USABILITY ISSUES IN SERVICE-ORIENTED ARCHITECTURE

Jaroslav Král and Michal Žemlička
Department of Software Engineering, Faculty of Mathematics and Physics, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Keywords: Service orientation, usability in SOA systems, user involvement, autonomous service, user-oriented component interface, software confederations, agile development of large systems.

Abstract: Usability is of a growing importance. It is crucial for the acceptance of software systems nowadays. Software usability in its classical sense is mainly the property of the user interface of a system. Usable interface should have at least three properties: it must be easily understood and remembered and not too laborious in use. We show that in SOA systems called confederation the first two properties should have the interfaces of constituent application services. It is a precondition for the usability of user system interface. The properties are crucially important for the software engineering aspects of confederations (scalability, modifiability, reuse of existing systems, stability) as well as for their functions, e.g. for business processes (flexibility, on-line modifiability, etc.). We discuss some standardization issues.

1 INTRODUCTION

Service orientation is the hot topic and crucial issue of contemporary software engineering. There is no agreement on the definition of service orientation, compare (MacKenzie et al., 2006; W3 Consortium, 2002). In fact, under service orientation quite different systems are understood. Based on our several decades experience with SOA-like software flexible manufacturing systems (Král and Žemlička, 2005) we will discuss a more specific class of SO systems called confederations.

Confederations are systems having service oriented architecture (SOA) in which the constituent services ”know each other”. It follows that partner services need not be looked for at the stat of the communication between the partners. The world-wide standards need not be applied. It opens the possibility to use proprietary communication protocols or proprietary (e.g. user oriented) message formats. It as obvious drawbacks, but due current state of art of the development of SOA systems and due to the low maturity of some standards the advantages prevail. Confederations appear to be the systems having the most important impacts in practice.

Confederations are typical result of integration of legacy systems (e.g. information services of offices) into a new whole (software of e-government). They are typical for heath service systems, information systems of local authorities, ERP of global enterprises, etc. It is advantageous to develop confederation as a p2p network of services of two types (Král and Žemlička, 2005):

- *Application services*, often wrapped legacy systems of third party products, providing the basis business capabilities (functions),
- *Architecture services* enabling to build system architecture. The services are used to enhance middleware functions supporting the collaboration of application services (filtering, routing, data stores, etc.) or as a service orchestration tools. The usability of service interfaces can be often enhance by architecture services called *front-end gates* or *proxies* (Král and Žemlička, 2002; Král and Žemlička, 2005) staying logically between application service and middleware and transforming e.g. fine-grained implementation-oriented messages into coarse-grained ones usable by users as well as communication partners.

We will show that the functionality of confederations crucially depends on the property which can be

Král J. and Žemlička M. (2007).
USABILITY ISSUES IN SERVICE-ORIENTED ARCHITECTURE.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 482-485
DOI: 10.5220/0002387704820485

viewed as a generalization of usability for classical (monolithic) systems (Nielsen, 1993). The usability of the interfaces of application services has crucial consequences for the system.

Usability is of growing importance as it is the kernel of trends to consumerisation (Morello, 2005) of software. Consumerized software should be very simple for use (usable), adaptable to user needs, **and** reliable. Usability of software is usually understood to be mainly the property of client layer of modern systems. In confederations the classical usability issues are solved in portals. The usability in this sense has only a little influence on the quality of system capabilities and properties of the functions provided by the system. It is not the case in confederations.

2 CONFEDERATIONS AND USABILITY

As service-oriented architecture (SOA) we understand the architecture of the systems consisting of a collection of components behaving like the real-world services; i.e. the components are:

- autonomous and permanently active providing some capability to service consumers;
- allowing to have more than one unsettled service request at a time; there can be in principle no limitations on service consumers exits;

It is not difficult to see that SOA in our sense must have the properties of a virtual peer-to-per network N of services. The basic service communication mode in N in asynchronous. The software components in SOA called in the sequel application services provide collections of capabilities (atomic services). The atomic services are invoked by service requests. The application services mutually communicate. Two cases arise: (i) the communication partners of the application service are known to the service developers, (ii) the developers do not know the communication partners of the developed service. The first case we call *software confederation*, the second *alliance*.

