A Meta-Model for Migrating a Legacy Information System based on Procedural Software Architecture towards Service Oriented Architecture

Slim Jendoubi
National School of Informatics Sciences (ENSI)
University of Manouba, 2010
Tunisia

Jamil Dimassi
National School of Informatics Sciences (ENSI)
University of Manouba, 2010
Tunisia

Henda Ben Ghezala
National School of Informatics Sciences (ENSI)
University of Manouba, 2010
Tunisia

ABSTRACT
During several years, the computing environments became more complex and more heterogeneous because of the diversity of customer's needs and of the technological evolutions. The agreement between applications and different technologies became a critical activity. So, to face this problem of interoperability, companies often resort to the solutions of enterprise application integration (EAI).

In what concerns us, we consider the EAI as being at the same time the strategy and the process that companies allow to reach a solution of optimal integration of the various heterogeneous, autonomous and distributed applications.

In this context, this paper consists on presenting a generic meta-model for integrating legacy applications with service oriented architecture. Indeed, to resolve the problem of interoperability between applications, this meta-model aims at capitalizing on procedural legacy applications and their integration into modern environments.

Keywords
procedural information system; legacy application; interoperability; software architecture; management process; SOA.

1. INTRODUCTION
During the last three decades, a significant number of software based on procedural paradigm was developed. Nowadays, these developments constitute a big part of the application heritage of companies. Indeed, this paradigm is allowed to release itself from the code machine and to strengthen a separation between the data representations and their processing. However, the development of complex systems using this paradigm was generated chaotic structures for information system.

So, the maintenance of the procedural systems is a complex and an expensive task and the re-use of the code is often compromised. Consequently, the modernization of these systems became indispensable.

To satisfy increasing needs to integrate applications, various tools are proposed. However, most of them present several limits [11]. On the one hand, these solutions miss abstraction and strategic methodology; they are developed without strategic vision and prioritize technological objectives. On the other hand, these solutions do not considerate the software architecture of the applications to be integrated. Moreover, none of these solutions answers strategically the need of integrating procedural legacy applications with service oriented architecture.

The interest to worry about these two paradigms, legacy and service oriented, comes because the first one is very former. Besides, the procedural legacy applications constitute a big part of the application heritage of most of the companies and envisage several difficulties for their integration with the new technologies, whereas the second is recent and fashionable. Furthermore, the service oriented architecture appears as the best approach allowing a flexible integration of the autonomous, distributed and heterogeneous applications within the company.

This paper is organized as follow: The second section gives a brief description of software architectures based on the procedural paradigm. The third section presents a meta-model that we propose for integrating procedural legacy applications with service oriented architecture.

2. THE LEGACY SYSTEMS

2.1 Definition
A legacy system is defined as " Any information system inherited from the last years, in practice from 1970 till 1995, generally developed 'in-house' to support the important functions, essential in the functioning of companies " [7].

Concerning us, legacy systems are information systems potentially impossible to maintain, whose total ownership cost is very high and for whom the skills are lacking. These systems are not rather flexible to support the evolution of the technologies and the business enterprise needs. However, these legacy systems represent the core of several information systems of companies, which depend on their specifications and their daily activities.

2.2 Problems related to legacy systems
The presence of legacy systems within companies creates certain problems, the main clauses of which are [12]:

- The technological disuse: difficulties to develop technical environments which are more or less supported by their suppliers.
- The rarity of the skills: given the competent staff on this type of applications is more and more rare (retirements, promotions, progress of career), the maintenance and the evolution of these systems became complex, even impossible.
- A significant loss of consciousness and a more and more anarchy development due to the age of the

applications and the staff turnover of the development teams.

- This type of systems underwent several revisions of code without maintaining the documentation up to date. So, it will be more difficult to understand their functionalities.
- The challenge to modernize enormous IT applications in a reasonable cost without risking to destabilize the daily functioning of the organization.
- The concern to release itself from the heaviness and from the rigidity of the legacy applications which were developed 20 or 30 years ago and which had the effect of crystallizing the working processes and often of depending them on the computing processes and on the data of the company.

Thus it is relevant that the preservation of the legacy systems, their evolution and their modernization put more and more a major challenge to the companies which, at present, are very dependent on technological environments of another age.

2.3 The legacy procedural systems

During the last three decades, a significant number of software was developed basing on the procedural paradigm. Nowadays, these developments establish a big part of the application heritage of companies. Indeed, this paradigm allowed to release itself from the code machine and to strengthen a separation between the representation of the data and the processing made on these data. However, the development of complex systems using it paradigm generated structures in "dishes of spaghetti ". Besides, the interoperability between the procedures is made by the global variables which are approachable by various procedures what engendered side effects. Moreover, the inter-dependence between the procedures makes that during the maintenance, the modification of a procedure can affect the others.

