Offering Open Hypermedia Services to the WWW: a step-by-step approach for developers

Nikos Karousos
Research Academic Computer Technology Institute
Dept. of Computer Engineering & Informatics
University of Patras
26500 Rion, Greece
+30 936 648648
karousos@cti.gr

Ippokratis Pandis
Dept. of Computer Engineering & Informatics
University of Patras
26500 Rion, Greece
+30 932 193088
pandis@ceid.upatras.gr

Siegfried Reich
Sun Technology Research Excellence Center (SunTREC)
Salzburg Research, Austria
+43 662 2288 461
sreich@salzburgresearch.at

Manolis Tzagarakis
Research Academic Computer Technology Institute
26500 Rion, Greece
+30 2610 960381
tzagara@cti.gr

ABSTRACT

Hypermedia systems and more specifically open hypermedia systems (OHS) provide a rich set of implementations of different hypertext flavors such as navigational hypertext, spatial hypertext or taxonomic hypertext. Additionally, these systems offer component-based modular architectures and address interoperability between hypertext domains. Despite multiple efforts of integrating Web clients, a widespread adoption of OHS technology by Web developers has not taken place. In this paper it is argued that Web Services - which offer a component model for Web applications - can be integrated in OHSs. An architectural integration is proposed, a step-by-step process is outlined and an example of integration is provided. This very approach is aimed to benefit both worlds: the Web community with new rich hypermedia functionality that extends the current navigational hypermedia; and the OHS community by opening its tools and platforms to the many developer groups of the Web community.

Keywords

Open Hypermedia Systems, Hypermedia Services, Web Services, Babylon System.

1. INTRODUCTION

During the past decade, considerable research on Open Hypermedia Systems (OHSs) was contacted in order to provide services of structuring and accessing information. Several hypermedia application domains were introduced, apart from the classical navigational, such as spatial [20] and taxonomic [24], so as to raise the number of cases where OHS structuring could be useful. Furthermore, the design of Component-Based Open Hypermedia Systems (CB-OHSs) [22] emphasizes the notion of services because it aims at enabling the cooperation between independent components each of which provides a specialized set of hypermedia services. The Open Hypermedia Systems Working Group (OHSWG) [23] was established in order to address interoperability problems between OHSs. The OHSWG developed perspectives for creating new inter-domain services (as opposed to intra-domain services which are valid for a single domain – such as navigational hypermedia – only). However, as it has been pointed out [21], the usage of hypermedia systems and services is narrow and failed to reach a critical mass of people.

The World Wide Web (WWW) [9], which was developed under different design principles, had the chance to meet global acceptance and rapid progress. The broad acceptance results in new services being developed continuously; additionally, new users are becoming members of the Web community, without being daunted by its well-known disadvantages and problems. Recently, the notion of *Web Service* and its well-defined specification [31] gave yet further boost in the progress of the Web, since interoperability and reusability problems of network applications seem to be addressed.

The Web Services technology was built under a perspective, which has common elements with the OHS architecture. In particular, the WWW is expanding as an environment, where several network applications submit their services in a common accessible manner. Such kind of reasoning is supported by OHSs either by CB-OHSs architecture or by new proposed architectures which differ from the classic notion of level design and are based in the distribution and cooperation between independent multiple open services [34]. Given the high expansion rate and the universal popularity of the WWW, the open hypermedia researchers are motivated to integrate their systems [1] or/and extend their system services with it.

However, the task of integration between OHSs and the WWW (or other third party client applications) is tricky and complicated. Most Web integration efforts by the Hypermedia research community are characterized by the following drawbacks: a) Although users want to use only one hypermedia service, in many cases they get to have access (and to pay) for all the OHS usability

[27]; b) Hypermedia extensions and Web integrations are usually ad-hoc implementations, created by non-standardized methodologies; c) The OHS community has so far not made great usage of standard Web technologies

This paper presents a well-defined step-wise approach for making open hypermedia services available to the broad Web community, in a platform-independent way. With respect to both the existing work in open hypermedia systems, and the open hypermedia protocol (OHP) [26], this process does not require rigid changes inside the architecture of OHSs, but proposes the creation of Web Services that map the functions of hypermedia services and operate autonomously as hypermedia clients. Hence, the usage of independent hypermedia services into the Web is made available. Moreover, like any other component technology, such as agents or also CB-OHSs, the simple fact that people "think" in components results in more usable software. OHS researchers find, through Web Services approach, a 'c – class' tool [12] to attain their goal for Web integration. 'C-class' tools are defined as the tools for tool builders (rather than 'b-class', such as Web Services, which would be tools for application developers – the 'a-class').

