Scenarios for the deployment of distributed engineering applications

K. Lehner, Institute of Computational Engineering, Ruhr-University Bochum
(Karlheinz.Lehner@rub.de)

D. Hartmann, Institute of Computational Engineering, Ruhr-University Bochum
(hartus@inf.bi.rub.de)

Summary
Although there are some good reasons to design engineering software as a stand-alone application for a single computer, there are also numerous possibilities for creating distributed engineering applications, in particular using the Internet. This paper presents some typical scenarios how engineering applications can benefit from including network capabilities. Also, some examples of Internet-based engineering applications are discussed to show how the concepts presented can be implemented.

1 Introduction
The widespread availability of today's Internet ranges from the high-speed connections of academic institutions and research sites to power users and small firms with broadband access and flat rates down to an increasing number of mobile devices such as PDA's with WLAN and cellular phones. With this background of universal network accessibility, the value, necessity and potential of developing engineering software as distributed applications and the resulting impact of these developments needs to be reconsidered.

Of course, there are many situations where engineering software need not be designed as a distributed application at all. For example, standard software such as an office program or an application for a CAD or structural analysis problem can usually be run with complete satisfaction on an up-to-date PC with adequate hardware resources to do the job. Therefore, once the necessary software has been procured, installed and configured, it remains readily available for everyday use until a substantial patch or an useful update is needed.

There are, however, many cases where there is a distinct advantage of designing software as a distributed application or even situations where the software must be modeled as a distributed application to function at all. Some typical scenarios are given in the following sections.

2 Scenarios of Distributed Engineering Applications

2.1 Providing engineering know-how
Software often contains the know-how and expertise of a firm in executable form. Thus, a company would like its customers to use the company's know-how for a price, but of course it is reluctant to divulge any information, especially to competitors. So, rather than relying on more or less fool-proof techniques or devices to protect software at the customer site, an Internet-based application can let the server component (containing the company know-how) remain at the owner's site and under the owner's complete control. The server can then be accessed by client components over the Internet designed to provide the server with input data and to present results to the user.

2.2 Open source development

At the other extreme, creating open source software thrives on the network-wide interaction of many participants to develop, extend and service applications. The general idea of open source is that programmers can read, modify and redistribute code to enable software projects to evolve and improve. Thus, apart from having no direct licensing costs or restrictive licensing restrictions, open source should be (at least in theory) relatively up-to-date and fairly stable, because of the numerous opportunities for many people to directly view program sources.

Successful open source projects include such applications as Linux, OpenBSD and other operating systems, the Apache web server, and the Eclipse software development environment. There are also some open source projects related to engineering problems such as the FElt (FElt 2000) and Felyx (Felyx 2001) to do finite element analysis or OpenSees (OpenSees 2004) to simulate the performance of structural and geotechnical systems subjected to earthquakes.

Although the point of developing open source software precludes an individual or company from gaining revenues by directly selling, leasing or licensing software, it can still be a profitable business to provide support and consulting for an open source project including distribution and packaging services. Because software, in contrast to manpower and knowledge, can be easily duplicated, this explains how firms can, for example, continue to offer offer Linux software on a commercial scale, while the operating system itself is freely available. It remains to be seen how engineering applications can also be developed on a commercially successful scale as other open source projects have shown.

2.3 Distributed project management

In large projects involving a multitude of often highly mobile specialists, an efficient project management can play an important role in keeping the communication between partners at a satisfactory level. Managing data of all sorts, including engineering specific data such as construction plans and computational results, requires a network-based solution with general access. In this scenario, the emerging agent technology already used in computer science which relies on mobile code to be carried out on foreign participating computers must, by definition, be implemented within an architecture containing distributed components.

2.4 On-demand computing

Although the hardware capabilities of typical personal computers of today are largely adequate to handle many everyday engineering problems, there may arise situations which require a great deal more computer resources than is readily available and maybe for only a short period. If a company, for example, would like to carry out a large scale simulation or solve a complex structural optimization problem, then it could possibly make sense to outsource the needed hardware and software requirements to a provider with the necessary resources. For example, assuming the required scalability is given, a problem that might require 24 hours of CPU time on a typical PC could be done in a 15 min. coffee break on a computer system with 128 processors. Extending this concept of “computing power at your fingertips” further leads to the introduction of the grid concept, which proposes the availability of computing power from a standard source which is always readily available, just as electricity from the power grid is always at one's disposal.

