Semi-automatic Creation of Adapters for Legacy Application Migration to Integration Platform Using Knowledge

Jan Pieczykolan[1,2], Bartosz Kryza[1], and Jacek Kitowski[1,2]

[1] Academic Computer Center CYFRONET-AGH,
Nawojki 11, 30-950 Cracow, Poland
[2] Institute of Computer Science, AGH University of Science and Technology,
Mickiewicza 30, 30-059 Cracow, Poland

Abstract. This paper presents a solution for semi-automatic creation of adapters – missing components between integration platforms or application migration frameworks and legacy applications – useful for collaborative environments. It introduces basic problems related to the integration process, presents a common elements of an integration platform and emphasizes a crucial role of creation of the adapter. The proposed solution is based on expert system approach; it speeds-up the whole process of development of an adapter, making the migration cost-effective, performance tuned and error prone.

1 Motivation

In the last period of time great IT expansion has been observed. Many of new applications (especially business ones) are created and deployed at enterprise organizations. But now, these applications have to coexist in a common environment with older ones – called legacy applications – which have been developed earlier often on old computer architectures with programming languages currently not supported.

Many of such legacy systems are critical – they are still running doing their job properly and efficiently. The main problem is that they often exist separated from other applications, having its own business logic, data model and data which often are the same for many of such systems (e.g. customer data).

Typical examples of such domain systems are coexisting billing and loyalty systems. Each of them has a database with similar client data. The problem appears when the company has to make a change in a particular client's data – the change must be propagated across a large number of domain systems, which often results in their inconsistency.

In response to such problems the industry has created a concept of the integration platform. The main role of this kind of software is to introduce a common data model and provide a common layer of communication between applications in a manner that enables them to invoke operations on integrated systems in a standardized way but also allows receiving notification of changes in other systems and taking suitable action to handle them.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part IV, LNCS 3994, pp. 252–259, 2006.

Integration of an application still requires a creation of a bridge which will plug the application into the platform. This process needs programming effort to develop a component which will communicate with an application, providing the platform with its functionality. This component is called an adapter; the process of adapters' development can be made easier by providing intelligent tools for developers. These tools can make the development of the adapter semi-automatic by providing an expert system to select an adaptation strategy together with a knowledge base for storing a description of it. Such a solution can accelerate adapter creation making the whole process more reliable and error prone and in the effect – increasing the application ROI. Also, by using the knowledge base the best adaptation practices can be stored and reused to increase the overall performance of the final component.

This paper presents a software solution for semi-automatic creation of the adapter. It describes its basic components and shows how they use each other to provide adapter logic for communication with the application.

2 State of the Art

A state-of-the-art section is split into two parts – the former, describing the classical software created especially to help with migration of legacy code to the Grid platform and the latter, which presents sample commercial integration solutions.

2.1 Migration of Legacy Applications to the Grid

There are two existing solutions which can support migrating legacy applications to the Grid environment.

GEMLCA [1, 2] focuses on migration of software, which represents typical computational problems. The range of its usage is rather limited, because business problems in the context of an integration are more related to data and distributed application API access rather than to performing computations.

The concept presented in LGF [3] is far more universal. Mainly, LGF is dedicated to wrap a legacy system and allow transactional and concurrent invocation of its methods. But it requires the user to provide an adapter, which is responsible for communication with the legacy system. The effect of the migration process (presented in Fig. 1) is a Globus Toolkit service, which connects to the

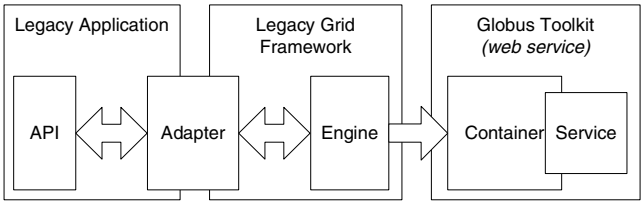

Fig. 1. Scheme of the migration process using LGF

application by Legacy Grid Framework that uses the adapter to communicate with the application.