Section 2 briefly describes the related work for augmenting Web Services based on OHSs. It focuses on the integration techniques used so far and discusses the three considering issues listed above. Section 3 analyses the Web Services as component technology, while Section 4 presents the proposed process for providing hypermedia services in the WWW. Finally, Section 5 is dedicated to a case study for the proposed methodology over Babylon Taxonomic System [9]. Summary and future work conclude the paper.

2. RELATED WORK

2.1 Web integration efforts

During the last years, important efforts for Web integration are reported in literature. Below are presented the most important of them.

The Microcosm group [17] has performed work in this area with several efforts, most notably the Distributed Link Service (DLS) [11]. A Web server's functionality is extended by hypermedia functional (including computational links, navigational links, etc.). Clients make requests, and the corresponding anchors are integrated and translated into HTML documents on the fly.

In DHM [16], a platform-independent framework as an extension to the Web browsers is presented (DHM/WWW [15]). For that purpose, Java is used for writing a general applet that handles browser integration and communication with the Hypermedia service, and CGI scripts are used for the communication between the browser and the Hypermedia (structure) server.

Chimera [1,4] extends a WWW server written in Java to make use of the Chimera Java API . In another integration effort, a client side server (or hypermedia session server) presents a user-interface, which allows the Chimera functionality to be wrapped [1,2].

Multiple Open Services project [34] presents an innovating idea. It aims to split up services into components. Each component provides a functionally independent service. Thus, is redesigned the way in which services are provided at all layers in OHS architecture.

Recently, an interesting effort, which was presented in [7], attempted to map SoFAR system to Web Services with respectful results.

Also, recent web technologies such as eXtensible Markup Language (XML) [10] in combination with XLinks [19] and other standards provide a limited degree of open hypermedia functionality (e.g., more flexible navigational linking). In [3] the use of XML for information integration enables complex information management tasks and gives flexibility and extensibility capabilities in prototype software tools.

A more detailed analysis over the techniques used for integrating OHS with the Web in literature is presented in [1]. Also, apart from the Web integration efforts, an architectural framework for modeling third-party network application integrations with OHSs has been presented [33].

The main deficiencies of the approaches outlined above can be structured into implementation costs, missing methodology and general availability. In the following these are described in more detail.

2.2 Drawbacks of the Web integration efforts

Cost: Shackelford argues that: *"...every service has a cost associated with it…ideally, one should be able to use a service when it is needed without having to pay for it when it is not.*" [27]. Until now, in nearly every Web integration effort the client has access to the whole API of the OHS, even though only a part of the OHS functionality is needed. Thus, the cost and the complexity of the OHS service provision are multiplied. The goal is the user to be provided with simple, flexible, scalable, and well-defined services.

Ad-hoc Implementations: In most cases, the implementations that enable Web integration are ad-hoc. Instead of the existence of a (standard) methodology that integrates OHS with in the Web, each time developers want to integrate an OHS with the Web they are forced to invent their own techniques. This issue is transforming into a serious problem in CB-OHSs, where components are dynamically added or even modified, requiring from the developer to make every time new integration efforts. The lack of a concise methodology makes such integrations error-prone and difficult to maintain. The observation that the developer has to do the same effort many times is an indication to raise his level of work to the 'c-level'. Therefore, a standardized way for developers to create Web Services through OHSs would be desirable.

New Web technologies: The OHS community has so far not made great usage of standard Web technologies. The reasons therefore are manifold, the key argument being that most Web standards do not explicitly express a notion of links. Web Services as an emerging standard for component-based software development could well be a technology whose adoption by the OHS community would benefit both worlds: the OHS people would base their tools on technologies that are widely available; the Web community would benefit by the much richer structuring facilities offered by typical OHS services.

3. WEB SERVICES

3.1 What is a Web Service

Web Services are an emerging technology to reuse software as services over the Internet by wrapping underlying computing models with XML [6].

The W3C definition of a Web Service argues that: *"a Web Service is a software application identified by a URI [IETF RFC 2396], whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols"*. [31]

3.2 Model

An ordinary Web Service model consists of a SOAP (Simple Object Access Protocol) server, which interacts with its environment through a collection of operations that are network-accessible through standardized XML-messaging [30]. Using a Web Service Description Language (WSDL), one can fully describe all Web Service access details.

The establishment of a set of standards for Web Service has driven most of the developers to design and implement Web Services in a very common way. Those standards gave application developers the opportunity to use Web Services easily and effortlessly. Figure 1 presents a schema of the usage of Web Services components.

