LEGACY SYSTEMS ADAPTATION USING THE SERVICE ORIENTED APPROACH

Francisco Javier Nieto, Iñigo Cañadas and Leire Bastida
European Software Institute, Parque Tecnológico #204, 48170 Zamudio, Bizkaia, Spain

Keywords: Legacy Systems, Service Oriented Architectures, Web Services, Methodology, Services Composition.

Abstract: Legacy systems are the core IT assets of the great majority of organisations that support their critical business processes. Integrating those existing legacy systems with the rest of IT infrastructure is a complex and difficult task. Legacy systems are often undocumented, inflexible and tightly coupled and imply high cost of maintenance. Many organisations are starting to look at Service Oriented Architectures (SOA) as a potential way to expose their existing legacy investment as functional units to be re-used and exploited externally. This paper is focused on providing guidance to those organisations which want to use SOA on legacy adaptation and transformation. For doing so, this paper defines a vision and a set of best practices that any organisation should follow in order to expose their useful legacy functionalities as part of a SOA environment, allowing the development of hybrid systems understood as compositions of new services as well as of legacy systems and existing components wrapped as services.

1 INTRODUCTION

Many organisations have mission-critical systems that have been running their business for years. These systems, where organisations have made a huge investment over the years, were built in order to manage complex business processes giving each company a unique competitive advantage. IDC (IDC, 2006) estimates that two-hundred billion lines of legacy code are in use today on more than ten thousand large mainframe sites, whilst the Hurwitz Group (Hurwitz, 2001) reckons that only 10% of companies have fully integrated their most mission-critical business processes. It is necessary to evaluate whether it is possible to leverage legacy assets, thus getting the most value from what already works while enabling technological and strategic evolution at the same time.

The challenge is to understand and face the issues which must be addressed in order to obtain a successful migration from existing legacy systems to a Service Oriented Architecture environment, enabling the interoperability between old systems and new ones.

Some of the problems that could arise during this adaptation and migration projects could be the lack of guidelines and standards for reusing legacy assets, lack of knowledge in SOA related technologies, programs with too many lines of code, data and communication mismatches and poor performance.

This paper provides a vision (section 2) and a methodological solution (section 3) which could solve or, at least, mitigate the problems related to the legacy systems adaptation to SOA environments.

2 VISION

This paper deals with the legacy adaptation problem from a functional point of view, rather than a data-centric one. There are many cases where it is important to enable the data access, but our vision is closer to the SOA concept, where services will represent functional units accessible through the network.

When thinking on how to adapt legacy software assets (from full systems to stand-alone components) to SOA environments, there are two options that can be envisaged for this adaptation.

- When there is a stand-alone component, the interfaces of the component can be wrapped, exposing that component as a service, with Web Services standards (SOAP (Gudgin, 2007) and WSDL (Christensen, 2001));
- When there are large legacy systems, such as an ERP (Enterprise Resource Planning), it could

Javier Nieto F., Cañadas I. and Bastida L. (2008).
LEGACY SYSTEMS ADAPTATION USING THE SERVICE ORIENTED APPROACH.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 330-333
DOI: 10.5220/0001712903300333

be interesting to split the legacy functionality into various services in order to offer reusable sub-functionalities that can be used in a separate way. But the company would be able to provide all the functionalities together, as it did before, adapting the offer to the client needs.

So, the main problem is to extract the functionality provided by the legacy software assets. This can be solved using two kinds of wrappers: basic wrappers and advanced wrappers, depending on the size of the legacy system.

2.1 Basic Wrapper for Stand-Alone Components

Apart from a syntactical description of the operations, the new service should deal with the behaviour, so the component can be integrated in an easy way and interaction errors can be avoided.

The proposed solution is to wrap components in several levels, such as in the approach presented in (Zemlicka, 2004). Several tiers are defined for the wrapper, each one covering one aspect of the adaptation of existing components and systems, and taking into account the behaviour modelling of the legacy components, but from a SOA point of view.
This way, the interface of the new service will be very stable, the behaviour wrapper can act as a mediator (managing access rights of the operations), the development and testing can be one using clearly defined processes, and a full SOA-compliant wrapper may be built, improving interoperability.

2.1.1 The Communication Wrapper (Tier 1)

It is the component which accesses directly to the component to be wrapped. It could be included as part of the existing system or as an external component acting as some kind of mediator, transforming SOA requests and responses to those communication mechanisms which are compatible with the legacy software.

