In an electronic blackboard, users can be automatically notified of new information depending on their preferences. Therefore, by registering a web service with the blackboard other users can be notified of the existence of a new service; modifications to the data may also trigger activities on behalf of the blackboard, e.g. the invocation of some service. Consequently, a web service can be invoked either directly or indirectly. A service is invoked directly if a consumer asks explicitly for it; indirect execution occurs when a service consumer requests service A and this needs an input produced by service B, or when the effects of the execution of some service A necessitate the execution of service B (e.g. the *announcement of move* service triggers the incorporation of the change to each PA database).

In order to implement the concept of the active blackboard, an architecture is required that will enable the registration of web services by service providers and their invocation by users. Fundamental in this architecture is the ability to correctly categorize web services to allow the effortless retrieval of the correct service. This can be achieved by employing an ontology that will cover both services and data managed by them. Figure 1 depicts the proposed architecture of the active, ontology-based blackboard.

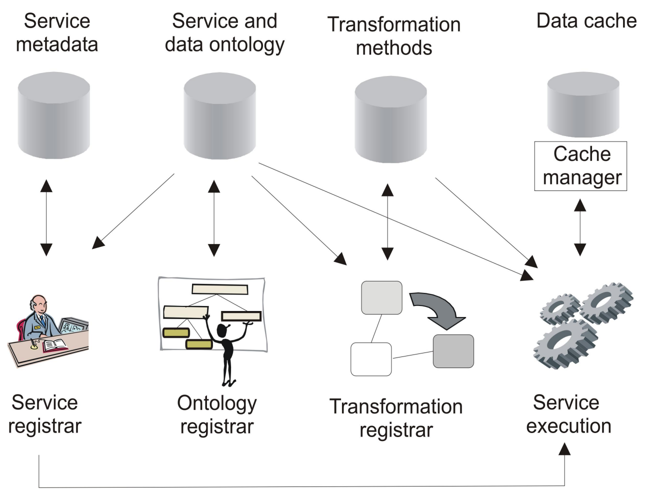

Figure 1 – The active blackboard, ontology-based architecture

As depicted in the figure, the ontology is central to this architecture, since it supports all the other modules. An ontology can be considered to be a simplified view of a domain containing the objects, concepts and other entities that exist in this domain along with the relations that connect them together.

A more mathematical definition can be the following [9]:

An ontology is a triple O = (C, R, <u>*isa*</u>) defined as follows:

1. C = {c_1, c_2,…, c_n} is a set of concepts, where each concept c_i refers to a set of real world objects (concept instances),

2. R = {r_1, r_2,…, r_m} is a set of binary typed roles between concepts.

3. *isa* is a set of inheritance relationships defined between concepts. Inheritance relationships carry subset semantics and define a partial order over concepts.

The ontology models the basic concepts related to services as viewed from a user perspective. In such a view the domain is modeled starting with the public administration, its ministries, local authorities, their departments and agencies, the services provided by them to the citizens or business, their inputs and outputs. Central concepts in this ontology are the services provided to the users, the organizations that provide services to users and the data passed to parameters or produced by them as results. This abstract model is instantiated for each web service producer. From an architectural point of view an ontology registrar is responsible for managing the service and data ontology. This component allows the categorization of services, their retrieval and provides all the information necessary for their interoperability.

When a service producer wants to provide a new service the service registrar will link the semantic representation of the service with the actual implementation of the service. The service producer will indicate which abstract service type the web service implements, data required for its execution and how the service can be called.

Once a service is registered it can be executed. The service execution module enables service consumers (end users or other web services) to request the service. The service consumer will provide the appropriate data needed for the execution of the service. The blackboard will -transparently to the end-user- invoke the service along with any other services that produce intermediate results or transformations upon them and will return the results to the user.

Transformations are generally simple functions that are applied to the data, such as database lookups and filling-in of default values, e.g. mapping social security numbers to VAT registration numbers or pre-pending a country prefix to a license plate no to turn a “national” vehicle identification to an international one. Transformation definitions are provided through the transformation registrar and are automatically used by the service execution module, when appropriate.

4. TECHNICAL CONSIDERATIONS

In this section we will provide a technical description of the design of the proposed system. The description is focused more on the user's side, the user being either the developer registering the service, maintaining the ontology, and transformations or the user being the end-user of the composite service.

A. Ontology Registration

The basis of the architecture is the ontology registrar and the underlying ontology. The ontology describes the public administration agencies, the services and data managed by them and the relationships between all these concepts. Possible relationships between concepts include *is-a*, *is-instance-of*, *has-a*, *offered-by, managed-by, consists-of, produced-by, implementation-of* and *is-related-to*. *Is-a* and *is-instance-of* are probably the simplest relationship. For example, Ministry of Finance *is-instance-of* Ministry *is-a* Public Administration. Tax Certificate *is-instance-of* Document *is-a* Data is *managed-by* Ministry of Finance. It has to be noted that before new web services are registered through the service registrar as implementations of existing ontology service instances, data managed by administrations and their instances consumed and/or produced by the respective web services have to be already modeled in the ontology.