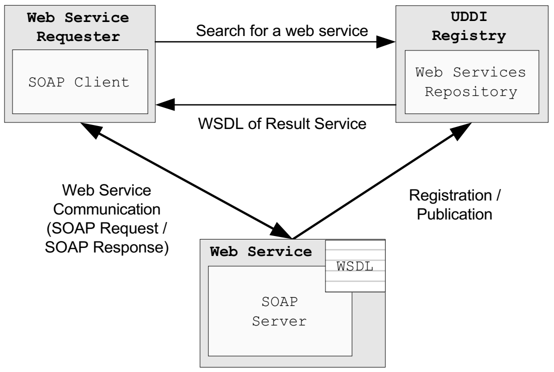

Figure 1. Web Service usage

SOAP (Simple Object Access Protocol) is a W3C standard [30], which is used to define protocols able to send simple objects in XML format. Thus, any SOAP-enabled service can receive calls for methods, passing objects in XML. The main protocol used in conjunction with the SOAP protocol is HTTP, although it can be easily used with many other protocols such as FTP or SMTP.

WSDL (Web Service Description Language) is an XML-based language, used to specify the interface provided by a Web Service. W3C standardized WSDL [32] for the description of a Web Service, enabling its use by any user that links to the certain WSDL file.

UDDI (Universal Description, Discovery, and Integration) is a registry protocol [29] for Web-based services. It is used to provide information about Web Services, and it can be private, public or hybrid. A registry entry must contain certain information about the service. Initially, it must have the WSDL file describing the service as well as the location of the runtime service. In addition, a registry entry may contain any other useful information about the service and its provider.

- A public registry provides any developer with the ability to publish a Web Service in the registry. Therefore, the registry entries are widely available to the public for searching and downloading.
- A private registry is not provided widely to the public. Instead, it is restricted to a single enterprise enabling the sharing of business components only between the enterprise clients.
- A hybrid registry is available to the public but with certain restrictions.

3.3 Advantages

The Web Service architecture is qualified by the following advantages [13]:

Interoperability. A Web Service provides hardware and software platform independence. Any client that uses the standard Web Services technologies can easily access a Web Service. By using WSDL files a SOAP client can easily be automatically created.

Encapsulation. A Web Service can easily be consolidated in any application, regardless of the internal programming details of the component.

Availability. The Web Service developer is able to publish enough information for any other developer to use, and create a Web Service client.

Modularity: Web Services provide modular advantages such as reusability and extensibility.

Self-description: Web Services have the ability to describe themselves in a way that can easily be recognized. Thus, the interface, the location and access information of the Web Service is identified by any external application.

Public ability: Web Services descriptions are provided through a wide public repository, and can be found and used by any user.

Summarizing, in contrast to using HTTP as a transport layer protocol with proprietary semantics attached to it, the notion of Web Services allows for the semantics of the remote method invocation to be explicit and thus to processed by software components autonomously. In combination with the exchangeability of the underlying transport protocol and the wide availability of implementations, this results in a new quality for component-based software development in Internet environments.

4. PROPOSED APPROACH

The main goal of this paper is to prescribe a process for creating Web Services and mapping functions of hypermedia services to operations of Web Services and vice versa. The proposed implementation process provides independence from the overall OHS architecture, and this is the main advantage from the other techniques described so far. Particularly, as shown in figure 2 that presents the proposed architecture, it is indicated that neither the structure of the hypermedia components nor their communication protocol should be changed when an OHS developer wants to publish a service on the Web.

It must be clarified that the target is not the OHS integration of a Web Browser or any other client side application. Instead, this implementation process targets on mapping structure servers to Web Services in order to make OHS features accessible through the Internet. Those Web Services behave as clients for the OHS but they also behave as servers to the SOAP-aware applications.

As discussed above, the OHS community has mainly focused on the integration within third party clients with a view to provide hypermedia services to a number of user-friendly applications. Those attempts usually use either wrappers or communicators or other communication applications of structure servers, so that the clients are provided with hypermedia awareness. Typically, wrapper-implementations of clients of OHSs are ad-hoc implementations. Each time, an OHS integration with an additional client application is needed, the necessity to create another ad-hoc communicator arises. Therefore, by adopting the proposed methodology, the effort for integrating new clients will be significantly reduced. Technically, in order to integrate a hypermedia service, the creation of a corresponding SOAP server is required. Furthermore coding of communication shims may be necessary. The notion of communicator has been referred in [33]. However in the proposed approach this communicator resides on server side and has to be created only once for a specific hypermedia service

The mapping of the structure server to Web Service creates a new middleware service. Hence, every time an application needs OHS services, the steps of integration method are simple. The generation of SOAP client can be done automatically (based on Web Service corresponding WSDL file). By the use of the widely accepted Web Services, the provision of a set of hypermedia services in the Internet through a common communication platform is achieved.

