process coordination, distributed computing,
kidney transplantation, internet technology,
web services architecture

Wiktor WANDACHOWICZ*, Zbigniew FILUTOWICZ *

APPLICATION OF WEB SERVICES ARCHITECTURE TO INFORMATION SYSTEM FOR COORDINATION OF KIDNEY TRANSPLANTATION

Cooperation Supporting System (CSS) is a project with one goal of prevailing importance: to improve information exchange between all participants of the kidney transplantation process. The project designed for the Regional Dialysis and Transplantation Centre in N. Copernicus Hospital in Łódź is carried on by the Artificial Intelligence and Software Engineering team of Institute of Computer Science. Since kidney transplantation engages many institutions, quick and effective flow of information plays a key role in the whole process. Given the current state of the system and the new concepts of its development, we decided to restructure the communication layer of the system. The present paper discusses how this change influences the overall system design, capabilities of the system, possibilities of future enhancements as well as possible integration with other information systems. To this purpose, the Web Services standard is presented and a brief overview of the evolution of distributed computing technologies is also given.

1. INTRODUCTION

Dialysis is the only form of keeping a human with chronic kidney incapacity alive. Moreover, it creates a possibility for a patient to wait for a kidney transplantation [1]. In order to increase the number of dialyses there are attempts to optimise the treatment as well as to estimate the dialysis process accuracy [2, 3]. However, in order to improve the health of the patients and their life standard it is essentially better to perform a kidney transplantation. Rehabilitation, quality and efficiency of transplantations, and accessibility of such services constitute a great problem in health care. In most cases, the kidney's donor is a person who died in an accident or during an operation. The time between the death of a donor and the kidney transplantation is limited and cannot exceed half a day. Because of the number of examinations that have to be conducted on the donor's organ in the immunology lab, the selection of a right recipient and the coordination in time and place have to be carried out optimally.

Technology is able to change dramatically the way in which people work and exchange information. In the reality, however, patients' data are usually dispersed in many places. The information about a dialysed patient's state of health is kept in his or her host Dialyses Ward. This information has to be sent to the place of operation. On the other hand, provisions against transplantation are kept in Regional or National Transplantation Centres.

* Institute of Computer Science, Technical University of Łódź, ul. Wólczańska 215, 90-924 Łódź, POLAND
e-mail: wiktorw@ics.p.lodz.pl, zfiluto@ics.p.lodz.pl

The process of selecting a right recipient constitutes another complex and time consuming problem. In Transplantation Centre there are about one hundred parameters of a patient that are important and should be considered in the case of transplantation. For each patient the information is updated every three months or when something important happens (e.g. patient dies). There may also be some important factors that may be revealed only at the moment of the decision making. All this influences the transplantation coordination process and increases the risk of unsuccessful intervention (including cancellation of intervention).

2. HISTORICAL NOTE

The work that is currently in progress involves the Institute of Computer Science of the Technical University in Łódź and its medical partner, N. Copernicus Hospital in Łódź. The research is focused on a Cooperation Supporting System (CSS), designed to help the decision makers involved in kidney transplantation. The system is already in operation in the Regional Dialysis and Transplantation Centre at Copernicus Hospital in Łódź.

The work concentrates on two areas – a back-end system for Dialysis Centre [4] and a system supporting the transplantation coordination. Our attempt to solve the latter problem started in the year 2000 [5]. At that point we tried to apply simple solutions (that is, an ASP technology), which eventually proved to be insufficient. During 2001 we decided to re-engineer the back-end system for Dialysis Centre [6] in order to easily integrate it with the transplantation coordination system. We also decided to analyse further the problem of creating a distributed system using internet technologies [7]. However, the communication connecting all parties involved in the transplantation coordination process was still a difficult task to solve. A turning point took place next year.

In December 2001 we encountered a new, promising technology – XML Web Services and it became obvious to us that our efforts should concentrate on this tool. However, before discussing the Web Services in more detail, let us investigate the requirements for the CSS.

3. INFORMATION REQUIREMENTS FOR COORDINATION OF TRANSPLANTATIONS

The Cooperation Supporting System that operates in the Regional Transplantation Centre is a key element of the architecture. It constitutes a central place for storing information about patients and their health condition, donors, kidney transplantation protocols, etc. The sources of current information include: Dialyses Ward (patients' quarterly data), Intensive Care Ward (information about the organs to transplant), Immunological Laboratory (medical tests for donors and recipients), National Transplantation Centre (provisions against transplantation) [7]. All these data should be constantly updated in real-time because they influence the status of the patient waiting for a transplantation (see figure 1).