The *offered-by* relationship links a *service instance* with an organization concept or instance, and dictates which organization or *class of organizations* has the authority to offer the specific kind of service. A service instance should not be confused with the actual implementation of the service. Since the same service can be offered by different public administrations when a new web service is registered relations of type *implementation-of* are created as described in more detail in the Service Registration subsection. Although *managed-by* and *offered-by* seem to be identical this is not the case. A public administration can be the administratively responsible for a service, but it may decide that the service can be provided by other administrations as well (e.g. the ministry of Internal affairs is administratively responsible for the “Birth Certificate” document, but permits municipalities offer the birth certificate issuance service to their citizens).

Figures 2, 3 and 4 depict part of the service and data ontology, as modeled using the KAON tool [10]. These are translated versions of the original ontology in order to be accessible to an international audience. The rectangle nodes are concepts, which are used to model classes of objects, while the rounded rectangle nodes are instances of concepts, which map to individual objects. For clarity reasons figures show only the *is-a* relationships between concepts and *is-instance-of* relationships between concepts and instances.

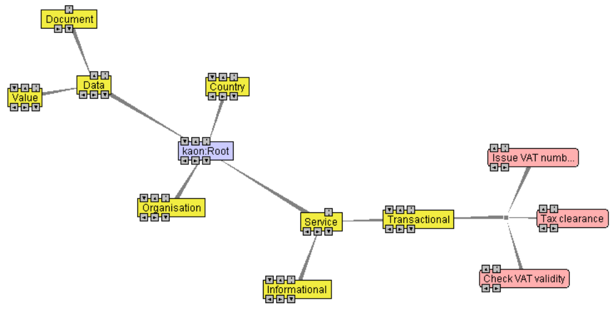

Figure 2 – Overall view of the ontology with part of the service branch extended

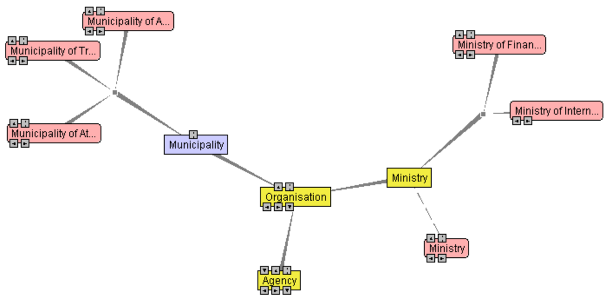

Figure 3 – Part of the organization branch

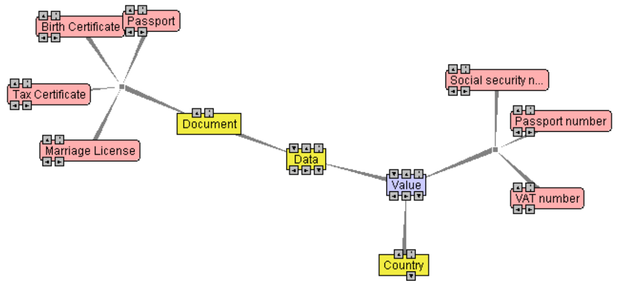

Figure 4 – Part of the data branch

Each node in the ontology has a number of *attributes* that describe various aspects of it. An instance of a *data* sub-concept has attributes dictating the valid format(s) for this element, a list of allowable values (e.g. the “Day of Week” node will list the days from Sunday to Saturday), minimum and maximum bounds (or more complex validation checks), the confidentiality level of the datum (e.g. publicly known vs. strictly personal), multilingual labels for its description etc. Attributes for organizations include descriptive data (e.g. physical location, phone and fax numbers) and *identification credentials* (user names and passwords, IP addresses, SSL keys etc) which are used by the registrar services to validate that connecting entities as acting on behalf of the specific organization. Finally, attributes for sub-concepts and instances of the “Service” concept include the confidentiality level, a *time to live* – i.e. for how long after its issuance by the respective service its result remains valid and an indication whether the service result is *reusable*, that is whether results of the specific service may be retained and used as input in a subsequent service invocation. The latter two properties can be exploited by caching mechanisms to improve execution time and reduce server load, by storing service results and reusing them instead of running the services anew.

To model the output of services the *produced-by* relationship type is used to link a single service to the data it produces. As discussed earlier this can be a value or a document.

B. Service Registration

When a developer creates a new web service the first step that has to be taken is its registration. In order to register the web service the developer has to provide the following information:

First the developer selects the public administration, which provides the service to the public, the service that is being implemented, the input and output data, the invocation method and the constraints that may govern the execution of the web service.