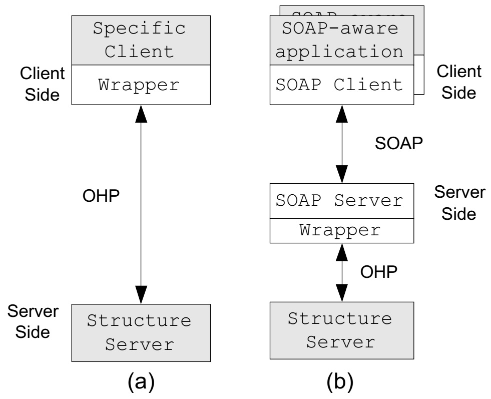

Figure 2. The classic (a) and the proposed (b) architecture for providing OHS services to external applications.

4.1 Steps

The steps of the proposed implementation process are the following:

Service specification: The service design starts with the specification of the service that will be created. Thereby, the service task, the target user group and the operations provided must be defined. A very important aspect of this step is the fact that there is no actual need to provide one-to-one mapping to every service provided by the OHS. What is more, there is no need to define a Web Service that maps every function of the structure server to operations. The services can emerge by a single hypermedia structure server or even by a subset of the functions supported by a structure server in an OHS. For example, the data navigation service, without authoring rights, in an OHS could be provided as a service on its own. Furthermore, versioning, storage, transaction and searching services could be published as independent Web Services.

A single Web Service is consisted of a set of simple operations. For each operation, there must be a specification of the name, the description, the type, the way it is called by a client, and the parameters it receives and passes. Every operation can map one ore more functions of the structure server to Web Service.

WSDL file creation: As long as the definition of the specifications is set, a description of the service is created using WSDL in a simple XML format. The latter WSDL file will comprise of the 'guide' for the development of the service as well as of the creation of the SOAP clients.

Even though, the WSDL file describes the Web Service API with a well-defined way, Web Service documentation has to exist. This documentation needs to describe specific details about the client behavior and the calling sequence of Web Service's operations giving to clients the hypermedia perspective of the Web Service.

The Web Service implementation assumes two basic characteristics: the existence of the WSDL file of the Web Service and the knowledge of the API of the structure server. The form that the OHS API is provided is not affecting the implementation procedure. The methodology does not alternate the functionality of the OHS, but, on the contrary, extends each of its services in a very simple and clear way.

SOAP Server creation: The development process continues with the implementation of a SOAP server, which can use the OHS or structure server functionality through its API. This is a fairly simple procedure because most of the present programming tools can automatically create SOAP servers and clients, as they seem to be aware of their usefulness (Microsoft .net, Delphi studio 7, etc).

The created Web Service, supports several operations. For each operation a corresponding internal function must be developed. This function must accept the parameters passed by the client as a service request. Furthermore, the function must execute one or more calls to the OHS, via its API, and then return a response. Communication between Web services and OHS structure servers requires data sets to be converted from structure server-supported data format into XML structures according to the corresponding WSDL file.

The internal functions of the Web Services are divided to a) simple mapping functions and b) complex functions. The simple mapping functions are those that map with one-to-one relationship the corresponding functions of structure servers. On the other hand, the complex functions call several functions of structure servers and combine the received results, aiming to provide clients with the requested information. Every complex function accomplishes a specific task, and has its own algorithm that can be presented with a diagram of the used hypermedia functions and the other data processes.

In order to clarify the ideas stated above, the CB-OHS conceptual architecture [22] will be the guide. The Web Service is implemented as a hypermedia client. The SOAP server substitutes the client layer that uses the structure server API. In addition, SOAP functions take the place of the client interface through a client API, which is the API of the Web Service described by the WSDL file. This creation procedure results in a new middleware layer, which acts as client for structure server and as server for third party applications (client layer).

SOAP Client creation: The next phase is to develop a simple SOAP client. The SOAP client is used for testing the communication of the Web Service. It is worth mentioning that there are various tools that can generate a SOAP client automatically. These tools only need to know the URI of the WSDL description file of the service.

Along with the creation of a SOAP client, a simple user interface is needed. The user interface is required so as the functionality of the Web Service to be checked.

Publication: The proposed methodology is concluded by the publication of the Web Service. A Web Service is published when it is registered using UDDI. Hence, the discovery procedure of the Web Service is possible. Additionally, new description languages have been created, such as DAML-S [5], providing semantic information about Web Services.

Despite the fact that a typical Web Service publication is formed only by the UDDI registration, there is also the need of demo software that is available to the users and demonstrates the usage of the Web Service (e.g Google Web Service API and Demo Software [14]).

