CLOSING THE BUSINESS-APPLICATION GAP IN SOA
Challenges and Solution Directions

Boris Shishkov
Department of Computer Science, University of Twente, Enschede, The Netherlands

Jan L.G. Dietz
Department of Software Technology, Delft university of Technology, Delft, The Netherlands

Marten van Sinderen
Department of Computer Science, University of Twente, Enschede, The Netherlands

Keywords: Business-software alignment; Software services; Service-Oriented Architecture - SOA.

Abstract: Adequately resolving the business-software gap in the context of SOA (Service-Oriented Architecture) appears to be a non-trivial task, mainly because of the dual essence of the service concept: (i) services are inevitably business-restricted because they operate in real-life environments; (ii) services are also technology-restricted because the software components realizing them have to obey the restrictions of their complex technology-driven environments. Hence, the existence of these two restriction directions makes the (SOA-driven) business-software alignment challenging – here current business-software mapping mechanisms can only play a limited role. With regard to this, the contribution of the current paper is two-fold: 1. it analyzes SOA and its actual challenges, from a business-software-alignment perspective, deriving essential SOA application desirable properties; 2. it proposes software services specification directions, particularly concerning the (SOA-driven) business-software mapping. This contribution is expected to be useful in current software development.

1 INTRODUCTION

Closing the semantic gap between business logic and application logic, is claimed to be of crucial importance for an adequate software development (Shishkov et al., 2006*a*). We argue that this endeavor fundamentally concerns the underlying technology used and especially the restrictions it imposes on the application. Hence, we focus on a particular technology in this paper, namely the Web Service Technology that can be seen as an implementation of the *Service-Oriented Architecture – SOA* (Pasley, 2005). SOA provides a structure for composing software applications based on the use of 'services' as building blocks that can perform distinct functions through well defined interfaces (Alonso et al., 2004). These *services* are self-describing and platform-independent, and as such they are actually considered as fundamental computational elements in the composition of business processes; this makes them advantageously usable in developing software solutions (that concern the business system to be supported), as concluded in previous work (Shishkov et al., 2006*b*).

We can consider SOA as a conceptual business architecture where business/application functionality is made available to users, as shared, re-usable services (modules of business/application functionality) on a network (Marks & Bell 2006), invokeable by messages through exposed interfaces.

Hence, the ability of finding, selecting and composing services without prior agreement results in a 'loose coupling', which, together with the availability of service standards, bring potential benefits of increased software re-use, easier application integration and higher business agility (Caceres et al., 2004).

Shishkov B., L.G. Dietz J. and van Sinderen M. (2007).
CLOSING THE BUSINESS-APPLICATION GAP IN SOA - Challenges and Solution Directions.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 333-336
DOI: 10.5220/0001326803330336

To fully benefit from this potential, it is necessary that the specification of software services (in the SOA context) is properly integrated in a software design approach where the business-software mapping is systematically elaborated. This points to the above mentioned semantic gap which (obviously) needs to find resolution in the SOA context. Nevertheless, this appears to be a non-trivial task, mainly because of the dual essence of the service concept: (i) services are inevitably business--restricted because they operate in real-life environments; (ii) services are also technology-restricted because the software components realizing them have to obey the restrictions of their complex technology-driven environments. Hence, the existence of these two restriction directions makes the (SOA-driven) business-software alignment challenging; as a comparison, introducing some software systems can enforce changes in corresponding business-level processes.

Therefore, current business-software mapping mechanisms can only play a limited role in the specification of software services, which appears to be an actual and widely recognized problem.

With respect to this, the contribution of the current paper is two-fold:

- it analyzes SOA and its actual challenges, from a business-software-alignment perspective, deriving essential SOA application desirable properties;
- it proposes software services specification directions, particularly concerning the (SOA-driven) business-software mapping.

The paper is thus organized as follows: Section 2 – analysis; Section 3 – proposal; Section 4 - conclusions.

2 SOFTWARE COMPONENTS AND SOA

The emergence of *Service Oriented Computing - SOC* is considered as a move towards combining real-life business concerns and technological concerns, envisioning a service (of a component/entity) as defining the goal, capabilities and/or behavior (of the component/entity) as observed by and relevant to the users (of the component/entity) (Pasley, 2005).

A *web service* (WS, for short) is considered as a self-contained, Internet-enabled service component capable not only of performing business activities on its own but also possessing the ability to engage other WS in order to form higher-order business transactions (World Wide Web Consortium, 2005).

We distinguish between *composite* and *constituent* WS – a composite WS consists of (is provided by an orchestration of) multiple constituent WS, and a constituent WS is an 'elementary' WS, i.e. a WS which can be used on its own or in a composite WS (Newcomer, 2002; Papazoglou & Kratz, 2006).