2.5 Flexible software components

Because client and server components in a distributed system communicate via well-defined interfaces, there is a greater amount of flexibility in the choice of hardware and operating system platforms for implementation. Thus, for example, we can implement the server

component of a structural analysis application using programming languages such as Fortran or C to do the number crunching work and, on the other hand, implement the client components in programming languages that readily support interactive graphical user interfaces. The use of low-end devices such as PDA's or even mobile telephones with Internet access and built-in Java virtual machines imply the complete separation of number crunching and displaying techniques.

2.6 Enhanced user access

Providing a server with an engineering application over the Internet to potential customers can reduce the amount of effort needed to actually use the software. For example, using a web browser familiar to many users allows a software to be used “on the fly”. Especially for testing or marketing purposes, one can allow the use of a simplified or restricted version of a software to be used directly on-line, which can hopefully entice a potential user. Also, because the software is available from the provider on-line, updates (including bug fixes) can be carried out quickly and often transparently. As an example, the Java Web Start Technology can be mentioned here.

2.7 Peer-to-peer architecture

The concept of a peer-to-peer architecture for cooperating systems is usually associated with computers sharing files of various (and often dubious) nature across the Internet. However, as a paradigm, peer-to-peer basically represents an extreme form of client-server computing where the server component in the architecture is greatly reduced or even completely eliminated. As the name suggests, in a peer-to-peer network, all the nodes in the network function as client and server components with equal standings. Any node is able to initiate or complete a transaction. There is not central command (server) to delegate or coordinate the individual peers (although there may be an auxiliary server for boot-strapping purposes).

The management of peer-to-peer networks includes additional software techniques such as distributed hash tables not found in typical client-server applications because of the independent nature of peer nodes. This also implies that security measures for reliable and secure information processing must be included in the design of peer-to-peer networks. For example, with the increased use of mobile devices (such as mobile phones, PDA's or wireless notebooks), the ad-hoc creation of spontaneous networks becomes an important issue. Engineers meeting at a construction site will expect to be able to share current information in a simple and straightforward fashion. Also, the agent technology mentioned above is another example which relies on the highly decentralized network platform of peer-to-peer systems. Software programming environments such as JXTA and JNGI allow the creation of decentralized components to create peer-to-peer engineering applications.

3 Examples

In this section, some typical engineering applications developed at the Institute of Computational Engineering will be presented that demonstrate some of the concepts discussed in the previous section.

3.1 A Distributed FE-Application with Technical Documentation

For the analysis of structural systems using standard finite element methods, the distributed engineering application *caFE* (Althoff 1998) has been developed at the Institute of Computational Engineering using the CORBA (Common Object Request Broker Architecture) technology (see Fig. 1).

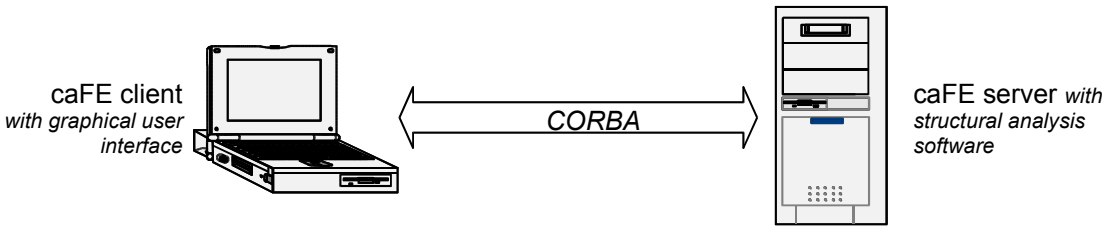

Figure 1. The distributed caFE software

In the basic client/server architecture, the client component of the distributed application has an interactive, graphical user interface. Its purpose is to allow the user to input all the data needed to model a structural system, to initiate the analysis in the server component and to present the results of the calculations as an interactive 3D model. Here, the 3D visualization in the client is implemented using the VTK toolkit software. On the server side, a structural analysis using the FElt FE-library allows the calculation of geometrical properties, such as translations and rotations, as well as the internal forces of the structural elements defined in the FElt package.

Because the structural analysis algorithms (in this case, the FElt library) are completely contained in the server component, this is a typical example of providing engineering know how to an end user and also an example of possible on-demand computing, given the proper hardware platform for the server component. Further, this prototype has been augmented with specific web services to include technical documentation output. This is because that, although the 3D representation of a structural model can easily let the user spot apparent input errors or let him get an intuitive feel of the structural response, the needs of accompanying documentation must not be neglected. In particular, there must be support to create printed technical documentation or documentation suitable for viewing in a browser. To this end, the client component needs to be extended to provide such technical documentation. However, in order not to increase the complexity of the client component, the generation of project documentation was identified as a separate process and implemented as an external web service (Lehner, Baitsch and Hartmann, 2003).