4.2 Advantages

The implementation process described above provides a set of advantages in conjunction with the existing Web integration technologies. Firstly, Web Services provide by themselves a set of advantages that have been mentioned before. Secondly, the proposed implementation process provides the following additional advantages:

- Since the Web Service creation mechanism takes place in the client layer of the structure server, neither structure or protocol alteration, nor the OHS operations modification is required.
- It provides the capability of creating selective services from the whole set of the hypermedia services.
- The amount of the creation time for such service is fairly reduced. Most of the times, the operation source code of the Web Service is identical to the OHS client layer functions. In addition, most development tools fully support SOAP service creation.
- It does not affect at other OHS services and it can simultaneously operate with other components and clients of the OHS.
- It can easily be extended. In such cases, the only required tasks are the enrichment of the WSDL description file and the additional created operations implementation.

4.3 Drawbacks

The main drawback is encountered in the early steps of the proposed methodology. That happens because it is difficult to define a Web Service with simple operations, from a complicated system.

Besides, when the OHS API or the content type changes dynamically, it may be is impossible to completely define the request and response parameters or even the operation at all. WSDL descriptor file cannot be dynamically changed.

Finally, some importnant issues like QoS, transaction iformation and sequrity issues are not yet fully researched and standarized in Web Services.

5. CASE STUDY: MAPPING BABYLON SERVICES TO WEB SERVICES

The main goal, that led to proposing this methodology was to extend some of Babylon System services as Web Services. A brief overview of the Babylon System is presented next (a more detailed description can be found in [18]), as well as a Web Service creation example.

5.1 Architecture and Specification Of Babylon System

5.1.1 Overall

The Babylon [1] system is based on the idea to create an integrated framework aiming to provide multiple categorization services using abstractions met in taxonomies. The main goal is to provide developers with an easy and fast way to construct Internet services for item categorization.

The model has the following features: The main entity of the data model is the object entity called ‘item’ in which several characteristics are assigned. Each item can be inserted in a single category. Each category is identified by a group of characteristics and may contain a set of items or other categories, formulating a tree-structure. The system supports the creation of category shortcuts in other categories of a tree, giving the notion of relevance. Finally, the meaning of association is also defined; a specific relationship between different tree categories that are being used for automatic tree-to-tree item transferring and sharing.

During the design and implementation of Babylon System, there was special care for two innovating features for OHSs a) user management and category sharing capabilities and, b) association capabilities between similar categories of different taxonomies, targeting on automatic update of all the common interest categories. This feature allows end users to get notified when associated categories of other taxonomies change (e.g. when items are added or deleted).

The targeted task is to redefine Babylon System as an integrated component supporting taxonomic services in CB-OHSs such as Callimachus [28].

5.1.2 Architecture

Babylon’s architecture consists of three layers: a) the storage layer that is responsible for storing and managing the structural and non-structural information imported into the system, b) the taxonomic management layer that provides taxonomic creation and manipulation services and c) the client layer, which consists of either independent applications or several tree-structure applications that provide tree service development capabilities. The conceptual architecture of Babylon system is presented in figure 3.

The storage layer and the taxonomic management layer comprise of the basic infrastructure that provides taxonomic services for developers. More specifically, the storage layer is the repository of the system, composed by the structure storage system and the data storage system. The structure storage system stores and organizes the structural information while the data storage system is responsible for the user-imported data. Besides, the taxonomic layer provides basic taxonomic management capabilities to the system by serving structure and content management as well as sharing and comparison requests. The kernel of this layer is a taxonomic structure server, which is responsible for the taxonomic structure creation, management and storing as well as the data categorization requests. The communication with the client layer is achieved using the HTTP protocol, sending the data in a very simple XML format.

[1] Inspired by the hanging gardens of Babylon

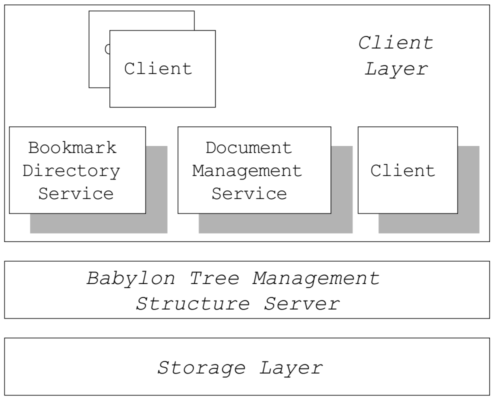
Client
Client Layer
Bookmark Directory Service
Document Management Service
Client
Babylon Tree Management Structure Server
Storage Layer

Figure 3. Conceptual Babylon architecture. The level division and the services naming was made according to the CB-OHS architecture.

5.2 Offering Babylon Taxonomic services through Web Services

The target is to extend some Babylon Services into the Internet as Web Services. Below there is an example for the creation of such a Web Service.

Service specification: The Babylon System is currently managing a set of taxonomic structures, each one of which is used for the categorization of type-specific pieces of information. One of these taxonomies is called Babylon URL Manager. Web Users can perform navigation and authoring tasks using the Babylon URL Manager in a Web directory service.