2.4 The migration towards SOA: benefits and techniques

According to Jiang and Willey [3], the service oriented architecture is an architectural style of development and of dynamic integration of the company's applications. It allows to the information system to be a collection of services (the basic brick of this architecture) which can be reused by the other information systems. The systems based on the service oriented architectures can offer a flexible solution to the problem of integrating information systems, data and processes.

Consequently, the strategies of modernization by migration towards SOA become more and more needed by companies to benefit from advantages of this architecture. So, it implies changes more vast than simple maintenance, while preserving a significant part of the former system. These changes often include the restructuring of the system, the improvement of the features or the modification of the software attributes [6]. So, the resultant system well be more agile and allows the company to release itself from problems bound to the re-use, the composition, the interoperability, and the coupling reduction between the various components of the information system by means of the Web services.

The evolution from legacy to SOA can be beneficial from both economical and technical perspectives. From an economical perspective, legacy to SOA evolution fosters change management including intra-organizational changes, and changes in enterprises [5][8]. From a technical perspective, seamless enterprise collaboration through service composition [4] and reduction in maintenance cost are claimed as long term benefits [8][10]. Motivated by these benefits, there has been significant research in legacy to SOA evolution. However, there is no systematic overview of legacy to SOA evolution, particularly focusing on the techniques, methods and approaches used to evolve legacy systems to a SOA environment. In the systematic literature review conducted by Razavian [9], an overview of SOA migration families is reported. It focuses on classifying the SOA migration approaches into eight distinct families. The classification is inspired by the reengineering horseshoe method [2] rather than giving a historical overview of SOA migration methods. Also, a brief overview of legacy to SOA evolution is reported by Almonaies [1] that divides the legacy to SOA evolution approaches into four categories: replacement, redevelopment, wrapping and migration.

3. A META-MODEL FOR INTEGRATING LEGACY PROCEDURAL APPLICATION INTO SERVICE ORIENTED ARCHITECTURE

In this section, we propose a meta-model for integrating a legacy information system with service oriented architecture. This solution is based on the "Wrapping" principle, which consists in surrounding the company's legacy system by a software coat which hides its complexity and its heterogeneousness and exports a modern interface.

The software coat consists of a set of adapters or "Wrappers", which are mainly used to mask the disagreement between the exported interface by the former system and the interfaces required by the new environment. Such solution requires a thorough understanding of the various modules of the system to be adapted [6].

