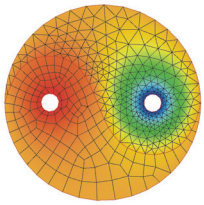
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 11, 2011, No. 2



SERVICE ORIENTED FRAMEWORK FOR DEVELOPMENT OF COMPLEX ENGINEERING SOFTWARE

NORBERT SCZYGIOL, JULIUSZ MIKODA, ANNA WAWSZCZAK*

Institute of Computer and Information Sciences, Czestochowa University of Technology
ul. Dąbrowskiego 73, 42-200 Częstochowa, Poland
**Corresponding author: anna.wawszczak@icis.pcz.pl*

Abstract

The service oriented architecture is commonly used in business software applications, but could also be successfully used while designing engineering packages. This paper describes a framework for development of complex engineering simulation software. It is designed on the basis of the service oriented architecture. Applications are built by composition of modules that realize particular step of simulation. Such modules are implemented with use of the Web Services technology. The architecture of the described framework is based on two layers: abstract layer and concrete layer. Applications are constructed from building-blocks on the abstract layer. The concrete services from concrete layer are matched with abstract services on the basis of constraints and Quality of Services attributes. The problem of sending large amount of data was also considered. The results of the conducted research on sending data in different formats were presented. It was shown that using proper data structure avoid decrease in application efficiency. The last section presents the utility of service oriented architecture on the example of service called e-MeshGen that offers functionality of finite element mesh generator.

Key words: finite element method, service oriented architecture; web services; engineering software; sending large data structures

1. INTRODUCTION

Over recent years the development of the Internet has resulted in significant increase in the popularity of applications based on service oriented architecture. This kind of architecture is commonly used in business software applications. Nevertheless, it also could be successfully used while designing engineering packages.

A substantial majority of engineering simulations, most notably simulations that use advanced numerical methods, could be easily decomposed into steps, that are independent of each other. For example, finite element method consists of three basic steps: preprocessing, processing and postprocessing, but these steps could be also divided into smaller tasks. Preprocessing module includes shape creating, mesh generation and optimization as well as defining boundary condition. In the processing phase, we should build global matrix, impose boundary conditions and solve linear equation system. The additional steps such as mesh optimization and renumbering or matrix preconditioning are also performed.

The fact that this steps are independent enables to accomplish them by autonomous modules. These modules could be implemented as services that provide particular functionality and afterwards they could be merged into the whole.

ISSN 1641-8581

2. FRAMEWORK

This paper describes our work-in-progress related to the framework for development of complex engineering simulation software. This framework is designed on the basis of the service oriented architecture and Web Services technology described by W3C Working Group (2004). Such approach gives operating system and programming language independence. Any operating system and any programming language can be used to build modules that are capable of cooperating with each other using standard internet protocols, such as HTTP or SSH.

Engineering applications are build by composition of services that realize particular steps of simulation. Therefore such services are accessible through SOAP (Simple Object Acces Protocol) developed by W3C Working Group (2007), they can be used not only in applications that are build with use of the described framework but also successfully invoked by external engineering packages without relaying on the framework. This software could use such services to accomplish just some particular steps of simulation.

3. SENDING LARGE AMOUNT OF DATA

Engineering applications usually operate on large amount of data, that are produced by mesh generators as well as processing modules. In particular, it applies to matrices. The sizes of matrices, that are created during conducting simulations that use numerical methods (such as Finite Element Method), reach hundreds of thousands or even a few million of elements. In engineering applications built from blocks, that are implemented as Web Services modules, these large data have to be exchanged between particular services. Sending such big data structures using standard protocols could significantly decrease the efficiency of application.

Nevertheless, this problem could be partially avoided by taking into consideration characteristic of matrices. In most cases, matrices created during computations have very few non-zero elements and can be regarded as sparse matrices. Several sparse matrix formats exist, described by Dongarra (2000). The most popular of them are: CRS (Compressed Row Storage), JDF (Jagged Diagonal Format) and CDS (Compressed Diagonal Storage). These formats take advantage of specific properties of the sparse matrix. The same properties can be used while exchanging matrices between modules based on Web Services. Conducted research, described by Mikoda et al. (2010), show that using appropriate data structures and appropriate data transcription forms enables to reduce the amount of time required for sending these data, to the relatively low level (in proportion to the amount of time required for computations).

4. SERVICE MANAGEMENT MODULE

The service management model used in the described framework is based on the idea of abstract and concrete services, described by Canfora et al. (2005). Such approach leads to the dividing framework into two layers: abstract layer and concrete layer. Elements of each layer are shown in figure 1.

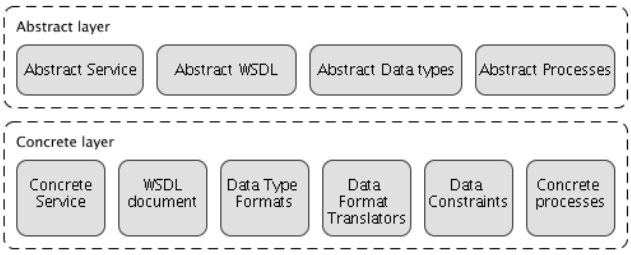


Fig. 1. *Framework layers.*

5. ABSTRACT AND CONCRETE SERVICES

The abstract service could be regarded as abstract module that is dedicated to perform some particular tasks. The abstract service groups concrete services that are functionally equivalent. The concrete services provide the same functionality, however they could exhibit different Quality of Service (QoS) attributes or constraints on data types. They could also perform their tasks in different ways.

