Modern business environment is characterized by constant changes and organizations must adapt quickly - they have to meet consumers’ new requirements, keep up with competitors, follow economic trends and implement innovations. Increasingly, they rely on computerized information systems (CIS) and new information technologies for enhancing the management of their activities and achieving competitive advantages.

Meanwhile, requirements concerning CIS are growing: they have to be flexible and adaptable, Internet-open and easily integrated with legacy systems. This brings forward the need for standard technological interfaces to facilitate organizations in providing or using their clients’ and partners’ information resources. Significant changes are necessary in the internal organization and structure of CIS.

All the requirements regarding today’s CIS emphasize the need for applying a web-services-oriented approach and methodology of building these systems. Web services are a dynamic and fast-developing concept, as well as a technology. They have been with us since the year 2000 and have not yet reached a stage of maturity but are widely used. These days they set the trends in the development of organizations computer information systems, enabling the improvement of organizations structure, standardizing interactions with outside systems and using Internet services. A number of experts consider the wider application of web services in building and developing an organization’s CIS “best practice” and one leading to more cost-effective flexible decisions and better integration with existing applications.

In building web services a number of standardized technologies are used such as XML, WSDL, SOAP, UDDI, WS-I, etc. As they greatly vary in character, problems arise in their uniform comprehension and use in building and developing CIS. For this reason we believe that the capacities of web services have to be represented in the CIS model at a higher level of abstraction and one that is relatively little dependent on the technologies they are based on. The UML method is a suitable instrument for this task, as it has been an established standard for specification and implementation of...
CIS. UML brings together leading software professionals’ and IT companies’ expertise in the field. Its use is methodologically secure and is supported by a number of developing environments. It is a general-purpose object-oriented modelling method as it can be used for the building, implementation and support of a wide range of systems – from the most complex corporate information systems to specific and closed embedded systems, including service-oriented ones.

Based on the above, the purpose of this paper is to define a compact and concise representation of web services basic structure and essential mechanisms by means of the UML method, emphasizing the exchange of different types of data in web services. To achieve this goal, a number of problems have to be solved:

First, define the essence of web services and, based on that, identify and systematize UML elements that adequately match their purpose and functions;

Second, study the dynamics of interactions in web services, which is the first important step in achieving data exchange between them;

Third, applying the mechanisms of formal parameters in Web services data exchange, comparing them to the message exchange schemas;

Fourth, study the capabilities of UML for representing object-exchange in web services and demonstrate how they are applied.

1. Basic elements of web services

Web services can be viewed as a technological platform for building CIS. In information technologies context the terms “web service” and “service” are often used synonymously. Web services are independent module-type applications which can be published and used in a network environment – usually the World Wide Web. Unlike other web-applications, they do not support a graphical user interface but share business logic, data and processes by means of programmable interfaces.

Consequently, a web service can be defined as a set of interrelated functions, which are used by means of interfaces. This is why, in its simplest form, a web service can be represented through UML in the CIS model as a class that enables access to its functions through its public operations (fig. 1a). For a more accurate and precise representation of web services, however, two UML elements can be used – interface and class. The interface consists only of signatures/titles of operations (i.e. functions) of the service which must be performed by the class operations (fig. 1b). Thus it is possible for one or several interfaces to be explicitly present in one or several classes. A point should be made that when web services are examined as a technology, the interface operation is called Operation Contract, and the interface – Service Contract.

UML allows for a more compact representation of the service interface and the connection with the class that gives details about it (fig. 1c). In a number of cases this is more convenient but is also less readable and involves the use of CASE environments.

---

3 http://www.webopedia.com/TERM/W/Web_services.html (September, 2012)
In many cases the preferred way of representing web services in CIS is as components (Web service component). The component presents a logical, easily replaceable part of the system that uses or provides a set of interfaces. This organization allows for the well-built components with precisely defined interfaces to be seamlessly replaced by new but compatible ones. In other words, the component is closely committed to the interface; on the one hand, it can provide interface and on the other hand – require interface (fig. 2). This is particularly convenient for representing interdependable services, of the kind used in building service-based systems.