This wrapper could be created manually by experts in SOA and the legacy software. Commercial vendors such as TIBCO, provide some adapters (for Siebel, Lotus Notes, COM, CORBA, etc.) and development kits to create more adapters.

2.1.2 The Behaviour Wrapper (Tier 2)

This part is in charge of the management of the behaviour of the legacy code. Its purpose is to control the operation calls managing asynchronous reception of data, data adaptation and calls to external components in a logical order, according to the legacy software behaviour. This is the real interface to the legacy system, allowing a standard communication with the functionality provided.

The idea behind its implementation is to apply SOA techniques and orchestration languages, such as WS-BPEL, which can manage the behaviour. Those operations which do not have behaviour constraints will be exposed and implemented just as an invocation to the wrapped component.

2.1.3 Web Service Interface and Specification (Tier 3)

At the top level, it is necessary some kind of interface description and specification. The standard for services description is WSDL, but it is focused in the syntactical description of the service, leaving apart semantic descriptions which support the understanding of the service purpose and functionality.
Other specifications could be used for providing that extra information,. Keeping separated this information and its publication will ease the maintenance.

2.2 Advanced Wrappers for Large Systems

This approach is focused on the Software as a Service (SaaS) paradigm, where large systems are split into smaller functional parts. But it is clear that those functional parts were combined as part of a higher level application, so there is no reason to loose the advantages of providing all the functionalities together, as in the original system.
In the case of large legacy systems, two more levels are added, for representing the original processes. This way, each functional part can be used not only in a separate way, but also in conjunction with other functional parts in a fully SOA-compliant way.

This way, the services may be grouped easily with standard languages, the interactions between the components of the legacy are clear, it is possible to host the overall wrapper and the services in different servers and the higher level service can be extended with more functionality, getting a richer system.

2.2.1 The Composition Wrapper (Tier 4)

This part is a set of combinations using the obtained services, according to the initial functionalities

which the large system was offering. Before splitting the system, there were several relationships between the different functional parts which were working together to obtain complex functionalities. This part of the wrapper implements those interactions in a SOA-based way (using orchestration languages), so the obtained services compositions will be able to maintain the original complex functionalities of the system. Other standards can be used for covering all the aspects of the interaction (such as security or transactions).

2.2.2 Composition Interface and Specification (Tier 5)

As in the basic wrappers, once a service is exposed, there is the need to describe its interface and other especial features (if possible). Again, this part of the wrapper will include a WSDL file for the syntactical description. Any other specifications may be used if available, for describing other aspects, such as the implicit semantics in the service.

3 METHOD FOR DEVELOPING THE WRAPPERS

For an organisation that has already adopted SOA, and which is ready to advance to the stage of leveraging its existing systems, the lack of guidelines and best practices frequently stalls its migration efforts.

Consequently, this method supports the migration of legacy systems by means of exposing those systems or parts of them as composable services to be integrated in the SOA environment.

3.1 Phases of the Methodology

This section describes all the phases and tasks which are part of the proposed methodology, focusing on the pieces of legacy software to be adapted and the divisions to be done in large systems, for obtaining smaller software components which could be combined.

An iterative lifecycle can be applied when carrying out the phases and activities, e.g. identifying some parts and operations to be wrapped during each iteration.

For developing the ***composition wrapper*** and the ***composition interface and specification***, it is strongly suggested to apply the methodology described in (Nieto, 2007), since it deals in deep with services composition development issues and offers good guidelines for combining several services.

3.1.1 Preliminary Analysis

At the beginning of the adaptation process, it is necessary to get all the available information (especially, about architecture and behaviour), identify those legacy software assets to be used and analyse the feasibility of the adaptation project.

It is important also to identify whether it becomes interesting to make available some business logic already implemented (because of emerging markets) or customers prefer to use some parts of large systems in a separate way.

The result will be the techniques to be applied (e.g. focus on data access, systems migration, systems re-engineering, etc.) and the information available about the legacy systems, with an estimated budget for adapting them.

3.1.2 Legacy Re-Engineering

This phase is focused on the in depth analysis of the legacy systems, identifying the concrete services to expose and how to do it.

There is a functional analysis, identifying the interfaces (operations) to be provided and the components which implement the expected functionalities. Then, there is the analysis of the data related to the legacy system, for performing adaptations. And, finally, the analysis of wrappers integration in services compositions, for identifying additional requirements.

Finally, there will be activities for solving other aspects such any syntactic and semantic issues, and other non-functional issues (such as security and transactions needs), whenever necessary.