In order to be usable on a large scale, WS (which are based on specific sets of standards) should be somehow reflectable in certain abstractions, as an instrument for their application in any platform through which the Internet user accesses them. Moreover, WS usually should not require design 'from scratch' because this would make them expensive. They should instead be re-usable, using one WS as a basis for developing another, by making use of its core functionality (Bosworth, 2001; Papazoglou & Kratz, 2006).

We thus consider it innovative that multiple users are able to access WS, personalize them and finally use them. Our first conclusion is that **this usage of WS implies advanced infrastructures and application platforms that utilize and coordinate such (globally) re-usable services**. Furthermore, employing such generic WS for work in domain-specific business environments means that the service use has to be driven by appropriate underlying business models.

Prior to their use, WS should have been discovered (by matching requirements to advertised names) and subjected to negotiation (since the user must of course accept using a particular WS).

All these considerations have contributed to the emergence of the Service-Oriented Architecture – SOA which goes beyond the sole consideration of WS (Alonso et al., 2004), being a useful paradigm that can support engineers in their designing, building and using distributed software systems. SOA facilitates the establishment of ICT support for business processes, which is readily available, flexible and easily maintainable across multiple organizations and platforms. The concept of service/WS adopted by SOA, has evolved from modular object/component middleware approaches, such as CORBA, DCOM and J2EE (Newcomer, 2002). However, WS has become the technology of choice for implementing service-oriented software systems, primarily because they are based on ubiquitous Internet standards, such as HTTP and XML, and because they support 'loose coupling'. Whereas the uptake of WS based SOA is impressive, there are still important fundamental challenges not addressed by this technology, as recognized by ACT4SOC (ACT4SOC, 2007).

Firstly, the 'plug and play' interoperability of WS to enable *ad hoc* cooperation of new partners is limited (Wang & Zhang, 2006). For on-demand composition of services in an open service-oriented world, interoperability has to be ensured at different levels (syntactic and semantic) and in different dimensions (information and behavior). Current research in this direction is using, for example, Semantic Web and ontology technologies (World Wide Web Consortium, 2005).

Secondly, the property of 'loose coupling' is not appropriate for many applications that involve stateful components. Hence, the benefits of Web services and SOA would be limited for developers of such applications if they themselves have to solve the issues of stateful interaction, notification of state changes, support for sharing and coordination (Shishkov et al., 2006*b*). It should thus be aimed that these concerns are placed at the service infrastructure level or that another solution is enforced. Thus, our second conclusion is that **enhancement needs to be achieved in the way applications which are by nature not loosely coupled, are supported by SOA-related technology**.

Finally, WS alone are insufficiently capable of guaranteeing an appropriate 'alignment' between business requirements and software functionality, as mentioned already. What is needed is a structured approach for developing service-oriented software solutions, in which consistency with business requirements, (de-)composition of application services, and mapping onto (alternative) technology platforms can be systematically and separately addressed (Alonso et al., 2004, Shishkov et al., 2006*b*). Hence, our third conclusion is that a **business-software alignment is needed particularly in the SOA context**.

Taking into account the 3 business-software-alignment-related conclusions made in the current section, we formulate the following 3 (corresponding) desirable properties concerning the SOA-driven application development:

1. Application architecture must allow usage of SOA infrastructure;
2. 'Loose coupling' should be enforced;
3. Application architecture must fit within the business context.

We will consider in the following section, several solution directions that are driven by these properties and especially by the third one.

3 SOLUTION DIRECTIONS

In presenting solution directions in this section, we will consider fundamentally an *Alignment perspective* and a *Loose-coupling perspective*.

The **Alignment perspective** basically concerns desirable properties 3 and 1, as defined in Section 2. The existence of their related restriction directions has already been mentioned in the Introduction. We claim that in order to adequately address them, it is necessary that they are distinguished clearly.

As for *business-level restrictions*, they obviously concern the enterprise that is going to be supported by the services-to-be-developed. Thus, the enterprise under consideration needs to be sufficiently explored, which can be appropriately accomplished through a *conceptual enterprise model*, as studied by Dietz (2006). This model nevertheless would need to be *coherent* (the distinguished aspect models constitute a logical and truly integral whole), *comprehensive* (all relevant issues are covered), *consistent* (the aspect models are free from contradictions or irregularities), *concise* (no superfluous matters are considered) as well as *essential* – revealing only the essence of the enterprise, its deep structure. Considering this last property as crucially important, we argue that meeting the business-level restrictions, as above formulated, should include a (SOA) application specification that is realized as a REFINEMENT OF A CORRESPONDING ESSENTIAL ENTERPRISE MODEL. Such a model could be properly developed using LAP-driven generic business process patterns, as studied in (Shishkov et al., 2006*a*).