Web services classes and their components are interrelated (fig. 3). To be more precise, classes carry out the component, that is, they are encapsulated in it and support the corresponding interfaces. Fig. 3 shows that the interface is the same for a class and a component. It is possible for the web service class to be represented as part of the corresponding component which carries out the interface by means of dependency-delegation type of relationship (fig. 3b).
2. Web services communication

A web service is used by its clients. The clients of the service use its functions; these may be various kinds of applications – e.g. a graphics application, a web-application, RIA, another service, system, etc. In some publications the web service is referred to as a service provider and the client – as a service requestor. Clients and web services communicate by sending and receiving messages. They interact directly or use an intermediary broker. Obviously, during the message exchange the web service interface is used.

In our view, the interaction between a web service and its clients can be most feasibly represented by a diagram of sequence or an UML communication diagram (fig. 4). These are behaviour describing diagrams representing the dynamic aspects

---

Fig. 3. Interrelation between a web service class and component

Fig. 4. Communication diagrams for representing a client – web service interaction

---

of the objects studied, which makes them suitable for an adequate and concise description of web services elements and the dynamics of their usage. Classes (fig. 4a) or components (fig. 4b) can be used as elements of these diagrams, defining the clients and the web services interfaces. In this case the exchanged messages appear as requesting operations on the web service interface, or on its class, respectively. It is possible that instead of classes or components their specimens are used; this facilitates a more precise representation of the dynamics of using Web services.

As it was mentioned above, a client of a web service can be another web service. Communication between those is the foundation of the building and functioning of service-based systems (fig. 5).

It is important to point out that service-based systems consist of inside/local and outside services. They can be provided by different developers, implemented through different technologies, programming languages and platforms, may have different versions, may be executed at a different time, etc. After all, given this wide variety, the interfaces of web services must be compatible and support communications in accordance with pre-defined schemas. They have to enable interaction within the system among local services, as well as interaction outside the system with remote clients.

![Diagram of service-based CIS](image)

**Fig. 5. Service-based CIS**

3. Web services data exchange through formal parameters

Data exchanged in web services according to WSDL\(^6\) standard are called types – i.e. types of web service data, and according to the software platform WCF\(^7\) – data contracts. They are a constituent of the messages transferred, which is closely related to the web services operations, provided by their interfaces.

\(^6\) [http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl) (September, 2012)
Web services can be viewed as invoking operations between two separate classes, respectively components. This means that mechanisms used in object-oriented programming can be applied to data exchange in web services but some inherent limitations should be considered.

To add some detail, method data exchange in object-based programming uses the following basic mechanisms: of global variables; of formal parameters; of method-functions; of dynamic memory.

To find out which of the data exchange mechanisms mentioned are suitable for web services, we shall first study and define data transfers in web services informally and after that present the relevant mechanisms formally, using UML.

A web service and its clients are implemented by programs. Consequently, behind the services provided and used, we find conventionally developed object-oriented applications and the communications between them are carried out by XML-based web technologies. Transported data can belong to various types: integers and real-valued numbers, strings, arrays, objects, etc. which have a different representation in the operating memory for the client and for the service. Therefore they need to be converted to a kind that is technologically neutral to implementation, can be comprehended by any client of the service and meets the WDSL standard. A typical characteristic of web services is the fact that they do not marshal addresses, but variable values. This mechanism is known as marshalling by value.

Having in mind the specifics and constraints of web services data exchange, we can summarize that they do not allow for the use of mechanisms for data transport by dynamic memory and global variables as is the way with conventional programming. This task can be accomplished using formal parameters and method-functions, though at a physical level data is mostly converted to XML-format (which is a text format) and meaning is conveyed. To be more precise, data transmitted in web services can be represented through parameters and the type of result of their operations. Thus web services operations can be used not only for the execution of certain activities and behaviour but also for data exchange with the clients of these services.

Formal parameters used in web services are as follows; input, input-output, output. They are closely connected with the patterns/schemas for message exchange (MEP - Message Exchange Patterns) in Web services.

The input formal parameters are used to convey meaning from client to service. Therefore, with this mechanism for data transport, a message is sent to the service and no reply is expected. This matches the semantics of message exchange pattern in web services without a reply, i.e. fire-and-forget MEP. With this pattern the client only needs confirmation that the message has been successfully received, and not a reply from the web service. This shows that in order to better convey the meaning of the message, input parameters must match a service operation implemented as a logical function.

Input parameters can be represented in the definition of the class operation, the web service interface operation, or in invoking the service. To achieve this, it is good
to use class diagrams including web service classes, or components diagrams with components and interfaces of Web services. In our opinion it is better to use classes in this case (fig. 6a), and not components as it makes things clearer. Input parameters are marked with the key word `in`, which can be omitted.