The outputs expected from these tasks will be the service requirements descriptions, the architectural design of the wrappers and mappings between legacy assets and functionalities.

3.1.3 Implementation of the Wrappers

This phase addresses the development of the required wrappers, implementing message passing between the calling and the called objects, and redirecting method invocations to the actual component services.

Following the vision of the wrapping levels mentioned before, the steps to be followed are: to implement the communication wrapper, to implement the behaviour wrapper, to implement the

service interface and specification, and to prepare the service deployment.

These tasks will result in a set of deployed services (using wrappers) with their services descriptions and specifications compliant with SOA specifications (such as UDDI and WSDL).

3.1.4 Composition and Delivery of Services

This task is related with the compositions to be designed in order to re-group functionalities or generate new ones taking the newly exposed services (result of wrapping the legacy systems) as the inputs of the task.

During this task, designs and requirements coming from the service composition will be analysed, in order to understand how the new services affect the service composition and whether some changes are needed in the services or they are useful without any modification. This may be an iterative process, updating the composition design and the legacy wrappers in order to end up with a successful integration.

3.1.5 Verification and Validation of Services and Compositions

In this last phase of the methodology, some actions will be performed in order to check whether the adapted legacy system works as expected, fulfilling all the requirements. There is a lot of literature about testing, verification and validation techniques. In this case, it is suggested to follow the Verification & Validation process presented as part of the SeCSE Methodology (SeCSE, 2007) which fits very well with the purpose of this phase.

The basic actions are to perform unitary testing against developed components, as well as integration testing, putting especial attention on the service behaviour. Doing test cases and analyzing the results will be the key for getting a successful validation.

4 CONCLUSIONS

The vision provided for the wrappers in several tiers is very useful, since it allows dividing the wrappers in well differenced modules with clearly defined purposes. This idea enables the usage of development methods while it improves the flexibility, extensibility, adaptation, maintenance and errors identification and fixing in wrappers.

The proposed methodology supports and guides the adaptation process, but it is not too concrete, allowing the application of existing and future techniques and standards, encouraging re-use and specialization in each tier. Some examples would be techniques described in (Canfora, 2006) and those included in the SMART methodology (Lewis, 2005), which could complement the steps defined in this methodology.

Next steps to face in this topic will be to analyse in depth the best techniques and practices, in order to create more concrete guidelines, templates and to support the implementation. Moreover, we will focus on validating the methodology, by applying it in real situations which are expected to arise in SeCSE project.

REFERENCES

Canfora, G., Fasolino, A. R., Frattolillo, G., Tramontana, P. (2006). Migrating Interactive Legacy Systems To Web Services. In *csmr, Conference on Software Maintenance and Reengineering (CSMR'06)*, pp. 24-36. Los Alamitos: IEEE Computer Society Press.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. (2001) *Web Services Definition Language (WSDL) 1.1*, W3C Note. http://www.w3.org/tr/wsdl

Gudgin, M., Hadley, M. Mendelsohn, N., Moreau, J.-J., Nielsen, H.F., Karmarkar, A., Lafon, Y. (2007) *Simple Object Access Protocol (SOAP) 1.2*, W3C Recommendation. http://www.w3.org/TR/soap12

Hurwitz Group (2001). *e-Business Process Integration (e-BPI) Study*.

International Data Corporation (IDC). (2007). *IDC 2007 Software Taxonomy*. http://www.idc.com/2007st/index.html

Lewis, G., Morris, E., O'Brien, L., Smith, D., Wrage, L. (2005) *SMART: The Service-Oriented Migration and Reuse Technique*. Technical Note CMU/SEI-2005-TN-029. Software Engineering Institute (SEI): http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn029.pdf

Nieto, F.J.; Bastida, L.; Escalante, M.; & Gortazar, A. (2007). Development of Dynamic Composed Services based on the Context. In Doumeingts, G., Müller, J., Morel, G. & Vallespir, B. (Ed.), *"Enterprise Interoperability, New Challenges and Approaches"* (pp. 3-12). London: Springer

SeCSE Consortium. (2007). A5.D4 – SeCSE Methodology. *SeCSE Project.* http://secse.eng.it

Zemlicka, M., Kral, J. (2004). *"Legacy Systems and Web Services"*. Technical report KSI MFF UK No. 2004/1. Prague: Charles University. http://citeseer.ist.psu.edu/zemlicka04legacy.html