2.2 Integration Platforms

Many commercial software products exist on the market. These are often highly complicated, sophisticated software packages, which offer complete solutions to a set of problems in the enterprise. They typically include the following components:

- Communication layer – responsible for exchanging messages between applications (e.g. IBM MQ Series, webMethods Broker, other JMS providers),
- Business process engine – that allows for easy creation of business processes using services provided by integrated applications (e.g. webMethods Modeller, Tibco StaffWare, JBoss jBPM),
- Integration server – responsible for hosting wrapped functionality of legacy applications and making it accessible on the platform.

Some products offer only partial integration functionality, for instance Tibco Rendezvous [4] offers only a middleware for connecting applications, without other services. All the above mentioned products lack functionality of easy adaptation of applications. Of course, adapters for popular products such as databases and JMS providers exist, but for tailor-made systems there is always necessity to create a dedicated adapter, that wraps its specific functionality.

3 Main Problems with Legacy Migration

Usually many problems which arise while migrating a legacy application to the Grid environment are related not only to the architecture of a particular application or a technology used, but also to development effort of the adapter, which will act as a bridge between the Grid and the application.

Specific problems of any particular application are related to its construction. A particular application can have no interface to which we can easily connect and exchange data, it is also likely to use domain-specific data structures, which have to be converted to common data structures called Canonical Data Model and the modification of the application interface may not be possible because of specific reasons, some of which had been described in [7]. This part of the migration process is definitely the most difficult one; it could be simplified by arranging an expert system, which would assist the user during the process. Such a system could offer hints, related to the type of the application interface and also help by providing contact information to a person who has already migrated a similar system – but this approach still requires a lot of programming effort while implementing the application specific logic part of the adapter.

The latter problems, related to the development effort, are definitely easier to solve. A general skeleton of the adapter can be implemented and the missing part of it, which is application specific logic, can be injected in it while the application

is deployed on the integration platform. Of course, the adapter skeleton has to be dedicated for every legacy application migration framework, but it is easy to provide a set of skeletons for every framework because of a low number of them.

4 The Platform

The main responsibility of the platform is to simplify the process of developing the adapter for a particular application. As mentioned above, universal frameworks for adapting applications already exist, but they need adapters for performing communication with a particular application. The adapter, can be created on the fly, by using a semi-automatic method of creating communication logic. This can be done by providing a formal application interface description, a set of adaptation strategies and a selector for them.

4.1 Adapter

The adapter is an element which is responsible for handling communication between the application and a particular legacy application migration framework. It can be composed of two elements (cf. Fig 2):

- *Logic* – which is an application specific element; its responsibility is to handle communication with this particular application, including invocation of methods and conversion of applications' data structures to the Canonical Data Model; logic expose a well-defined interface to a skeleton,
- *Skeleton* – which is a common element; its responsibility is to provide a runtime environment for *the Logic* element.

The Logic is embedded in *the Skeleton* – it makes methods of the adapted application available to the legacy application migration framework. A method call is invoked on *the Skeleton* and it is passed to *the Logic*, which calls the legacy application method.

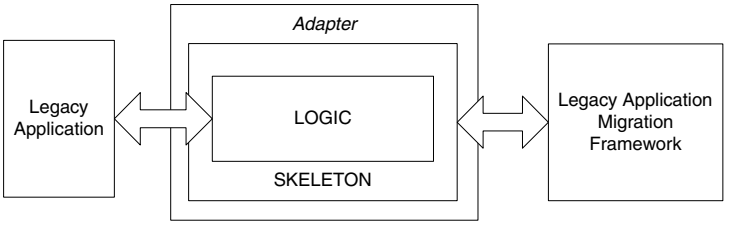

Fig. 2. Adapter block diagram

4.2 Application Adaptation Process

Development of the Adapter can be automated by providing a common Adapter Skeleton, compatible with a particular legacy application migration framework,

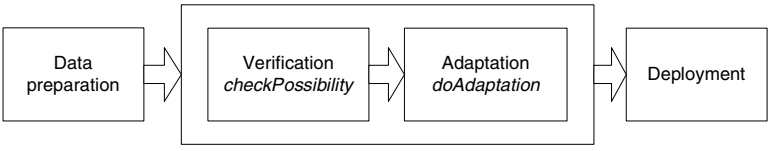

Fig. 3. The overall process of adaptation

and a mechanism for semi-automatic creation of the Adapter Logic. *The Skeleton* is delivered with the platform but *the Logic* must be generated for every application separately because it is application specific.