![Web service class diagrams](image)

**Fig. 6. Using formal parameters and method-functions in Web services**

Input-output parameters are applied when they are needed to convey meaning towards the web service and a result of the execution of its operation returned. For this reason they must be variables. In context, the message exchange patterns which make use of these parameters are synchronous request-response MEP and asynchronous request-response MEP. These parameters are visualized analogically to the representations of input ones, but are marked with the key word `inout` (fig. 6b).

Output parameters are used for returning meaning after the corresponding web service operation is invoked. Like the input-output ones, they must be variables. They are marked with the key word `out`. In a sense, output mechanisms are similar to input-output ones, but they must not have initial meanings, that is, they are not initialized (fig. 6c). Their semantics does not exactly match any of the message exchange patterns, but we believe they can be used in defining synchronous and asynchronous requests – responses. Both patterns employ messages to transport data to the web service and to return the result of the execution. This means that if we use output formal mechanisms in messages, no data will be transported to the web service, but they enable the return of necessary data to the client of the service.

Method-functions are suitable to use when a result needs to be returned to the point of sending the message towards the web service (fig. 6d). They can be used along with various types of parameters and are applicable in all kinds of message exchange patterns in web services: synchronous request - response, asynchronous request - response, no response.

Types of data that can be transported in web services are*: primitive; objects; interrelated objects; collections of objects, etc.

*Primitive data types are supported by most programming languages such as C#, Java, Delphi, etc. Scalar data types char, bool, int, double, float, etc., string, countable type also belong to the primitive data category. These data types are usually supported.

in a hassle-free way by both web service consumers and providers, which means that no additional descriptions are necessary in transporting them.

Relaying and using primitive data types in web services is shown in fig. 7. It demonstrates the transfer and use of information of cross currency exchange rate of a currency pair, specified by its denotation – e.g. GBP/JPY, AUD/JPY, CHF/CAD. For this purpose, two communication diagrams are developed – with classes and with objects, using UML. In order to specialize UML for a more accurate representation of web services we have introduced some stereotypes\textsuperscript{10}. The first diagram shows at an abstract level the defining and using of the Web-service (fig. 7a), and the second (fig. 7b) shows how the service is used with the help of concrete data about consumers and providers. It is shown on fig. 7b that the operation argument is replaced by the corresponding currency pair, and the operation result – by a concrete currency ratio. This model of data exchange between client and service uses method-function with input parameters.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig7.png}
\caption{Representing web service primitive type of data exchange}
\end{figure}

We can make the conclusion that standard UML tools are suitable and sufficient to represent the transport and use of primitive data types in web services. Thus web services basic model of interaction (see fig. 4) can be rendered more detailed and precise. Here we cannot identify any peculiarities that require defining and using additional UML mechanisms, apart from the stereotypes introduced.

\textsuperscript{10} A stereotype is a new element of UML, which is introduced in developing specific systems models. The introduced stereotype is noted as follows: \texttt{<<name of stereotype>>}
4. Mechanisms of transporting objects

Transferring of objects and collections of objects in web services is more complex, compared to the transferring of primitive data types. In this case the class, which is the basis of the transported objects and which the web service depends on, must be also accessible for the clients of this service.

Objects are created based on the classes that define their structure. They are composite, complex data structures. To describe these data structures in the context of web services developer-defined types of data are used, which we are going to call complex data types of web services. They are composed by data items. The object type definition is part of WSDL definition and XSD is used, which is a text format again.

To recapitulate, complex data types of web services match the object classes and data items match their fields or properties. To present them, therefore, we can use class diagrams and components diagrams.

In defining it, a Web service is designated in a space of names, thus making it visible for its consumers. To distinguish it from the space used in conventional programming we shall name it Web service namespace.

The defined complex data structures for object representation in web services may share the same space with the service, but it may have its own dedicated space, which has to be accessible to the service. Namespaces are used for grouping together web services and complex data structures related to them. They can, therefore, be represented as UML packages, and the interdependencies between them - as relationships between packages.