Since there is a pressing need for automation of the exchange of an increasing amount of information between all those institutions, thus the creation of CSS is fully motivated. Since, in turn, the information is dispersed in many places and information systems, and moreover, there is a variety of forms and ways of communication, the CSS must be of a distributed architecture. In order to establish an efficient way of information exchange between all the systems involved in the coordination process much care must be taken to choose the right communication architecture for the CSS. This is crucial because such an architecture has to be of "the least common denominator" character. If extensibility, easy integration and cross-platform independence become the other factors, the only possible choice is the Web Services.

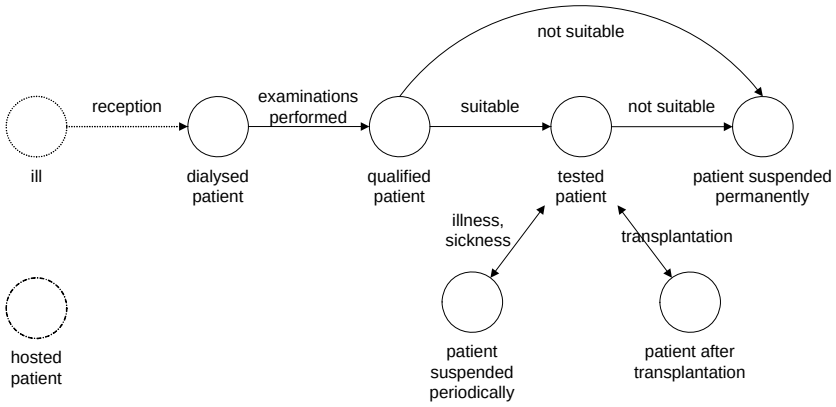

Fig.1. Schema of the status of patients with chronic kidney incapacity [5].

4. WEB SERVICES ARCHITECTURE

The driving force of the *Web Services* [8, 9] is to establish a standard of communication interface between different parts of a distributed system. Not long ago, in order to connect diverse information systems it was necessary to exploit low-level technologies (like RPC, DCOM or CORBA) to design data exchange details. It worked but at the cost of tight coupling of the integrated systems – before integration the protocols used and the development tools supporting them had to be settled. Sometimes cooperation was impossible or reconstruction of systems was too costly because of the differences between platforms. With the use of Web Services, systems can be connected in the loosely-coupled, well-defined, standard way, in which addition of just another communication layer to every system involved can be applied, instead of introduction of numerous internal changes.

The essence of Web Services is the possibility to access the self-contained, modular applications – described, published and callable through the Internet [10, 11, 12]. Moreover, Web Services are based on XML – an easily understandable, portable standard – and they use the SOAP protocol (extended HTTP protocol) as a method of transporting requests and answers, which are two properties that distinguish this technology among others. Apart form this, the abilities of each web service are described by an additional contract – an electronic document with the structure defined in a standard way.

The principles of this technology allow the use of an existing information system so that, after the addition of a web service compatibility layer, the services of the system become available

through the Internet. The internal structure of such a system is not of essential importance, and frequently it is unnecessary to change it, which has to be done with integration based on other technologies. Typically, only a few new parts are built, and the only thing that matters is that the system should implement correctly the services described in its contract. In this way, it becomes relatively easy to start cooperation with a given system and it is possible for everyone interested in it.

The business world has practically accepted the Internet as a medium of the business information exchange. Throughout the years, it has also become known that the data processing automation both increases the efficiency and decreases the susceptibility to errors – practically in all aspects of the remote data transfer. Consequently, it is possible to lower the costs and increase the flexibility of cooperation between business partners, but only when there is proper technical background for it. That is why the web services initiative has been very well received by the commercial world. On the other hand, information technology specialists have for long experienced different types of troubles using data transfer standards. Integration of software solutions from different vendors – especially of a bigger scale – has never been an easy task, and has now become even more complicated. Thus the current trend concerning web services, and the convergence of interests of business and information technology, is a phenomenon with a rather short history.

4.1. DEVELOPMENT OF DISTRIBUTED SYSTEMS TECHNOLOGY

Among the technologies used for the information exchange between elements of distributed information systems there were standards like: RPC (*Remote Procedure Call*), DCOM (*Distributed Component Object Model*), or CORBA (*Common Object Request Broker Architecture*). They allowed processing distribution so that part of it could be performed on a physically different machine or in another process, with a separated address space on the same machine. In this model the caller requested some remote processing in order to finish its own process. The intermediate layer had the responsibility to intercept the request and transfer it to the target machine. Then, the proper program or a component fulfilled this request and produced desired results. Afterwards, the intermediate layer had to carry over the response and transfer it back to the original caller.