Development of *the Logic* can be made semi-automatic by utilizing an expert system to select a proper adaptation logic strategy together with a knowledge base to get description of this application interface. The process of adaptation consists of the following phases (see Fig. 3):

- data preparation – it consists of steps related to preparation of knowledge for a given kind of application, i.e., annotating its interfaces and defining the strategy; if already migrated, there is no need for this step,
- verification of data – the "checkPossibility" step consists of actions related to checking if there is any available strategy for a given application, if that application is described properly and if the knowledge base is ready to use,
- adaptation – the "doAdaptation" step describes all actions related to preparation of *the Skeleton* class (or classes), also, in this step the knowledge base is queried for the application description according to the strategy that has been determined in the previous step; finally *the Logic* is generated,
- deployment – the final step of adaptation, currently all required classes are prepared for use with a particular legacy application migration framework, i.e. they can be deployed as a service representing the adapted application.

4.3 Adapter Creation

The solution for semi-automatic development of the adapters' logic requires the following components (see Fig. 4):

- Knowledge Base – a module which role is to hold and manage descriptions of applications used by the strategy selector and the adaptation logic generator; it is used in the "Data preparation" step,
- Strategy Selector – a module which selects a strategy for application adaptation in the "checkPossibility" step,
- Adaptation Engine – a core part of the implementation which uses knowledge base and strategy selector to generate the adapter; it performs the mentioned "doAdaptation" step.

The above components are loosely coupled, so it is allowed to replace a concrete implementation of each component by the solution more suitable for the current needs. For example, if the knowledge base uses an ontology approach to

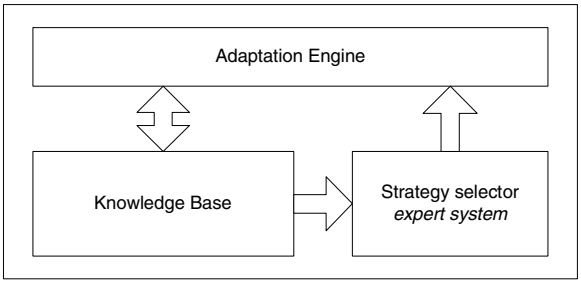

Fig. 4. Adaptation Platform architecture

represent data it can be replaced by another knowledge base implementation, using 'if-then' rules to express knowledge. The sufficient condition is that the replacing element should fulfill the interface contract with other components.

4.4 Knowledge Base

Descriptions of applications and their interfaces are stored in the knowledge base component of the system. The current implementation uses simple XML in-memory database approach to store them, but an ontological one seems to be more robust and efficient. When the application or its interface are described in this more formal way than a simple structured XML file, it is easier to reuse gathered knowledge while adapting other similar applications. This description could be an ontology individual stored in Grid Organizational Memory [5, 6].

4.5 Strategy Selector

The strategy selection process is supported by an integrated expert system. It contains descriptions of strategies with links to particular entities in the knowledge base. The adaptation strategy is a set of steps which have to be performed to create the application specific logic element of the adapter.

The current approach to strategy selection makes use of Bayesian Networks [8]. The overall process of selecting the adaptation strategy relays on the user feedback. The strategy selector asks the user a couple of questions. By the use of the Bayesian approach the system can suggest the best answer based on its percentage probability (e.g., there are more database systems than application server ones, so the system can suggest the user that it is more probably that he adapts a database system). The system can have predefined strategies (e.g. for C-language function adaptation, for Oracle PL/SQL procedure, etc.).

4.6 Adaptation Engine

The engine is a core part of the platform. Its main responsibility is to generate *the Logic* for accessing a concrete application and create an adapter by combining the provided skeleton and that logic. If the user has not provided a configuration

file for the adaptation he is asked for that information. Then, when the strategy is known, the Adaptation Engine is trying to get description of the application from the Knowledge Base. If this step fails, the application is introspected to develop its description (which can be later stored in the Knowledge Base for reuse). If introspection fails a user is asked to provide the application description. Based on gathered data *the Logic* is generated and then combined with a proper Skeleton. The adapter is ready to use.