This process is supported by the ontology registrar, which holds not only the model of the domain but also the instantiation of the services and the necessary information for calling the web services. Input and output data can be simple *values* or *documents* necessitating the distinction between them. The system transparently to the user creates instances of the "Data" concept and relates them to the input and output data of the web service. In cases where there exist alternative data that can be given as input to the web service, a new *container concept* is instantiated as a child of the "Data" concept, and the alternative input types are linked under it. Then, the service is linked with the container concept, to denote that *any of its children* can be used as input to the service. For example, if the execution of a web service requires either the provision of the social security number or the passport number, the container concept 'Identification Data' is created, which then is linked to social security number and passport number. The web service is then linked to the 'Identification Data' container, rather than the individual instances, denoting that *any* of the data inputs can be used. Since the data used and produced by each administration are entered in the ontology before any services manipulating them are defined, the developer will select the input and output data types from the list of the existing types, securing thus the consistency of data types between different services.

Constraints for the execution of the web service in the values of the input data can also be set to limit the range of acceptable values. These constraints are set as attributes to the service implementations and can also dictate the conditions under which the service can be executed. For example, if a web service is offered by more than one public administration, each for a different user group, by setting a constraint such as Zip_code =”165*” the user group that can execute the service can be limited to those with the specified range of zip codes. Finally, the invocation method specifies how the web service can be called. The actual calling method depends on the service implementation: for services implemented as web services [1], a WSDL specification can be provided; for services delivered through RMI [11] the name of the remote object, the interface name and method should be listed etc.

C. Transformation Registration

The transformation registrar manages the *transformation method* repository, whose entries specify how certain descendants of the “Data” concept in the ontology can be transformed to some other descendants of the “Data” concept. These transformations generally include database lookups and filling-in of default values, and are provided for the convenience of the end users of services. Consider for example the case that some service of the Ministry of Health requires the social security number of the citizen, for identification purposes, however the citizen only has the driver license at hand. In the presence of a suitable transformation method, the service user could enter the driver license number and this would be mapped to the social security number, thus the service would proceed. Similarly, the citizen’s identity number and country could be derived from the passport number, while an Austrian VAT number can be transformed to a European VAT number by prepending the constant string “AT”. Such transformations can be automatically performed by the service execution module.

D. Service Execution

When a service consumer wants to execute a web service the entry point is to query the ontology registrar. This can be done by either navigating through the ontology aiming to locate the appropriate result, that being a document or data or by searching using keywords. Once the result is found the service execution module can search for a service that produces the requested result and when found the service consumer can request the execution of the service.

The service execution module can automatically create a list of the input necessary to invoke the web service(s). In the simplest case there will be one web service that can be invoked. In such a case the service consumer will provide the values for the input data.

If the requested result needs the synthesis of a composite service (because the selected service needs as input a document that some other service produces) the service execution module will query the ontology to find the services needed to complete intermediate steps. One such case can be when the service that produces the desired result requests as input one or more documents. Through the ontology the service execution module will retrieve the services that produce the documents and create a union of all the input parameters needed.

Once the set of the required input (including possible alternative input) is collected the end-user can execute the service. One issue that has to be tackled is that of the end-user interface. As mentioned earlier web services have an interface that allows them to be called, usually by other services rather than end-users. Therefore, for the proposed architecture to be complete, it has to cater for the interaction between end-users and the active blackboard. This can be achieved by dynamically creating a form for the end-user. For the creation of this form, the blackboard exploits the attributes that have been defined for the data within the ontology, such as value lists and validation checks. Data elements with value lists can for example be rendered as closed-selection combo boxes, while validation checks can be translated to code for the appropriate service delivery environment, such as Javascript for web browsers, XForms constraints [12] for native XForms interfaces and so forth.

Since the necessary input has been established, a cue-card paradigm can be used to guide the user to fill-in the input fields. According to it the user can be asked to provide values for input fields or to skip fields and give alternative input. If for example, the user wants a tax certificate this can be issued by providing the VAT number. If however, the user does not remember the number alternative input can be given such as the personal identification number. Values for fields participating in the service implementation constraints are also collected, since the execution engine will exploit these values to determine which particular service implementation can be invoked to satisfy the user request.

Finally to improve the efficiency of the service execution module caching mechanisms for dynamic objects (e.g. [13] [14]) may be employed. In general, documents and certificates produced by public authorities have a *period of validity*, within which they may be used in transactions. For example, a tax clearance certificate, attesting that the citizen has no due taxation debts, may be valid for three months after being issued and within this time period it may be used in the context of any transaction with the government. The blackboard exploits this property of the documents, in order to re-use documents issued by services, avoiding thus service re-execution and obtaining benefits both in terms of reducing overall system load and minimizing total service delivery time.