Different technologies took different approaches to the above idea, hiding more or fewer details concerning the way of access to a remote service. The goals were to achieve the transparent distribution of application components and to perform transformation from procedural calls (like RPC) to the support of object-oriented calls (like CORBA). In the long term, the most important weakness it was the binary standards of information interchange. Every machine that turned out to be taking part in the distributed processing needed an operating system support; additional processing threads had to be created for call acceptance and work organization.

If those existing solutions worked effectively, why did people begin to leave them? In the era of global internet and communication, such solutions were insufficient. It is difficult to imagine a secure system installation handling efficiently up to tens or hundreds client of requests per minute, keeping a live connection to each of the clients and letting every one of them do their work as if it were one of the machines from their local network.

Such requirements fit internet servers (WWW servers) perfectly well. This software can handle security (in many ways), distribute the load, and has no problem with persistent connections (they last only while being used). The last point holds true due to the HTTP protocol, which is

connectionless (stateless) and scales well. Thus, it was only a matter of time to effectively combine distributed processing with the abilities of internet servers.

Internet servers have also evolved. Their primary function has always been to serve static WWW pages. Soon, however, the CGI (*Common Gateway Interface*) mechanism was introduced, which allowed execution a program on the server as a result of sending properly formatted request address. The executed program had the obligation to dynamically generate the text of WWW page, which, in turn, was sent to the requesting client. Thus this looked like it was enough for everyone to build solutions based on CGI. Later, more options were introduced, like an ability to interpret a server-side script in order to generate the WWW page (e.g. ASP or PHP). Another option was to execute programs combining functionality of both CGI and script interpreters (JSP, servlets [13]).

However, as a result of running a program or a script on the internet server, it was the WWW page that was most frequently produced. The next step was then to integrate the possibility of an information system processing distribution with the flexibility of internet servers. The solution that accomplishes both demands is Web Services.

4.2. STRUCTURE OF XML DATA

The key to the web services is XML (*Extensible Markup Language*), which is a standard defininig the syntax used to create declarative languages. Due to its structure, clarity, extensibility, and portability, XML is very likely to become the most popular common denominator in the area of information interchange. Simplicity of defining a document structure together with the information contained in it transforms into the simplicity of creating new types of XML documents for many applications. Although XML is not a remedy for all problems, its wide acceptance and various applications enable putting forward the thesis that in the upcoming years modern information systems will have the ability to send and receive data in XML format – either as a basic data format or as one of the options.

Despite its name, XML is not a language. It is rather a commonly recommended specification enabling creation of new markup languages [14]. XML is an entirely textual format, in which *text* should be understood as a sequence of symbols written in the Unicode standard [15]. The actual information is indicated by tags, just like in HTML. But in XML one can define the tags, their order of appearance, nesting, attributes etc., as well as constraints on the content of particular tags by creating data types. It can be done by specification of document type (DTD – *Document Type Definition*) or document schema (*XML Schema*). Therefore it can be said that XML documents reflect the exact structure of data and, what is interesting, they still remain readable for a machine as well as for man.

XML also features document validation. On one hand, the structure of an XML document has to conform with the standard, i.e. it has to be *well-formed*, which means that e.g. all open tags have to be properly closed, their names should refer to the definition of a document (XML recognises the case of letters), they can contain other tags but they cannot overlap, etc. On the other hand, the document is considered *valid* when additionally it is in accordance with its schema or a document type definition (as it has already been mentioned, there is a structure description of a document and data types of its particular elements). The former is performed by the *parser* (which divides the sequence of received characters into units and checks their correctness), while the latter is done by the *validating parser*, which also takes into account the description of the XML document structure.

In consequence, a system can process only valid documents, decreasing its complexity and possibility of mistakes.

4.3. ARCHITECTURE OF WEB SERVICES BASED SYSTEM

The main requirements of a system built with the use of web services are to receive properly formed requests in the form of SOAP messages, realise them and send answers also as SOAP messages. Since commands reach the system typically as requests of the HTTP protocol (possibly extended), it can be concluded that in order to handle such requests the system has to include the internet server. Although it does not have to be a physically separate machine but rather a software component, one general tendency has to be taken into account – the complexity of such a system is increased by an additional software layer. In order for the information system to communicate successfully with the environment, generally accepted standards have to be applied. Such a communication protocol for web services is SOAP.