Next, we define a simple service called Babylon_WebDirectoryService that provides free navigation capabilities over Babylon URL taxonomies. The service provided to public, prohibits the authoring either of the tree or the data.

The WebDirectoryService communicates with SOAP over HTTP. The available operations are:

1. DoOpenCategory: SOAP Request – Response Action - The operation opens a category and returns all the sub-categories and the urls included.
2. DoGetCategoryPath. SOAP Request – Response Action - The operation returns the absolute path of the categories that a user must follow to find a certain category.
3. DoGetUrlCategory. SOAP Request – Response Action - Returns the category in which a specific url is located.
4. DoSearchUrl. SOAP Request – Response Action - Returns the categories and their urls by their title.

WSDL file creation: For each operation, we define the parameters and finally we create the Babylon_WebDirectoryService WSDL file [8].

SOAP Server creation: Following the proposed methodology, we create a PHP-SOAP server, using a corresponding PHP-SOAP unit [25], in which we define the URI of the WSDL file.

Afterwards, for each operation defined, a function that accepts and returns the proper parameters is created. For instance, the DoOpenCategory function just receives the category id and then calls the Babylon server's procedures GetChildren(idcategroy) and GetNodeItems(idcategory). Then, it converts the data (using XSL and XML parsing) in SOAP oriented data. This is a complex function. On the other hand, the functions used for the last three operations (DoGetCategoryPath, DoGetUrlCategory, DoSearchUrl) are simple mapping functions. These functions just call the corresponding Babylon server function and return the data in SOAP format.

SOAP Client creation: A PHP-SOAP (testing) client is created, based on the corresponding WSDL file and the communication with the Web Service is tested. Next, a plain Web Interface is created so the Web Directory Service Tree to be visible. A demo site of Babylon's Web Service is located in [8].

Publication: The last important task is to publish the created Web Service through UDDI. Also, it is rather simple to create a corresponding demo version. This is achieved by a slight transofmation of the allready created testing client. Then it is published as an Downloadable Demo Version.

5.3 Resulted architectures

The implementation process, which was presented in the previous subsection, can act as a "guide" (a 'c-class' approach) so that new Web Services are created or existing Web Services are altered. The conceptual architecture when a new service is added in Babylon System is altered to the schema presented in figure 4, in which Web Service requester is any SOAP-aware network application. In figure 4 the combination between the former conceptual architecture schema of Babylon System (figure 3) and the schema of the usage of Web Services (figure 1) is obvious.

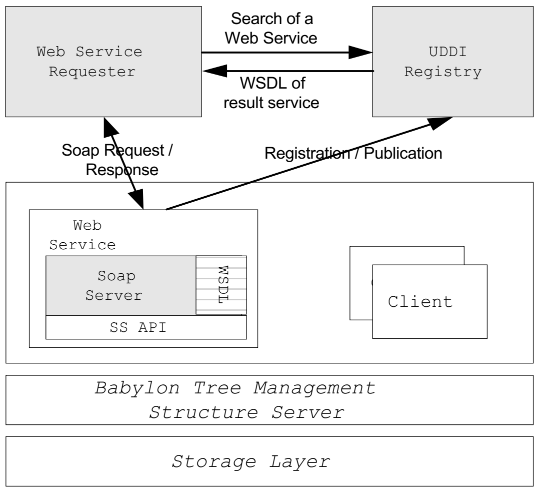

Figure 4. Mapping a Babylon Service to Web Service.

An interesting aspect of the proposed Web Service creation process occurs when someone tries to create a Web Service that has mapped all the services provided by the OHS. In this case, a central Web Service is implemented. This central Web Service can serve other specific-purpose Web Services. In this way, the whole OHS API is extended through a single Web Service. The main drawback of this aspect is that in most cases this implementation will be rather tricky due to the complexity of the operations of OHSs components.

Nevertheless, Babylon system exhibits a controlled complexity at the functional level. Therefore, the implementation of a central Web Service is possible, for the process of a large set of taxonomic services. By this way Babylon system's services are offered through a single Web Service. The resulted architecture, in this case, for Babylon System is presented in figure 5.

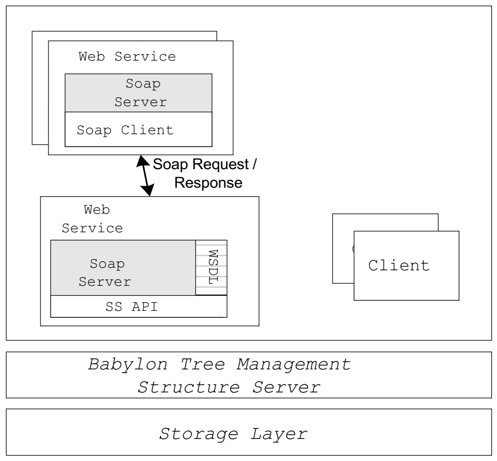

Figure 5. Full mapping of Babylon structure server.