Abstract service is defined by abstract WSDL (Web Service Description Language) developed by W3C Working Group (2001) document that consists of input and output data type definitions. These types are defined as abstract data types (for example Matrix or Vector). Concrete services are described by WSDL files, input and output data format definitions and constraints on these data. The input and output data formats are connected to input and output data types from abstract layer.

The exemplar abstract service and two concrete services are shown in table 1. The linear equation system solver takes matrix and vector as input and returns vector as output. Two implementations of such solver differ in constraints and Quality of Service attributes. The first one (Conjugated Gradient Method) is faster, but it could be applied only to symmetric and positive-definite matrices. The sec-

ond one (Biconjugated Gradient Method) is slower, but it could be applied to any matrices.

Table 1. *Abstract and concrete services example.*

Abstract Service	**Linear equation system solver** Input: *Matrix and Vector* Output: *Vector*	
Concrete Service	**Conjugate gradient method** QoS: *faster* Constraints: *Matrix has to be symmetric and positive-definite*	**Biconjugate gradient method** QoS: *slower* Constraints: *no constraints*

6. TESTING CONCRETE SERVICES

Concrete services, that are connected to the same abstract service, should provide the same functionality. In most cases, two concrete services invoked with the same input data should return the same output data. Using abstract and concrete services gives the possibility of automated testing of concrete services service. The testing module, that is part of the described framework, enables to check if the particular concrete service realizes its functionality correctly. The sets of unit tests defined for the abstract services are used for this purpose. For example, set of tests dedicated for linear equation system solver consists of a few linear equation systems and their solutions. When the new concrete service is added to the framework, it is automatically tested with these unit tests. Because most of services use float pointing operations, the results of service invocation have to be compared to the expected results with given tolerance. The unit tests verify the correctness of service functionality, but the testing process is also used to determine the time efficiency of the concrete services. The results of such tests are used to specify the Quality of Service attributes.

There are some abstract services, that can not be verified in the way described above. In some cases, the concrete services that belong to the same abstract service group, could return equivalent but not identical data. For example, the services that realize the imposition of boundary conditions, could perform their task in a few different ways. Dirichlet boundary condition can be imposed on the linear equation system in, at least two different ways. The data returned by such services could be different, but from engineering point of view equivalent and correct.

7. BASIC AND ADDITIONAL SERVICES

Abstract services are divided into two groups: basic services and additional services. Basic services converts some data into another. Additional services are responsible for the improvement of data quality.

Table 2. *Basic and additional services examples.*

Basic services	Additional services
– equation system building service – equation system solver – mesh generator service	– finite mesh refinement service – finite element mesh renumeration service – matrix preconditioner service

These services return the same type of data as they take as input. A few examples of basic and additional services are listed in table 2.

8. SERVICE COMPOSITION MODULE

The service composition module is one of the most important parts of framework. It enables to compose a few services into one complex service. The service composition is performed on the abstract layer. Abstract services are arranged into abstract processes destined for the realization of particular task. This process could be regarded as a new abstract service and it could constitute a building block of another processes. The exemplar abstract process of the Finite Element Method simulation is presented in figure 2.

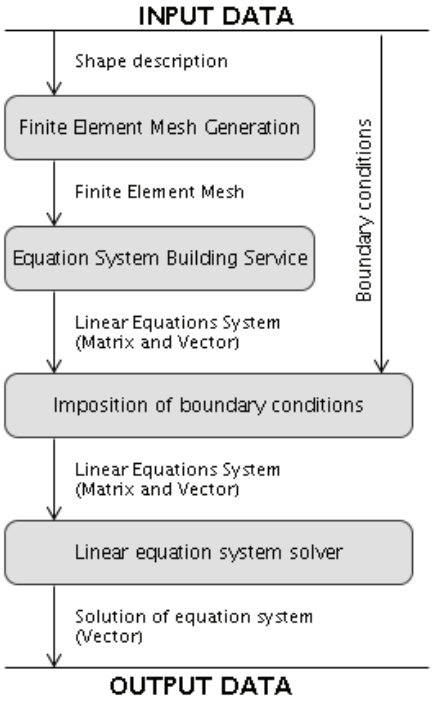


Fig. 2. *Abstract process of Finite Element Method simulation.*

When the abstract process is ready, the concrete services are matched to abstract services. The services are chosen in such way so that all constraints on input and output data are respected. If there is more than one set of services that fits to the abstract process model, the concrete services are chosen on the basis of the Quality of Services attributes and data formats matching.

9. COMPOSITION OF ADDITIONAL SERVICES

Adding additional services is the last step of service composition process. While performing this task the following information are considered:

– additional service recommendation,
– data type formats,
– Quality of Service attributes.