The SOAP protocol (*Simple Object Access Protocol*) [16] was developed by IBM, Microsoft and DevelopMentor to facilitate the information exchange in a distributed environment. It is an extension to the most commonly used HTTP protocol, and it is based on sending additional information within HTTP messages. This information is called *SOAP envelopes*. Specification of the SOAP standard defines messages format (requests and answers) in XML; the format is described by a public definition of a SOAP document.

The SOAP envelope contains the call of a service including parameters. After the call is processed by the Web Service, the SOAP envelope containing the answer in the XML form is also sent back. Practical implementations of web services can process not only HTTP-SOAP requests, but also HTTP-GET as well as HTTP-POST requests – in order to cooperate with the older generation of clients. The effect of such a call is the XML document, but the SOAP envelopes are no longer used.

In order to let the other interested parties know what services are available, what parameters they need and what responses they give, the standard WSDL (*Web Services Description Language*) language has been created. It enables describing those issues in a form which can be machine processed. It is a kind of contract between a specific web service and its clients – the programs calling its methods – describing SOAP messages and a way of their exchange. A thing worth noting is that the WSDL itself is defined as an XML document with publicly available schema.

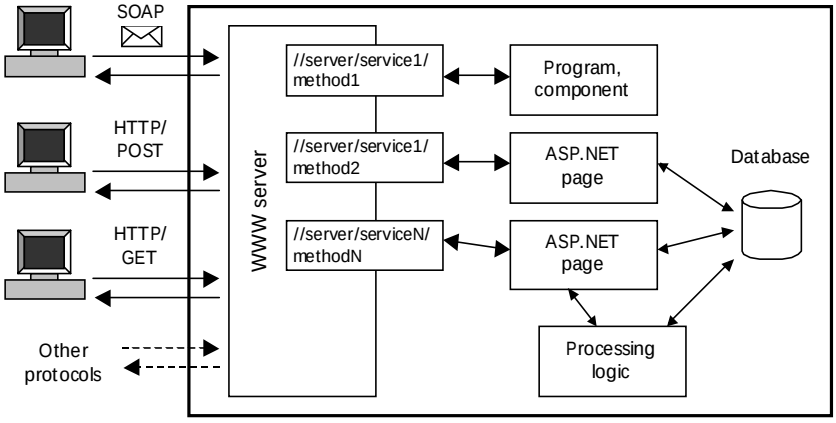

Fig.2. Web service containing services (methods), accessible through different protocols.

The WSDL document is just a collection of definitions. The web service description formed in this way contains names of available services, i.e. URI addresses (*Uniform Resource Identifier*), the protocol in which communication takes place with each service, the collection of available data as well as details of exchanged messages and formats of data sent. In short, it is a form that can be automatically processed by various electronic devices.

The WSDL document for a specific web service is needed by a party that wants to use such a web service. The UDDI standard (*Universal Descirption, Discovery and Integration*) specifies the ways of publishing registries with public descriptions of various web services [17]. By design, the description stored in the UDDI registry is highly detailed, and consequently contains enough information required for using the web service functionality by the development tools. The core activity in the context of UDDI is the process of business information publication. Three types of information are stored in the registry: company information (*White Pages*), categories of products and/or services (*Yellow Pages*), and the technical description of services (*Green Pages*) [18]. This information does not have to be made public. It is perfectly possible to create a private UDDI registry, e.g. for the intranet use only. In this case the WSDL document has to be transferred in a manner different than through the network. This has a drawback, namely, when a description of the web service is changed (expanded), the WSDL document has to be redistributed. For typical WSDL documents the UDDI registries automate this process.

From the general point of view, the Web Services give the possibility to integrate different, loosely-coupled information systems. The system containing complex functions exposes them with web services interface. Another system interested in that functionality can now send requests to the web service, thus getting desired processing. Dependency between systems is as small as possible, that is the former (publisher) describes the functionality in the form of WSDL document and schemas of XML documents for requests and answers. The latter system (consumer) is obliged to follow the protocol to send the request and receive the answer correctly. From now on it can do whatever it needs with the resulting XML document. It is not necessary to establish a proprietary format for communication or to settle a common database schema in both systems. All this is already specified by the schemas of WSDL and XML documents – and the Web Services standard describes how the whole infrastructure is going to work.