3.1 A general model for the proposed solution

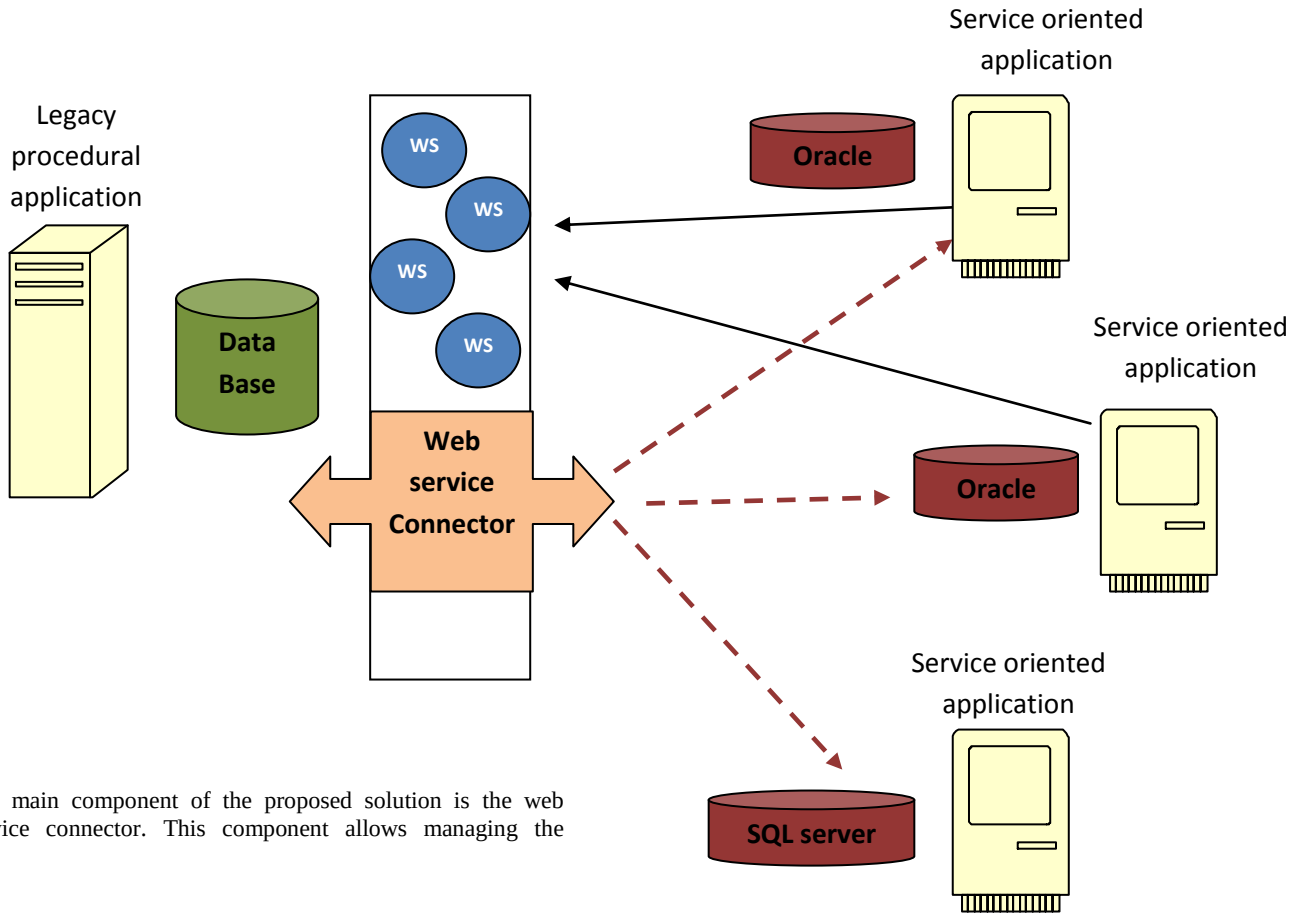
The proposed solution concerns systems composed by a myriad of heterogeneous applications, which are procedural legacy applications and service oriented applications. These two types of applications are confronted with a problem of the interoperability between their software architectures. So, we suggest encapsulating the procedural application by a software coat in order to mask this heterogeneousness and incompatibility, as comic in the figure1.

This coat must be separated from the procedural application, and having an independent implementation that it must not affect the functioning of this application. This software coat has to support service oriented architecture. Therefore, the procedural application is seen as being a service oriented application, as far as it can:

- On the one hand, consume the services supplied by the other applications, through the *web service connector*. Indeed, this component makes the intermediary between the procedural application and the other service oriented applications in order to allow him to consume their services. Moreover, it takes care of the data exchanges managing, namely the format compatibility of managed data, the synchronization and the integrity of the data...

- On the other hand, to expose its reused features as web services, that should be consumed by external other applications.

exchange in both directions between the concerned information system and the external applications, as well as the synchronization of the data between the various managed Databases. The design and the implementation of this component are exposed on the next subsection.



The main component of the proposed solution is the web service connector. This component allows managing the

Fig 1: General architecture of the proposed solution

3.2 The web service connector designing

This component that we consider essential for our process of integration will be the object of this section. In fact, with the aim of loosening a generic design of the connector, we propose a set of solutions for implementing this component.

3.2.1 Implementing the web service connector as a web service

The web service connector should be considered as being a web service which takes care of the integration at the level of data, including the management of the data compatibility and the data synchronization...

So, some applications cannot support calls of web services (applications in COBOL, VB6,…), what requires the use of tools allowing the invocation of these services. We propose in this case the use of the "soap toolkit" tool of Microsoft.

The major inconveniences of such solution are that:

- This tool, although it is compatible with a significant number of programming languages, imposes constraints on the nature of the concerned system, as far as this solution is applicable only to Windows operating systems.
- The procedural application should call web services, what means that it became a customer of Web services, what will affect in a way its procedural architecture.

3.2.2 Implementing the web service connector as a Dynamic Link Library (DLL)

To mitigate the inconveniences of the previous solution, we tried not to affect the architectural characteristics of the procedural application. So, the web service connector takes care of the invocation of the web services and the data exchanges.

To do it, we propose to conceive our web service connector as being a Dynamic Link Library. So, the procedural application uses the existing web services by means of the DLL functions without caring about their details of invocation.

The advantage of the DLLs is that it can be developed and called by various programming languages; thus, the implementation of the DLL is independent from that of calling it.

However, a DLL works only under Microsoft Windows. There are similar mechanisms for the other operating systems (the shared libraries '.so' for Linux for example). Besides that, the dependence in the operating system and the application is not autonomous anymore; it needs a separate module of DLL.