Because additional services are optional, they can be removed or added to the process by user.

10. E-MESHGEN

E-MeshGen service, described by Sczygiol et al. (2010), is a practical example that shows the possibilities of using service oriented architecture and technologies such as Web Services while designing engineering software. This service provides functionality of 2D and 3D finite elements mesh generator. The authorial mesh generator, developed by Sczygiol&Mikoda (2008) was used for this purpose. It enables generation of finite element meshes for 2D and 3D multi-domain areas.

The interface provided by the service enables:

– loading description of the area's shape (in GID ASCII format, described in Gid Reference Manual),
– setting generator's parameters,
– launching and supervising time-consuming process of mesh generation.

The mesh generated during this process could be obtained as text, image (GIF file) or as a set of points (this format is dedicated for OpenGL visualization).

11. SUMMARY AND CONCLUSIONS

This paper presents the work-in-progress related to the framework for development of complex engineering software, based on service oriented architecture and Web Services technology. This framework shows that this kind of software architecture can be successfully used while building complex engineering applications.

The described framework is based on the layered architecture. It is divided into abstract layer and concrete layer. The abstract layer is comprised of abstract services and abstract processes that are built from such services. Elements that belong to the concrete layer are matched to their abstract counterparts. Layered architecture used in the described framework gives some benefits, such as automated testing of services. The applications created with use of the described framework are provided as services accessible through SOAP and can be easily used in independent software.

The development of this framework shouldn't be regarded as research on Web Services technology and service composition methods. It is designed as practical tool for engineers, especially for these who are working in the field of mechanic, that want to conduct some engineering simulations. Reusing modules in such framework could significantly decrease the amount of time, that they have to spend building their applications.

REFERENCES

Dongarra, J., 2000, *Sparse matrix storage formats*, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia.

GiD: The Personal Pre and Post Processor, *GiD Reference Manual* http://gid.cimne.upc.es/.

Canfora, G., Corte, P., De Nigro, A., Desideri, D., Di Penta, M., Esposito, R., Falanga, A., Renna, G., Scognamiglio, R., Torelli, F., Luisa Villani, M., Zampognaro, P., 2005, The C-Cube Framework: Developing Autonomic Applications through Web Services, *Proc. 2005 workshop on Design and Evolution of Autonomic Application Software*, 1-6.

Mikoda, J., Sczygiol, N., Wawszczak, A., 2010, *Metody reprezentacji danych w aplikacjach inżynierskich wykorzystujących architekturę SOA*, Metody Informatyki Stosowanej, (in Polish).

Sczygiol, N., Mikoda, J. 2008, Renewed Generation of Finite Elements Meshes in Parts of Domain Changing from Structural and Technological Respects, *Proc. RELMAS'2008, Assessment of Reliability of Materials and Structures: Problems and Solutions*, 1, 322-325.

Sczygiol, N., Mikoda, J., Wawszczak, A., 2010, Web service for finite element mesh generator, *Computer Methods in Materials Science*, 10, 176-180.

W3C Working Group 2001, Web Services Description Language (WSDL), W3C note,. http://www.w3c.org/.

W3C Working Group 2004, Web Service Architecture. W3C Working Group Note, http://www.w3c.org/.

W3C Working Group 2007, Simple Object Access Protocol (SOAP), W3C Recommendation, http://www.w3c.org/.

FRAMEWORK DO TWORZENIA ZŁOŻONYCH PAKIETÓW INŻYNIERSKICH W OPARCIU O ARCHITEKTURĘ ZORIENTOWANĄ OBIEKTOWO

Streszczenie

Architektura zorientowana obiektowo jest powszechnie stosowana do tworzenia aplikacji biznesowych, ale może być również z powodzeniem stosowana podczas projektowania pakietów inżynierskich. W artykule przedstawiono platformę umożliwiającą tworzenie złożonych aplikacji przeznaczonych do symulacji inżynierskich. Framework zaprojektowany został z wykorzystaniem architektury zorientowanej na usługi. Aplikacje są tworzone przez kompozycję modułów, które realizują określony etap symulacji. Moduły te implementowane są z wykorzystaniem technologii Web Services. Architektura opisanego systemu oparta jest na dwóch warstwach: abstrakcyjnej i konkretnej. Aplikacje konstruowane są z modułów na poziomie warstwy abstrakcyjnej. Usługi konkretne, z warstwy konkretnej, są dopasowywane do usług abstrakcyjnych na podstawie ograniczeń i parametrów QoS (jakości usług). W pracy zaprezentowano również problem przesyłania dużych ilości danych. Przedstawione zostały wyniki badań dotyczących przesyłania danych w różnych formatach. Wykazano, że przy odpowiedniej strukturze danych można uniknąć spadku wydajności aplikacji. Ostatnia część pracy prezentuje możliwości zastosowania architektury zorientowanej na usługi na przykładzie usługi o nazwie e-MeshGen, która oferuje funkcjonalność generatora siatek elementów skończonych.

Received: October 19, 2010
Received in a revised form: November 24, 2010
Accepted: November 24, 2010