6. FUTURE WORK

The future work of the approach presented in this paper focuses on the implementation of a set of tools that can automatically produce Web Services from UML schemas.

There are a variety of tools available that can create a WSDL file from an IDL definition of a service. In addition, most of the modern programming languages and some stand-alone tools can generate the SOAP Server and SOAP client code or executable based on a giving WSDL file. The main goal is to build an integrated developer framework able to generate Web Service skeleton sources that have hypermedia awareness and map every hypermedia structure server function. This framework will be able to create also automatically Web Service clients that support the SOAP communication with Web Service.

The Construct development environment [34, 35] provides development tools that assist the system developers in the generation of set of services that make up a hypermedia system [35]. Construct could be extended in order (a) to support IDL to WSDL tools and (b) to rebuild CSC (construct service compiler) so that it creates also a mapped Web Service based on the structure server, as outlined in figure 6.

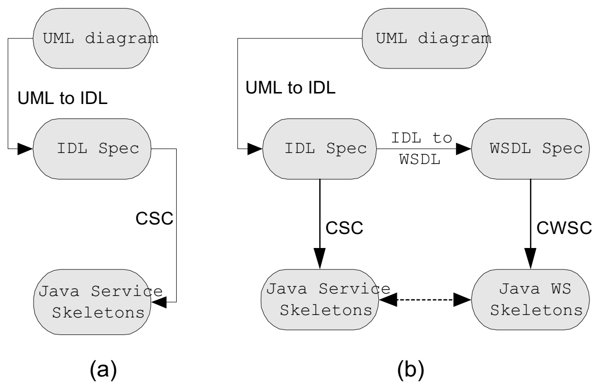

Figure 6. The current (a) and the proposed development process (b) with the Construct development tools. CWSC stands for Construct Web Service Compiler.

7. CONCLUSIONS

This paper argues for a step-wise approach to make the rich functionality of Open Hypermedia Systems (OHS) available to the web community. Using well-defined and accepted component-based technologies such as Web Services will provide a broad platform for further usage and extension for web developers. Additionally, it was shown an example of integration with the Babylon system.

It is believed that this approach offers multiple benefits: by providing open and standard services the OHS community benefits by the acceptance and popularity of existing tools and services, the web community on the other hand can build on well established research prototypes; finally, end-users will benefit by the unique features that can be offered only by OHSs.

8. ACKNOWLEDGMENTS

The authors would like to thank Ioannis Panaretou from the Business Management & Administration Department at the University of Patras for his support during this research.

REFERENCES

[1] Anderson, K. M. (1997). Integrating Open Hypermedia Systems with the World Wide Web. In Proceedings of the 1997 ACM Hypertext Conference, (Southampton, UK).

[2] Anderson, K. M. (2001). The extensibility mechanisms of the Chimera open hypermedia system. In Journal of Network and Computer Applications, 24, pp. 75-86.

[3] Andreson, K. M., and Sherba S. A. (2001). Using XML to support Information Integration. Proceedings of the 7th Workshop on Open Hypermedia Systems, Aarhus, 2001. Lecture Notes in Computer Science (LNCS) 2266.

[4] Anderson, K. M., Taylor, R. N., and Whitehead, E. J. (1994). Chimera: Hypertext for heterogeneous software environments. In Proceedings of the ACM European conference on hypermedia technology (ECHT '94), Sept. 18-23, 1994, Edinburgh, Scotland, UK, pp. 94-107.

[5] Ankolenkar, A., et al. (2002). DAML-S: Web Service Description for the Semantic Web. Presented at The First International Semantic Web Conference (ISWC), 2002.

[6] Aoyama, M., et al. (2002). Web Services Engineering: Promises and Challenges ICSE'02, 02, May 19-25, 2002, Orlando, Florida, USA.

[7] Avila-Rosas, A., Moreau, L., Dialani, V., Miles, S., and Liu, X. (2002). Agents for the Grid: A comparison with Web Services (part II: Service Discovery). In AAMAS'02, July, 2002, Bologna, Italy.

[8] Babylon System's Web Service Demo. At URL: <http://www.optionsnet.gr/babylonproject/Webservice>.

[9] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. (1994). The World-Wide Web. Communications of the ACM, 37(8), pp. 76-82.

[10] Bray, T., Paoli, J., and Sperberg-McQueen, C. M. Extensile Markup Language (XML) 1.0, W3C Recommendation, 10-Feb-1998. At URL: <http://www.w3.org/TR/REC-xml>.