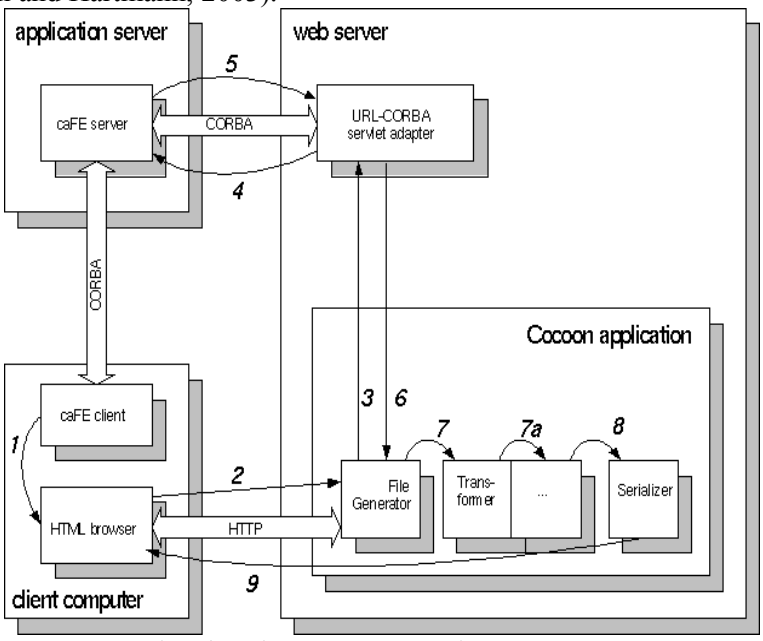

Figure 2. A distributed engineering application using Cocoon as a XSL transformer to create PDF documentation. For details on the data flow, see(Lehner, Baitsch and Hartmann, 2003).

The process of creating technical documentation can be generally defined as transforming a set of structured input data according to specific rules to generate the desired output data. The input data needed to create the documentation can described using a suitable XML representation containing all the information of the structural model as input by the user as well as the results of a structural analysis. The transformation of the generated XML data can be implemented using XSLT (eXtensible Stylesheet Language) Transformations. XSLT is basically a functional programing language designed for the efficient transformation of XML structures. The output data, i.e. the desired technical documents, must be suitably presented to the user. In order to keep the client component simple, the output data can be displayed using standard HTML browsers, containing not only HTML code, but possibly also vector graphics such as SVG (Scalable Vector Graphics) and PDF (Portable Document Files), suitable for printed documentation.

The transformation of XML input data with XSL transformation files can be implemented using the Cocoon web service from the Apache project. In particular, Cocoon can be set up as a publishing framework servlet, i.e. the XSL transformations are carried out on the web server that accepts URL's encoding the type of transformation requested along with the necessary XML data. By clearly identifying and separating data, logic and layout aspects, it is possible to generate different types of documents serving different needs from one XML source. In Fig. 2, the general architecture of the distributed application using Cocoon is shown. For details on the data flow, see (Lehner, Baitsch and Hartmann, 2003).

3.2 Distributed engineering applications using J2EE

The J2EE (Java 2 Platform, Enterprise Edition) technology is a programming environment for the development of client-server applications using the Java programming language. Although basically developed for the creation of e-commerce and other business applications, concepts such as multi-tiered applications and ready deployment of client components lend themselves to engineering software as well. In (Wang 2002) and (Lehner 2002) there is description of a prototype engineering software for the structural analysis using the J2EE platform. It contains a Java implement ion of a FE package and provides various web services to access the FE server.

In the simplest case, a distributed application consists of a client and a server component, often called a 2-tier architecture. The client is the component of the application which directly communicated with the user. The server component contains the so-called business logic or, in engineering parlance, the computational logic which does the actual numerical calculation. But apart from purely doing computations or data processing, it has been recognized that the efficient storage of data on the server side is also becoming an important issue. For example, the management of project data in persistent storage is important for long-term engineering applications. Therefore, the inclusion of standard interfaces and data base components in the server has extended the server side to include an additional tier called EIS (Enterprise Information Service) in J2EE. Thus, a distributed application in J2EE consists of a client, a middle and an EIS tier.