As for *technology-level restrictions*, they could in no way concern essential issues as formulated above. Instead, they concern *realization* and/or *implementation* issues which are to stay consistent with corresponding essential issues (Dietz, 2006). Nonetheless, after mapping technology-level restrictions on the essential enterprise model, it might appear that a model re-design seems sensible – this is an example of an *indirect impact* of technology-level restrictions on the enterprise model. Apart from this, technology-level restrictions concern the application's integration with respect to the service infrastructure – the APPLICATION IS TO ALWAYS FUNCTION CONSISTENTLY WITH THE (SERVICE) INFRASTRUCTURE THAT PROVIDES TO IT GENERIC (RE-USABLE) SERVICES. Such a consistency can be enforced through a network-infrastructure-application-layered software development, as studied in AWARENESS (2007).

The **Loose-coupling perspective** concerns desirable property 2, as defined in Section 2. If straightforwardly reflecting an enterprise model in a SOA-driven application model, the services (which would be identified) would inevitably be tightly coupled because (normally) there is a dependency of the services provided by one entity on services provided by other entities. As concluded in (Shishkov et al., 2006*b*), this could be resolved, by introducing 'in between' an ADDITIONAL APPLICATION COMPONENT (labelled 'Orchestrator') THAT HAS THE TASK OF COORDINATION. The Orchestrator is application-specific (as the coordination is application-specific). The (subordinate) services, however, which are coordinated by the Orchestrator, may be useful for many different types of applications. Their description may hence be published through a public (or corporate) registry, such that they can be discovered, and selected for invocation by an orchestration component. Related to its coordination tasks, the Orchestrator could sometimes supply to one service the result of another service, if this is necessary for the service to perform its task(s).

4 CONCLUSIONS

This paper proposes improvements to the business-application alignment, particularly in the context of SOA. Reporting work-in-progress, the paper has only identified (relevant) application desirable properties and corresponding solution directions. According to the first solution direction, the SOA application model must be developed as a refinement of a corresponding essential enterprise model. According to the second solution direction, the application functionality must be specified consistent with the (service) infrastructure. These two solution directions are relevant (in combination) to the objective of overcoming (especially in the context of SOA) the semantic gap between business logic and application logic. As for the third solution direction, it mainly concerns the usage of generic services, which characterizes SOA. In particular, it is suggested that an additional application component is introduced to handle the (application-specific) coordination activities with respect to (subordinate) services, in delivering application's functionality.

We expect that this paper and the discussion it opens would be useful to the on-going SOA developments aiming at putting SOC on a more solid (theoretical) background.

ACKNOWLEDGEMENTS

This work is part of the Freeband A-MUSE project (http://a-muse.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

REFERENCES

ACT4SOC 2007: *Proc. of the First Int. Workshop on Architectures, Concepts and Technologies for Service Oriented Computing – ACT4SOC*, INSTICC Press.

AWARENESS 2007: Dutch project context AWARE mobile NEtworks and ServiceS, Gvrnm. supported.

Alonso, G., F. Casati, H. Kuno, V. Machiraju, 2004. *Web services, concepts, architectures and applications*, Springer-Verlag. Berlin Heidelberg.

Bosworth, A., 2001. Developing Web Services. In Proc: *17th International Conference on Data Engineering*.

Caceres, P., Marcos, E., De Castro, V., 2004. Integrating agile and model-driven practices in a methodological framework for the Web information systems development. In *ICEIS'04, 6th Int. Conference on Enterprise Information Systems*. INSTICC Press.

Dietz, J.L.G., 2006. *Enterprise ontology*, Springer-Verlag Berlin Heidelberg.

Marks, E.A., Bell, M., 2006. *Service-Oriented Architecture, A Planning and Implementation Guide for Businesses and Technology*, John Wiley & Sons Inc., Hoboken, New Jersey.

Newcomer, E., 2002. *Understanding Web services, XML, WSDL, SOAP and UDDI*, Addison-Wesley. Boston.

Papazoglou, M.P., Kratz, B., 2006. A Business-Aware Web Services Transaction Model. In *ICSOC'06, International Conference on Service-Oriented Computing*. Springer Berlin/Heidelberg.

Pasley, J., 2005. How BPEL and SOA are changing Web Services development. *IEEE Internet Computing, Vol.9, Iss.3 (2005)*. IEEE Press.

Shishkov, B., Dietz, J.L.G., Liu, K., 2006. Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In *ICEIS'06, 8th Int. Conference on Enterprise Information Systems*. INSTICC Press.

Shishkov, B., Van Sinderen, M.J., Quartel, D., 2006. SOA-driven business-software alignment. In *ICEBE'06, IEEE International Conference on e-Business Engineering*. IEEE Press.

Wang, H., Zhang, H., 2006. Enabling enterprise resources reusability and interoperability through Web services. In *ICEBE'06, IEEE International Conference on e-Business Engineering*. IEEE Press.

World Wide Web Consortium, 2005. *Web Services Description Language 1.1*, W3C Note, http://www.w3.org/TR/wsdl.