3.2.3 A description of the interface in IDL (Interface Definition Language)

Both solutions proposed previously, although they are functional and effective, add constraints to the concerned system. So, to release itself from requirements depended on platforms, on implementation and to give a generic conception of the connector, we suggest describing the interface of the component in IDL.

So, an effective interaction between incompatible applications on heterogeneous platforms requires a strict separation between the interface and the implementation. IDL helps to carry out this separation; it defines the types of objects by specifying their interfaces. An interface consists of a set of operations and their parameters. Afterward, we should make a projection of this interface definition towards an implementation suitable to the studied system. We note that the interface definition depends on the concerned system seen that every system requires a set of operation specifying him.

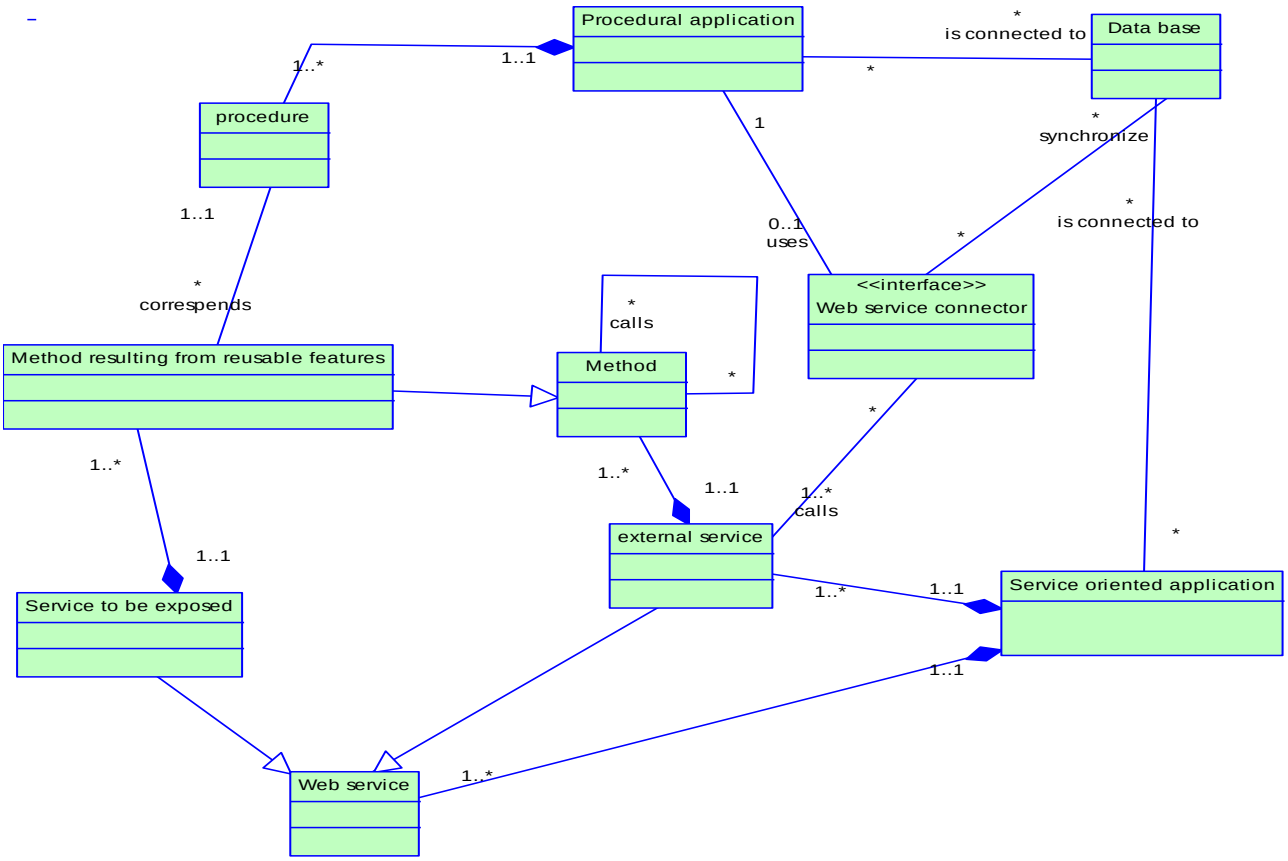
3.3 Architecture meta-model of the target system

Our model of architecture illustrated by the figure2 can be considered as meta-model because it will be instantiated within every information system, to be able to integrate his procedural applications into service oriented architecture.

The procedural application is composed of a set of procedures containing its processings. These procedures represent the features offered by the procedural application which can be reused by the other oriented service applications in the form of Web services; these are the services that well be exposed.