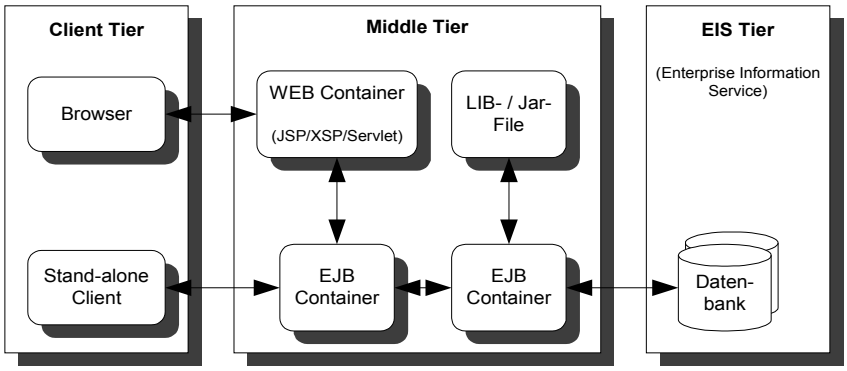

Figure 3. A typical J2EE 3-tier application

Web services are realized in the middle tier by providing so-call Enterprise JavaBean Containers (EJB). Each EJB is associated with an interface which describes which services the component provides to a client (or other server) components. The container itself must, of course, implement the corresponding services. Also, components of the middle tier can include additional software such as libraries (Java JAR files), such as the FE package mentioned in the application above. Finally, the middle tier can include so-called web containers which can provide dynamically generated HTML pages or include Servlets or JSP/XSP (Java Server Pages, XML Server Pages) components. This allows clients components to access the server using standard web-based mechanisms known to most users.

3.3 Multi-Agent Systems

Agent systems, in particular Multi-Agent Systems (MAS) can be used to model the holistic nature of concurrent processes found in many structural engineering projects. In (Bilek, Mittrup, Smarsly and Hartmann. 2003) two examples are given that show how MAS can be used to incorporate the distributed and interactive nature of structural design and monitoring of engineering structures. In the first example, a reference system, a steel bridge constructed over the river Mulde in the city of Dessau, Germany, was analyzed according to structural and work flow aspects. For example, the documents on the work flow reported that the project started with a simplified digital model of the bridge, then advanced to a detailed structural analysis, where a qualitative examination showed that the dynamic behavior of the bridge was unacceptable and had to be redesigned. In this iterative process, all the members in the project (including consulting engineers, a steel and concrete company, structural designers and city authorities) has to be informed on the changes. To facilitate this interdependence of participants, a MAS system has been set up where so-call personal cooperative agents represent individual project participants. While in the real world the project is coordinated and governed by human managers, additional non-personal project agents have been created in accordance with the other task agents. The project agent allows for the collaboration by providing specified interfaces to the personal agents.

Also, to model the bridge itself within the MAS, product model agents (PMA) have been include which can communicate technical details of the structural system to the corresponding personal agents. In particular, in this project the product model agents include a PMA for the overall architecture, a PMA for the bridge deck, two PMA for the abutments and a supervisory PMA that has knowledge about the dependencies between the other structural components.

3.4 Web-based monitoring of dams

Dams are large engineering structures exposed to various environmental factors and have to be continually monitored to assure standards of operational safety and reliability. Because of the large distances involved, the frequent access of data at a site can be time consuming and at times difficult. Having remote data access over network connections is therefore desirable. In (Smarsly, Mittrup, Hartmann and Bettzieche 2003) an implementation of a web based monitoring system for dams is presented. In this scenario, data loggers regularly collect data from numerous sensors (to measure temperature, water leachate, displacement or pore water) positioned across the dam and store the raw input locally. The remote administration of the system, including the logger and sensor components, is possible by interfacing the control computer to an available network. This allows an administrator with the proper credentials to remotely check and configure the system on a routine basis and requires an on-site inspection much less often.

The data that is routinely collected is evaluated by a variety of specialist, each needing a different view on the data according to their expertise. For this purpose, and also to reduce the amount of data traffic, the systems has a web-based visualization and reporting component that selectively processes data according to predetermined configurations.