The confederations are quite common. They are typically the result of the integration of existing applications (usually existing information systems) by attaching connectors to the inerconnected applications. The connectors connect the applications to a middleware. A confederation usually contains user interfaces and architecture services having specific roles. Confederations are often the result of the integration of legacy systems like information systems of offices to provide software of e-government, collaboration of departments/divisions of an enterprise, the system supporting collaboration of a health care institution, or cooperation of companies within supply chain or in specific coalitions. Note that in all these examples we must distinguish what components (services) form the system and what are their interfaces.

Confederations are the way of reuse software systems by a smooth integration which is now an ultimate request to save software investments and even to have such systems like *e*-government in reasonable terms. The solution usually fails if the application services are not properly designed, it is, if they do not mirror the real-world services.

3 ENGINEERING ADVANTAGES OF SERVICE USABILITY

Usable systems having application services with usable interfaces[1] have some important software engineering advantages:

Stability in time: As user-oriented interfaces mimic the interfaces of real-world services being used for long time, it is a good chance that they will not be changed in the future. They will be well understandable for all communication parties. It is especially true for operational information systems.

Agile development and use: User-oriented interfaces must be developed in tight cooperation with users. Other principles of agile development must be used too. It therefore opens the way of the application of agile philosophy in the development of large systems.

Easy insourcing and outsourcing: User-oriented interfaces are advantageous for insourcing and outsourcing as technically insourcing and outsourcing must be implemented by message redirection. The same turn can be used for screen prototyping of non-existing services. In this case the messages are redirected to portals.

Incremental development: This turn is very useful in the case of incremental development. It enables debugging allowing user involvement.

Agile business processes: Confederations enable such an implementation of business processes allowing business agility.

[1]Service interfaces are usable only if they reflect user domains' knowledge and languages. As such they must be usually declarative and coarse grained. We say that such interfaces are *user-oriented*.

4 BUSINESS PROCESSES

One of the crucial requirements on information systems is the ability of information systems to support business processes. The implementation must take into account that the processes must in their individual steps use capabilities of basic (application) processes and coordinate (orchestrate) their tasks.

Business processes are of various complexity: some are just a short linear list of simple tasks, some complex networks of such activities. They often evolve in time. They must immediately react on changes in the surrounding environment – e.g. changes in law, changes in cooperating companies, or changes in market demand or in the availability of resources needed for it (see (Král and Žemlička, 2005)). This requirement of immediate adaptation of business processes to the changes of environment is a difficult challenge for software developers.

There should always be someone (process owner) responsible for the business consequences of the process. No process owner will agree with the responsibility unless he/she can certificate/commit the process steps and have the possibility to modify the process if necessary. It is process owners must be able to supervise and control the process and modify it online. It includes its specification, creation, and also modification during the process run. It implies that the business processes cannot be fixed in the software, they must be specified by users and use specific process control data (process model).

The supervision by the process owners implies in fact that basic (atomic) services providing atomic steps of the business processes must have user-oriented interface as otherwise the quick modifications of the process would be impossible. In this case it is necessary to involve programmers transforming the non-user oriented interfaces into the user-oriented ones. It is inflexible, expensive, and time consuming.

User orientation of the interfaces implies that the messages are declarative (i.e. saying rather what to do that how to do it) and therefore not procedural (like SOAP). Problem is the optimal standardization policy for user-oriented message formats (see below the details of this topic).

The requirements on business processes support by modern software (possibility to supervise, user-modifiability and definability, extensibility by newly defined and encoded trivial activities, maximal robustness) are (or should be) crucial for the software architecture of software implementing business processes. The possible way how to implement business processes to have the desired properties can be based on the generation of a new service called process manager (peer) for every new business process, see (Král and Žemlička, 2005) for details.

5 USABILITY AND STANDARDS

The most important objection against user-oriented interfaces of services is based on the fact that at least nowadays and in the near future there is only a very little chance that such an interface is fully standardized. It is correct but if we want to use current standards based on universal standards we have to give up the idea that we can use declarative (and often coarse grained as well) interfaces and the advantages dependent on it.

The use of standards like SOAP implies that we must use low-level programming concepts like remote procedure calls. The reward is that there are tools supporting the use of SOAP as well as such tools like WSDL allowing to obtain the definition of the interface in executable form and a tool (not too successful) allowing to find an appropriate service. The tools are necessary in *e*-commerce where communication partner must in principle be looked for.