[11] Carr, L., et al. (1995). The Distributed Link Service: A Tool for Publishers, Authors and Readers. In Fourth International World Wide Web Conference: "The Web Revolution". Boston, Massachusetts, USA.

[12] Engelbart, D. (1998). HT'98 OHS Wokshop Keynote Address.

[13] Fremantle, P., Weerawarana, S., and Khalaf, R. (2002). Enterprise Services, Communications of the ACM, October 2002/Vol.45.No 10, pp.77-82.

[14] Google Web APIs Home. At URL: <http://www.google.com/apis/>.

[15] Gronbaek, K., Bouvin, O. N., Sloth, K. (1997). Designing Dexter-based Hypermedia Services for the World Wide Web. In Proc. of Hypertext '97. Southampton, UK, pp. 146-156.

[16] Halasz, F., and Schwartz, M. (1994). The Dexter Hypertext Reference Model. Communications of the ACM, 1994, 37 (2), pp. 30-39.

[17] Hall, W., Davis, H., and Hutchings, G. (1996). Rethinking Hypermedia: The Microcosm Approach. Kluwer Academic Publishers, Norwell, Massachusetts, USA.

[18] Karousos, N., Panaretou, I., Pandis, I., and Tzagarakis, M. (2002). Babylon Bookmarks: A Taxonomic Approach to the Management of WWW Bookmarks. MIS'02, (Esbjerg, Denmark).

[19] Maler, E., and DeRose, S. (1998). XML Linking Language (XLink). At URL: <http://www.w3.org/TR/1998/WD-xlink-19980303>.

[20] Marshall, C., and Shipman, F. (1995). Spatial hypertext: Designing for change. Communications of the ACM, 38(8), pp 88-97.

[21] Nürnberg, P. J., and schraefel, m. c. (2002). Relationships Among Structural Computing and Other Fields. JNCA Special Issue on Structural Computing, 2002.

[22] Nürnberg, P. J., Leggett, J. J., and Wiil, U. K. (1998). An agenda for open hypermedia research. In Proceedings of the 9nth ACM Conference on Hypertext, June 20-24, 1998, Pittsburgh, PA, pp. 198-206.

[23] Open Hypermedia Systems Working Group – OHSWG WWW site. At URL: <http://www.cs.aue.auc.dk/ohswg>.

[24] Parunak, H. (1991). Don't link me in: Set based hypermedia for taxonomic reasoning. In Proceedings of the 1991 ACM Hypertext Conference, (San Antonio, TX, Dec), ACM Press, pp. 233-242.

[25] PHP-SOAP Unit (NuSOAP). At URL: <http://dietrich.ganx4.com/nusoap/index.php>.

[26] Reich, S., Wiil, U. K., Nürnberg, P. J., Davis, H. C., Gronbaek, K., Anderson, K. M., Millard, D. E., and Haake, J. M. Addressing interoperability in open hypermedia: The design of the open hypermedia protocol. Special issue on open hypermedia. The New Review of Hypermedia and Multimedia, 5, pp. 207-248.

[27] Shackelford, D. E., Smith, J. B., Smith F. D. (1993). The Architecture and Implementation of a Distributed Hypermedia Storage System. In Proceedings of the 1993 ACM Hypertext Conference, (Seattle, WA, Nov), ACM press, pp. 1-13.

[28] Tzagarakis, M., Avramidis, D., Kyriakopoulou, M., Schraefel, M., Vaitis, M., and Christodoulakis, D. (2002). Structuring Primitives in the Callimachus Component-Based Open Hypermedia System. JNCA Special Issue on Structural Computing, 2002.

[29] Universal Description, Discovery and Integration of Web Services (UDDI). At URL: <http://www.uddi.org>.

[30] W3C Simple Object Access Protocol (SOAP). At URL: <http://www.w3.org/tr/SOAP>.

[31] W3C Web Services Architecture Domain. At URL: <http://www.w3.org/2002/ws/>.

[32] W3C Web Services Description Language (WSDL). At URL: <http://www.w3.org/tr/WSDL>

[33] Whitehead, E. J. (1997). An Architectural Model for Application Integration in Open Hypermedia Environments. In Proc. of Hypertext'97, (Southampton, UK), pp. 1-12.

[34] Will, U. K., Hicks, L. D., and Nurnberg P. J. (2001). Multiple Open Services : A New Approach to Service Provision in Open Hypermedia Systems. In Proceedings of the 2001 Hypertext, (Aarhus, Denmark), pp. 83-92.

[35] Wiil, U. K., Nürnberg, P. J., Hicks, D. L., and Reich, S. (2000). A development Environment for Building Component-Based Open Hypermedia System. In Proc. of ACM Hypertext'00, (San Antonio, TX), pp. 266-267.