Advances in Engineering Software 52 (2012) 72–80

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Application of service oriented architecture to finite element analysis

R.I. Mackie *

Civil Engineering, School of Engineering, Physics and Mathematics, University of Dundee, Dundee DD1 4HN, UK

ARTICLE INFO

Article history:
Received 23 March 2012
Received in revised form 26 June 2012
Accepted 4 July 2012
Available online 1 August 2012

Keywords:
Component-oriented
Object-oriented
Distributed computing
Service oriented architecture
Finite elements
Microsoft.NET

ABSTRACT

This paper examines the application of service oriented architecture (SOA) in finite element analysis. SOA is a technology for designing and developing interoperable services. These services can reside on the same computer or, more commonly, on distributed computers. The paper demonstrates how SOA can be used within the context of scientific computing. The implementation and application of SOA to equation solvers and finite element analysis is described. There are advantages in terms of software engineering, as it facilitates the separation of areas of complexity. SOA can be used on standalone computers, intranets and on the internet. The data transfer costs are examined. It is shown that SOA principles can be used to design applets that make use of finite element analysis and a simple example of this is described.

1. Introduction

The finite element method has seen many developments over its history, both in terms of finite element technology itself, and in terms of software engineering. There are a multitude of different finite element types, with new ones being developed all the time. Then there are developments in the associated technologies such as equation solution, optimisation, mesh generation. As well as the standard finite element mesh approach, there are other variations such as meshless methods, generalised finite elements, isogeometric elements. This latter technology is being developed to enable better integration between finite elements and CAD programmes [1]. This is just one example of developments in the software itself. Finite element programs are all coupled with graphical user interfaces, and within the applications themselves there can be a variety of features, such as modules to link the results of finite element analysis to concrete reinforcement design. Computers themselves have advanced by a great amount, so everyday computers have large amounts of memory and multiple processors. Moreover, all computers are linked together via local intranets and the internet. So distributed and parallel computing are no longer specialist areas, but are completely mainstream. Indeed, with the increased interest in cloud computing distributed computing is likely to become even more relevant.

All these capabilities introduce additional complexity. Over the last 20 years or so a large number of researchers have studied the application of object-oriented programming methods to finite element software development. Most of this work has been carried out in C++, though other languages such as Java and C# have been used as well. While object oriented programming offers a number of advantages in handling complexity, it is no panacea and as class libraries increase in size problems can arise. The next development in programming was to use component oriented programming. This works together with the object oriented approach, but is better at isolating complexity.

The next step in programming development has been service-oriented architecture (SOA), and this paper will consider the relevance and application of SOA to finite element analysis. The work described herein used Windows Communication Foundation (WCF) and .NET, though the principles elucidated are generally applicable to SOA. Following a review of software engineering development in finite elements, the ideas of SOA will be introduced using equation solvers. Then the application of the SOA approach to finite element software design will be looked at in detail.

Any software development needs to be assessed for its impact in the following areas:

- Speed of execution.
- Program design.
- Software capabilities.

The application of SOA will be assessed in each of these areas, in particular the opportunities it gives for enhancing and increasing the use of finite element analysis.

2. Literature review

It is over 20 years since object oriented programming was first applied to finite element analysis. The early papers focused on

* Tel.: +44 1382 384702; fax +44 1382 384389.
E-mail address: r.i.mackie@dundee.ac.uk

0965-9978/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2012.07.002

straightforward implementations of finite elements in an object oriented language [2,3], with separate classes being created for degrees of freedom, nodes, elements, etc. One of the goals of this work was to produce a new generation of tools to develop easily maintainable and extensible finite element software systems [4]. Initially there were concerns over the computational efficiency of object-oriented programs. Part of this concern was a result of some of the early work being carried out using Smalltalk. However, the move to C++ largely overcame these objections [5]. While C++ continues to be widely used, other languages and environments, like Java and C# have been used. These languages use just-in-time (JIT) compilers and can be a little slower than C++. However, Nikishkov et al. [6] have reported favourably on the computation speed of Java, and Mackie [7,8] has reported that the speed costs of using C# compared to C++ are limited. The primary advantage afforded by object-orientation (OO) is its richer data modelling capabilities compared to traditional languages. Accordingly, OO has been applied to a number of complex algorithms [9–12].