3.5 Optimization and reliability analysis server

As another example of distributed engineering applications, a CORBA-based system for structural optimization is presented. The system consists of individual components (structural analysis component, optimization components, clients with graphical user interfaces) that use CORBA as the middle ware. The structural analysis component can be a FE program (FElt) wrapped as a component or a Java library (Mini-FE) bound to the client component. The optimization component (Baitsch, Lehner and Hartmann, 1999) provides a variety of optimization methods, ranging from gradient-free search methods such as evolution strategies to gradient-based methods such as sequential quadratic programming. As experience shows, there is no single numerical optimization method that works well for all optimization problems, so it important to have a variety of methods available that can be used interchangeably. To this end, the optimization server therefore contains a collection of optimization methods, each of which can be accessed with the same (CORBA-based) programming interface.

Finally, an example of a distributed engineering application specifically designed to perform reliability analysis is given in (Spitzlei 2003) and (Ballnus 2003). Normally, a designer uses a deterministic method to analyze a design. In many cases this method is satisfactory. However, in some cases this design method is not adequate, because it results in a system design that is to expensive or not reliable enough. In these cases a probabilistic approach or risk analysis may be helpful. Risk analysis experts support the designers to guarantee the costs and safety of complex systems. They analyze whether the system can perform its major functions, such as carrying load and executing motion, using a probabilistic model. Both the load carrying capacity of a system as well as the external loads can show a stochastic behavior. Thus, this system contains a server component with numerical methods for probabilistic analysis and a client component to help model stochastic FE-structures.

3 Conclusion

With the rise in programming environments where network access is considered a standard feature, the potential to create new and innovative engineering applications must be explored. Standard engineering software can be augmented with additional services and functionality which allows developers to create applications that were not even considered some years ago.

Of course, users must also accept the new paradigms and the value-related benefits. This paper explores some aspects specifically related to the purpose of designing and creating distributed engineering applications. Some representative example developments are presented which are to show how typical engineering applications can be implemented using Internet-enabled engineering.

4 References

Althoff. C. (1998). *Entwicklung eines objektorientierten Strukturanalyse-Servers auf der Basis der CORBA-Technologie unter Verwendung der Finite Element Bibliothek FElt,* diploma thesis, Ruhr-University Bochum, Dec. 1998

Baitsch, M., Lehner, K. and Hartmann, D. (1999). *Ein CORBA-basierter universeller Optimierungsservice,* in: 1st ASMO UK/ISSMO Conf. on Engineering Design Optimization, pp. 233-240, Ilkley, GB, 1999

Ballnus, D. (2003). *Entwicklung eines CORBA-Servers für die Zuverlässigkeitsanalyse,* diploma thesis, Ruhr-University Bochum, Sept. 2003

Bilek, J., Mittrup, I., Smarsly, K. and Hartmann, D. (2003). *Agent-based concepts for the holistic modeling of concurrent processes in structural engineering,* in: 10th ISPE International Conference On Concurrent Engineering Research and Applications, Madeira, July 26-30, 2003

FElt (2000). *The FElt Demo Document*, http://felt.sourceforge.net/, Feb. 22, 2000

Felyx (2001). Koenig, O, Wintermantel, M., Zehnder, N.; FELyX - *The Finite Element Library Experiment*, http://felyx.sourceforge.net/, Nov. 11, 2001

Lehner, K. (2002). *Entwicklung einer verteilten Ingenieurapplikation unter Verwendung der J2EE-Technologie*, Forum Bauinformatik 2002, Bilek, J. (ed.), University of Bochum, Sept. 16-18, 2002

Lehner K., Baitsch M. and Hartmann D.; (2003). Using XSL Web Services for Project Documentation in an Engineering Application, in: Intelligent Computing in Engineering (Ciftcioglu, Ö., Dada, D., eds.), Proceedings of the 10th European Group for Intelligent Computing in Engineering Workshop, Delft, July 3-4, 2003

OpenSees (2004); Pacific Earthquake Engineering Research Center, *Open System for Earthquake Engineering Simulation - Home Page*, http://opensees.berkeley.edu/index.html, 2004

Smarsly, K., Mittrup, I., Hartmann, D. and Bettzieche, V. (2003). *Implementierung eines webbasierten Talsperren-Monitoring-Systems,* in: Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen 2003, IKM 2003, Weimar, June 16, 2003

Spitzlei, K. (2003). *Entwicklung eines CORBA-Servers für stochastische Finite-Elemente Berechnungen in Zuverlässigkeitsanalysen,* diploma thesis, Ruhr-University Bochum, Sept. 2003

Wang, H. (2002). *Development of a distributed engineering application using J2EE technology*, master thesis, Institute of Computational Engineering, University of Bochum. 2002