5. USING WEB SERVICES FOR DESIGN OF CSS

Implementation of Web Services technology to reconstruct an existing system supporting coordination of kidney transplantation seems promising and is being developed and tested [19]. The goal is to provide a web services based integration as the standard of communication between all systems involved in the process under investigation. The examples of data exchange using web services are:

- Receiving a kidney for transplantation the coordinator may demand a current list of patients scheduled for transplantation, as well as the information about their health, examination results, etc. (Transplantation Centres use web services provided by Dialysis Centres).
- Dialysis centres are able to submit patients as potential kidney recipients (using web services provided by Regional or National Transplantation Centres).
- Dialysis centres can perform periodic updating of information about their patients in the coordinator's system (using web service as in the previous example).
- The transplantation coordinator can obtain information about the possibility of receiving organs from a potential donor (for this Intensive Care Wards can use web services provided by Transplantation Centres).

Another interesting issue to solve is how a module supporting the donor-recipient matching on the basis of a given set of rules can be integrated into the system. The module is a decision supporting one with standard interfaces, available from the Internet as a web service in different versions provided by various vendors. It suffices to define a set of input data necessary for the matching process as well as expected output information. Then, the interface for one or two types of web services can be created: one at the coordinator's side supplying information about patients; the second one at the service provider's side suggesting the set of potential transplantation patients.

BIBLIOGRAPHY

[1] T. ORMOWSKI, *Przeszczepianie nerek*, Wydawnictwo Lekarskie PZWL, W-wa 1995

[2] J. WANIEWSKI, P. ŁUCJANEK, A. WERYŃSKI, *Alternative Description of Combined Diffusive and Convective Mass Transport in Hemodialyser*, Artif. Organs, 1993, 17, 1, 3-7

[3] M. MISRA, K.D. NOLPH, *Adequacy in Dialysis: intermittent versus continuous therapies*, Nephrology, 2000

[4] J. FILUTOWICZ, Z. FILUTOWICZ, P. SZCZEPANIAK, A. KMIECIK: *Object and Component Oriented Modelling of Complex Medical Systems*, Journal of Applied Computer Science, Vol. 7 No. 1 (1999), pp. 91-98.

[5] Z. FILUTOWICZ, J. FILUTOWICZ, A. KMIECIK, D. ZAWADZKI, P. SZCZEPANIAK: *Sieciowy system informacyjny do integracji działań podmiotów biorących udział w przeszczepianiu nerek*, „Multimedialne i sieciowe systemy informacyjne", materiały konferencyjne, C. Daniłowicz, Wrocław 2000, pp. 363-369.

[6] W. WANDACHOWICZ, A. KMIECIK, *Architecture of Multitier Information System for Dialysis Centre*, Journal of Medical Informatics and Technologies, Vol. 2/2001, pp. MI117-123

[7] A. KMIECIK, Z. FILUTOWICZ, W. WANDACHOWICZ, *Cooperation between institutions involved in kidney transplantation using internet technology*, Journal of Medical Informatics and Technologies Vol.2/2001, pp. MI131-137

[8] *Web Services Architecture*, http://www.w3.org/TR/2002/WD-ws-arch-20021114/

[9] *What is an XML Web Service?*, http://xmlu.com/tag/Article.asp?v=16&i=7&p=1&s=1

[10] DOUGLAS J. REILLY, *Designing Microsoft ASP.NET Applications*, Microsoft Press, 2001

[11] D. A. CHAPPELL, T. JEWELL, *Java. Usługi sieciowe*, Wydawnictwo RM, 2002.

[12] R. BRUNET, *Java w komercyjnych usługach sieciowych. Księga eksperta*, Helion, 2003.

[13] Sun Microsystems Inc., *The J2EE Tutorial*, http://java.sun.com/j2ee/tutorial/

[14] *Extensible Markup Language (XML)*, http://www.w3.org/XML/
[15] *Unicode Home Page*, http://www.unicode.org
[16] *Simple Object Access Protocol (SOAP)*, http://www.w3.org/TR/SOAP/
[17] *UDDI.org white papers*, http://uddi.org/whitepapers.html
[18] *Web Services Road Show - Warsaw, Poland,* http://www.architag.com/events/event.asp?id=wsef2002-Poland
[19] B. BILICKI, Using XML Web Services for cooperation of transplantation coordination systems and dialysis wards, MSc thesis, Institute of Computer Science, Technical University of Łódź, 2003