The interdependencies between web services, complex data structures and namespaces are shown in fig. 8. Web services and data types can be located in one namespace (fig. 8a) or in different but related namespaces (fig. 8b). The latter shows that only the web service interface is visible outside its namespace and its class is private; fig. 8b also shows that complex data types are public and are used by both the web service and its clients.
The question of the representation of complex data types at a component level is debatable, in our opinion. There certainly are some issues that have not been clarified. On the one hand these stem from the meaning the UML standard gives to the elements of interface and port, while, on the other hand, there are not enough examples of their usage. Specifically, access to the component function is usually carried out by means of the interfaces this component provides, yet they cannot be used for exporting data types. For this purpose, we believe the components ports are a feasible choice.

In the specialized literature\textsuperscript{12} a port has two functions. The first one is for the port to connect with the interface. More accurately, a port is defined as a point of interaction with a well-defined interface. Each port has a set of provided and requested interfaces, which define its interaction with the outside environment. In its second function the port is linked with a property or attribute of the component. We shall use the second interpretation of the port, but applied in the web-services context. More precisely, the port can be linked to a complex data type provided by the web service.

\textsuperscript{12} Unified Modeling Language: Superstructure (Ver. 2. 1. 1). Object Management Group, 2007.
On the one hand, the data type is public, and on the other – it can be used by the web service interface, which accurately describes its function (fig. 9).

**Fig. 9. Defining and using a web service data types at a component level**

A typical example of the web service use of objects is an online book of contacts or address book (fig. 10). This is a service provided by any type of component business.

**Fig. 10. Using objects in the web service Book of Contacts**

Stereotypes used:
- WSNS (Web Service Namespace)
- WSCDT (Web Service Complex Data Type)
- WSOp (Web Service Operation)
- WSDI (Web Service Data Item) a)
software, Windows operating system, email clients (Windows Live Mail, MS Outlook, etc.), MS Office, Internet platforms like Google, Windows Live, etc.

The Book of contacts web service provides essential functions for the management of particular contacts such as adding a contact, changing a contact, deleting a contact. Hence, data stored for each contact are name, family name, email, telephone, address, etc. The Contact type objects are assigned as parameters to the corresponding web service operations. In order to correctly interpret the diagram in web services context the necessary stereotypes are added to its elements.

* * *

The use of web services is a major trend in the development of modern CIS. It is widely accepted that the web services orientation of CIS is a best practice and an innovative technology in the building of those systems. Web services are a sophisticated, complex and heterogeneous technology. This calls for representing their functions at a high level of abstraction and in our opinion the UML standard is suitable for the purpose. Thus conditions are created for an easier comprehension of web services by a wide range of specialists. In relation to this, our article attempts to study and systematize essential capabilities of web services, having established that they possess two basic forms of representation – classes and components, which are closely related. The basic mechanism of communication among web services is studied. This mechanism has been developed and supplemented by data exchange mechanisms between a client and a web service. The close relation is shown between data exchange mechanisms and message exchange patterns in web services. Finally, by means of concrete models the use of complex data structures (objects) of web services is visualized. Parallel to that we have defined some stereotypes which facilitate the application of the UML method in modeling service-based CIS. Outside the range of this study remain some issues concerning the exchange of hierarchically organized objects, aggregates and collections of objects which we plan to study and present in our next publications.

In conclusion we can summarize the enabling role of UML for a clear, concise and unambiguous representation of the basic elements, mechanisms, and exchangeable data which is necessary for the building and documenting of service-based systems. We believe this paper can prove useful for both building service-based CIS and for the acquisition and development of web services concepts and technologies.

SOME ISSUES OF THE UML PRESENTATION OF DATA EXCHANGE MECHANISMS WITH WEB SERVICES

Assoc. Prof. Dr Filcho Filipov

Abstract

It is accepted that the orientation of computer information systems towards web services is a „best practice“ and an innovative technology in their development. Along with that, web services are a complicated, complex and heterogeneous technology. This requires that their
functions are presented at a high level of abstraction, and for that purpose, according to us, it is appropriate to use the UML standard – thus there are created conditions for them to be more easily adopted by a wide circle of specialists.

With regard to this, in this publication, having systematized the main capabilities and elements of web services, there are offered solutions for their presentation by means of UML. It has been determined that they can be presented in two main forms – as classes and components, which are closely connected with each other. There is studied the main mechanism of communication between web services, and it is further developed and supplemented with the mechanisms for data exchange between the customer and the web service through the parameters of its operations. By means of specific models there is illustrated the use of complex data structures with web services.