4.7 Application Adaptation Example

Many backend applications have PL/SQL interfaces. Such an interface consists of a group of database procedures which are similar to the C-language methods, they have arguments of basic or complex types and there are no return values. The only difference is that arguments can be input, output or both – input and output.

The Adaptation Engine (AE) asks the user a set of questions regarding the application to adapt. The conclusion is that PL/SQL Procedure strategy should be applied to the adaptation process. Following the selected strategy, AE requires additional information from the user about database connection parameters (hostname, port number, database name, user name and password). Based on these values it connects to this database by using JDBC [9] and performs introspection to get a list of existing procedures. The user selects the procedure to adapt and then AE gets procedures' attributes and tries to map them to a particular common data type from the Knowledge Base (KB). For instance, if the procedure name is *add_contact*, it probably will have attributes like *first name, last name, date of birth, address*, etc. KB looks through existing mappings and finds, that such attributes belong to the *ContactModel* type. AE creates an adapter for the procedure *add_contact*, which takes only one attribute of the *ContactModel* type, being able to connect to a particular database instance and invokes that procedure.

5 Conclusions and Future Work

The solutions used to migrate legacy applications to integration platforms (cf. sections 2.1 and 2.2) require an efficient adaptation layer to communicate with existing legacy applications. Such a layer can be created from scratch every time when a new application is integrated for the price of higher time-to-market period, lower efficiency, lack of serviceablity and quality. There is a need for a dynamic adaptation layer, i.e. a set of components, which can help with adaptation using semantic description of services and data. Also, there is a place for an expert system to support their selection in a semi-automatic way.

The main target of presented solution is to make an adaptation process easier and semi-automatic. Use of the expert system for adaptation strategy selection together the knowledge base to keep description of application allow the user to easily extend it with new functionality and scale to fit one's needs.

Acknowledgments

This research has been done in the framework of EU IST-2002-511385 K-Wf Grid project. Discussions with Prof. Gabriele von Voigt and Nils Jensen from the University of Hannover and AGH-UST grant are also acknowledged.

References

1. Delaitre, T., Goyeneche, A., Kacsuk, P., Kiss, T., Terstyanszky, G.Z., Winter, S.C.: GEMLCA: Grid Execution Management for Legacy Code Architecture Design, Conf. Proc. of the 30th EUROMICRO Conference, Special Session on Advances in Web Computing, Rennes, France (2004) 477-483
2. Terstyanszky, G., Delaitre, T., Goyeneche, A., Kiss, T., Sajadah, K., Winter, S.C., Kacsuk, P.: Security Mechanisms for Legacy Code Applications in GT3 Environment, Conf. Proc. of the 13th Euromicro Conference on Parallel, Distributed and Network-based Processing, Lugano, Switzerland (2005) 220-226
3. Baliś, B., Bubak, M., Węgiel, M., A Solution for Adapting Legacy Code as Web Services, in: V. Getov, T. Kielmann (Eds.), Component Models and Systems for Grid Applications, Springer (2005) 57-75.
4. Tibco: TIBCO Rendezvous(TM) Concepts, Software Release 7.0, April 2002, http://www.tibco.com.
5. Krawczyk, K., Słota, R., Majewska, M., Kryza, B., Kitowski, J.: Grid Organization Memory for Knowledge Management for Grid Environment, in:Bubak, M., Turala, M., Wiatr, K. (Eds.), Proceedings of Cracow Grid Workshop - CGW'04, Dec.13-15,2004, ACC-Cyfronet AGH, Cracow (2005) 109-115.
6. Kryza, B., Majewska, M., Słota, R., Kitowski, J.: Unifying Grid Metadata Representations through Ontologies, in Wyrzykowski, R., et al.(Eds.), Proc.Inter.Conf. PPAM2005, Sept.12-14, Poznan (2005)
7. Abramson, D.,Kommineni, J.,McGregor, J. L.,Katzfey, J.: An Atmospheric Sciences Workflow and its Implementation with Web Services, Future Generation Computer Systems, 21(1) (2005) 69-78
8. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning, Kluwer, 29 (1997) 131-163.
9. http://java.sun.com/products/jdbc/