While object-orientation yields many advantages [13], it has limitations in dealing with complexity. This becomes particularly apparent as class libraries increase in size. Accordingly object-orientation has been complemented with component oriented design [14]. OO relies heavily on inheritance and polymorphism, component oriented design uses interfaces and object composition [15]. This tends to lead to more flexible designs and smaller class hierarchies. Dolenc [16] applied the component oriented approach to finite element software. Both Java and the .NET framework allow, and indeed encourage, component oriented software design.

The reason that component orientation presents a step forward in terms of software design is that it facilitates the logical separation of areas of complexity. The need to do this was recognised quite early on in the application of OO to finite elements. Archer et al. [17] sought to achieve flexibility and extendibility and a key step in achieving this goal was the separation of tasks, using distinct objects to carry out separate tasks. Dubois-Pelerin and Pegon [10] used a separate class for each problem type and this class worked on a domain class. Doing this separated out the problem specific aspects from those related to the domain (which represented the discretised model). Patzak and Bittnar [18] describe the programming principles behind their finite element framework, again a crucial part of the design was the separation of tasks.

Perhaps the most important developments in the computer architecture on which programs run now is that most computers are multi-core, and all computers are linked together on intranets and the internet. This means that parallel and distributed computing is now no longer a specialised activity, but is an integral part of mainstream computing [19]. The most widely used technologies are MPI [20] and OpenMP [21]. These are fairly low-level systems and this is a disadvantage in terms of software engineering. However, there are a number of interfaces from object-oriented systems like Java and .NET for MPI [22–25]. The .NET framework was developed specifically with distributed computing in mind and the latest version includes the Task Parallel Library (TPL) [26]. This facilitates parallel programming, enabling the programmer to focus on the tasks to be carried out, rather than the threads on which they run. This emphasis on tasks is reflected in work in other technologies such as OpenMP [27] and Java [28]. Distributed computing is relevant to using clusters of computers and to internet based computing [29–32]. Work by the current author has shown how component oriented programming and assist with the design and deployment of distributed numerical applications [33].

Just as the component oriented approach complimented the object-oriented approach, and addresses some of the short-comings, service-oriented-architecture (SOA) is the next step in development. According to [34] the four tenets of SOA are:

- Service boundaries are explicit.
- Services are autonomous.
- Services share operational contracts and data schema, not type-specific metadata.
- Services are compatible based on policy.

This means that services are exposed purely as interfaces, should reveal nothing about implementation detail and should be completely self-describing. The next section will describe how this works in practice for Windows Communication Foundation (WCF). The outcome is that clients should be able to use a service without being concerned about its location or implementation. So the service, as far as the client is concerned, is entirely described by the operations contract (which defines what the service can do) and the data contracts (which define the data required). All this is aimed at minimising coupling between client and service.

Now since its inception, .NET has had remote objects, and these can be accessed through interfaces. Indeed, this is generally the best way to expose a remote object. So in some ways WCF formalises this use of interfaces. However, the key difference is that in remoting remote objects are treated just like local objects. This has advantages, but also drawbacks. The drawbacks include the fact that with remote objects issues such as lifecycle management, security and reliability become important. WCF is designed to handle these and other issues. With remote objects, the client is using the interface to manipulate an object that might exist on a remote computer. In WCF the client is accessing a service only. The service has full control over what happens the other side of the boundary. The interface is a contract of what the service will provide. While WCF is a .NET technology, WCF services can be accessed from other environments. So a Java program running under Linux can access a WCF service running under Windows.

WCF is a technology with its roots in business computing, so one might ask how relevant is this to engineering computing? There are two answers to this. First, at the most basic level a distributed computing technology only needs to enable computation to be executed on another computer and to transfer and receive the associated data. This is essentially what MPI does. Remoting allows this to be done for .NET. WCF also allows this to be done, as do various Java technologies. So there is no inherent reason for not using something like WCF. What matters is how effectively it can do this task in terms of engineering computing. Secondly, issues such as reliability and security need to be considered. WCF is a complete framework and takes account of these matters. Engineering design and analysis is an increasingly global and distributed exercise. So there are good reasons for at least considering the applicability of WCF.

WCF is relatively new, and since its roots are more in general business computing there has so far been limited use of WCF in scientific or engineering computing. Exceptions have been in the area of distributed information systems. One example is work by Chang et al. [35] who used WCF on distributed 3D-GIS . They cited the advantage of WCF being that it brought together several communication mechanisms, including .NET, DCOM, Message Queuing and Web Services. Chengping et al. [36] have done work on the application WCF in the water industry, and Yang et al. [37] have used it for borehole logging data obtained from geological investigations. Stopper and Gastermann [38,39] have worked on the use of WCF in information systems in the manufacturing environment.