Note that the hurried standardization of the interfaces could lead (and usually leads) to premature standards. It will not be the case if the standards are based on the interface formats that have been used for some time. So it seems to be bearable to sacrifice standardization (i.e. standardized XML dialects and associated tools as well) for some time and to some extend. Using modern XML-related tools like XSLT the sacrifice need not be absolute.

The user-oriented interfaces strongly tend to be coarse grained. It to some degree contradicts the recommendations from SOA Open Reference Model (MacKenzie et al., 2006) or at least indicates that there is a broad class of very important service-oriented systems for which is the use of coarse-grained interfaces essential. The recommendation of the Model is therefore in this respect from the pragmatic point of view a bit misleading.

6 INFORMATION SYSTEMS INTEGRATION

Current trends in economy towards integration and globalization enforce the necessity to integrate organizations and their information systems. Integration of information systems can be done as follows:

1. replacing the individual systems by new ones;

2. integration the services of the individual subsystems into business processes (typical for insourcing or for purchasing of new organizational units, integration of newly developed components or third party products, system integration);
3. removing a service from the system (outsourcing, selling out an organizational unit).

The first type of integration is possible only when the developers of the new system can have access to all necessary information and when individual systems are of similar type and about at the same security level. It is applicable in the case when individual information systems of one organization are integrated – if the subsystems are information systems and not control systems.

The second type seems to be the only solution in the case when heterogeneous systems are to be integrated or if the legacy systems are based on some knowledge that is currently inaccessible. This type of integration is usually achievable at lower costs and in shorter time for its implementation than for the redevelopment of the entire system from scratch. The resulting system is usually a confederation.

7 CONCLUSIONS

The ability to integrate and reuse existing applications is probably the most important property and crucial effect of service orientation. It allows saving of redevelopment investments but also investments into staff (user) training and related expenses (DeForest and Rosenbloom, 2005). The use of legacy systems is not without problems – especially in the cases when there are no good user-oriented interfaces of the legacy systems.

If we are able to use good user interfaces, we gain many advantages of technical as well as managerial nature. The advantages are so important that it quite frequently makes sense not to use some world-wide standards.

The experience with the use of user-oriented interfaces makes it possible to define standards for them and to find a proper balance between fully standardized and proprietary solutions.

We did not discuss the case when the system has a heterogeneous architecture, e.g. some parts are confederative, some are batch ones and in other parts the services are web services. Such systems can be integrated using a generalization of data stores (Král and Žemlička, 2005) known from structured development (Yourdon, 1988).

We believe that much research in service-orientation is overly oriented towards full computerization. We believe that such systems must take people (users) as an integral part of the system with their abilities as well as disabilities. It is also the case for such systems like web services in semantic web.

An open problem is the balance between decentralization and centralization of services. The important of this topic is clearly visible on the history of UDDI (W3 Consortium, 2001; UDDI Initiative, 2003).

ACKNOWLEDGEMENTS

This research was partially supported by the Program ”Information Society” under project 1ET100300517.

REFERENCES

DeForest, B. and Rosenbloom, S. (2005). SOA: The perennial legacy issues. *Business Integration Journal*.

Král, J. and Žemlička, M. (2002). Autonomous components. In Hamza, M. H., editor, *Applied Informatics*, pages 125–130, Anaheim. ACTA Press.

Král, J. and Žemlička, M. (2005). Implementation of business processes in service-oriented systems. In *Proceedings of 2005 IEEE International Conference on Services Computing*, volume II, pages 115–122, Los Alamitos, CA, USA. IEEE Computer Society.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., and Metz, R. (2006). Reference model for service-oriented architecture 1.0, commitee specification 1, 19 july 2006. http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf.

Morello, D. (2005). The IT professional outlook: Where will we go from here?

Nielsen, J. (1993). *Usability Engineering*. Academic Press, New York.

UDDI Initiative (2002–2003). Universal definition, discovery, and integration, version 3. An industrial initiative, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3.

W3 Consortium (2001). Web service definition language. http://www.w3.org/TR/wsdl.

W3 Consortium (2002). Web services activity. http://www.w3.org/2002/ws/.

Yourdon, E. (1988). *Modern Structured Analysis*. Prentice-Hall, 2nd edition.