Of other one quoted, to be able to consume the external Web services, the procedural application needs a component which comes to interface with that this, to allow it to communicate with heterogeneous service oriented applications. This component serves to call upon existing Web services and to communicate the result with the procedural application.

Fig 2: Architecture meta-model of the target system



Before instancing the meta-model on the concerned information system, the latter has to undergo several processings to determine its architectural and functional characteristics.

These treatments are based on the analysis of the concerned system. It supplied the necessary information concerning the architecture of the system, the software components which constitute it and the current state of the system (as the level of

maintenance, the level of complexity and the level of coupling of its software components)… So, we can speak about a reengineering of the concerned information system which can be based on several steps such as:

- *Use cases study:* it allows us to have a global vision of the functional behavior of the concerned system and afterward, to know the features offered by the various applications. This stage can be combined with the reengineering of the business process of the concerned system to be able to insure an effective re-use of the features.
- *Reengineering of business process:* this step is based on the revision, the update and the analysis of the existing "As Is". Afterward, this stage recommends the design of the target business process "To Be ". This stage allows us to re-conceive the business logic of the system, to find the improvements which be brought to the procedural application through the consumption of the existing services and identify the features which it can expose as web services. Then, we suggest modeling this business process, by using for example the language BPMN (Business Process Management Notation).
- *Cartography of data:* this step allows to list all the structures and the constraints concerning the data managed by the concerned system and to understand the data exchanges. This stage is necessary to define the integrity constraints of data and insure their synchronization during the integration. Besides, this cartography of data is indispensable afterward, because it gives us a global vision onto the managed data which facilitates the integration at this level.

4. CONCLUSION

Generally, the information systems of companies present a delicate assembly of independent applications designed to satisfy a specific need which engendered complex architecture hardly managed, where from the necessity of the adoption of an approach of IS urbanization and the integration of its applications. The importance of such approach lives in the capitalization of the application heritage of the company and its integration into modern environments.

From this assertion, in this paper, we have exposed a meta-model for integrating legacy procedural applications into oriented service architecture.

To test its efficiency, the proposed meta-model was applied on a procedural information system implemented with VB6, that we want to integrate it with external web services implemented with VB.net.

We also have tested the data synchronization between the existing database source (implemented with Microsoft access on 2003) and a new database that we implemented with MYSQL4.0.

The results are very satisfying, and the proposed solution will be confirmed by the other experimentations.

5. REFERENCES

[1] Almonaies, A. A., Cordy, J. R., and Dean, T. R. (2010). *Legacy system evolution towards service-oriented architecture.* Paper presented at the Interrnational Workshop on SOA Migration and Evolution (SOAME 2010), Madrid, Spain.

[2] Bergey, J., Smith, D., Weiderman, N., and Woods, S. (1999). *Options Analysis for Reengineering (OAR): Issues and Conceptual Approach* (No. CMU/SEI-99-TN-014): SEI.

[3] Jiang, M. and Willey, A. Service-Oriented Architecture for Deploying and Integrating Enterprise Applications, Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), 2005.

[4] Khadka, R., and Sapkota, B. (2010). An evaluation of dynamic web service composition approaches. Ppresented at the 4th International Workshop on Architectures, Concepts and TechnologiesService Oriented Computing (ACT4SOC 2010).

[5] Khadka, R., Sapkota, B., Pires, L. F., Sinderen, M., and Jansen, S. (2011). Model-Driven Development of Service Compositions for Enterprise Interoperability. Paper presented at the 3rd International IFIP Working Conference on Enterprise Interoperability (IWEI'11). Retrieved from http://dx.doi.org/10.1007/978-3-642-19680-5_15.

[6] Malinova, A. Approaches and techniques for legacy software modernization, 2010.

[7] National Computing Centre, How Advanced Are Your Legacy Applications?, Atos Origin, May 2006.

[8] Papazoglou, M., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. Computer, 40(11), 38-45.

[9] Razavian, M., and Lago, P. (2010). A Frame of Reference for SOA Migration. In E. Di Nitto and R. Yahyapour (Eds.), *Towards a Service-Based Internet* (Vol. 6481, pp. 150-162): Springer Berlin / Heidelberg.

[10] Schelp, J., and Aier, S. (2009). *SOA and EA-sustainable contributions for increasing corporate agility.* Paper presented at the 42nd Hawaii International Conference on System Sciences.

[11] Spackman, D. and Speaker, M. Solution for the EAI, French Quercy, Microsoft edition, February 2005.

[12] http://www.technocompetences.qc.ca/files/systemes_patrimoniaux.pdf, March 2012.Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.