5. USAGE SCENARIO

To better illustrate how the architecture will work we will present a usage scenario for a real world service. Consider the case of a European citizen requiring health service in a European country different than that of her origin. Health authorities can retrieve through the blackboard the necessary web service, which provides the citizen's health record. The service registrar will help retrieve the appropriate service by means of the ontology registrar. Through the ontology, the instantiation of the health administration and the implemented services provided by it can be found. Alternatively, the end-user may select the "Health record retrieval" generic service (independently of the offering administration) and delegate the responsibility for choosing the appropriate concrete implementation to the execution engine, which will base its decision on the value of the "Nationality" input, which will be provided by the service end-user. Once the service has been chosen, the service execution engine can formulate the appropriate web page to send to the client, through which the user input will be collected. The web page may offer options to the end-user regarding the data that have to be provided, exploiting the input alternatives declared in the ontology registrar and/or the available transformation methods. For instance, if the service designer has designated the passport number and the social security number as alternatives for identifying the requested health record, the end user may provide either piece of information; moreover, if a transformation that maps driver license numbers to social security numbers (or passport numbers) exists, then the driver license number will also be a plausible input. In the latter case, the execution engine will firstly employ (transparently to the service end-user) the transformation to map the driver license number to the respective social security number, and will afterwards invoke the service, providing to it the result of the mapping as input.

6. CONCLUSIONS

In this paper the basic architectural components that form an active, ontology based electronic blackboard were presented. The architecture allows interoperability between the registered services and offers service consumers a smart method for service invocation. The ontology employed in this architecture incorporates rich semantics to model different aspects of electronic service provision, including public authority entities, data and documents, abstract services and concrete service implementations, as well as the relationships between them. The ontology can be used by end-users, to locate the services they need to execute and by the service execution engine to satisfy user requests through simple service invocation or by dynamic service composition and service workflow execution. The active blackboard also undertakes the responsibility for formulating appropriate user interfaces for the services it publishes.

According to the scheme presented above, the blackboard is a centralized entity, undertaking a number of tasks; thus the blackboard is both a potential bottleneck and a single point of failure. Replication of the blackboard and the associated repositories can be used for addressing these problems, however repository consistency has to be sought after. Request execution handover, to dynamically balance system load and exploit the network proximity of resources are also issues that will be addressed in the context of future research.

REFERENCES

[1] Newcomer E. Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison Wesley Professional, 2002. ISBN: 0201750813

[2] Tambouris E. "An Integrated platform for Realising Online One-Stop Government: The eGov Projet", in: Proceedings of the DEXA International Workshop "On the Way to Electronic Government", IEEE Computer Socity Press, Los Alamitos, CA, 2001, pp. 359-363

[3] Bunting D. et al. Web Services Composite Application Framework (WS-CAF), Ver1.0, 2003, Available: http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf

[4] Casati, F., Ilnicki, S., Jin L.J., Krishnamoorthy V., Shan M.C. "Adaptive and Dynamic Service Composition in eFlow". Proceedings of Advanced Information Systems Engineering: 12th International Conference, CAiSE 2000, Stockholm, Sweden, 2000. pp. 13-31.

[5] Evren Sirin, James Hendler, and Bijan Parsia. "Semi-automatic composition of web services using semantic descriptions". In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003, Angers, France, April 2003

[6] M. Klein, A. Bernstein. "Searching for services on the semantic web using process ontologies", The Emerging Semantic Web - Selected papers from the first Semantic Web Working Symposium, Editor(s): Cruz, Decker, Euzenat, McGuinness; 2002, IOS press, Amsterdam, pp. 159-172

[7] Erman, L., London, P., Fickas, F. "The Design and an Example Use of HEARSAY-III". Proceedings of the 7th International Joint Conference on Artificail Intelligence, 1981. pp. 409- 415.

[8] Carver, N., Lesser, V. "Evolution of Blackboard Control Architectures". Expert System with Applications, Vol. 7, 1994, pp. 1-30

[9] B. Amann, I. Fundulaki, "Integrating Ontologies and Thesauri to Build RDF Schemas", Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, 1999, pp. 234 –253.

[10] KAON development team. KAON web site, 2004. Available: http://kaon.semanticweb.org/

[11] Sun Microsystems. Java^{TM} Remote Method Invocation, 1999. Available: http://java.sun.com/j2se/1.3/docs/guide/rmi/index.html

[12] W3C. XForms - The Next Generation of Web Forms, 2004. Available: http://www.w3.org/MarkUp/Forms/

[13] Zhu H., Yang T. "Class-based Cache Management for Dynamic Web Content", IEEE INFOCOM, 2001, pp. 1215-1224

[14] Vassilakis C., Lepouras G.. "Controlled Caching of Dynamic WWW Pages". Proceedings of the DEXA 2002 conference, , 2002, pp. 9-18.