Engineering design optimization using services and workflows

BY TOM CRICK¹, PETER DUNNING², HYUN SUN KIM²
AND JULIAN PADGET¹,*

¹Department of Computer Science, and ²Department of Mechanical Engineering,
University of Bath, Bath BA2 7AY, UK

Multi-disciplinary design optimization (MDO) is the process whereby the often conflicting requirements of the different disciplines to the engineering design process attempts to converge upon a description that represents an acceptable compromise in the design space. We present a simple demonstrator of a flexible workflow framework for engineering design optimization using an e-Science tool. This paper provides a concise introduction to MDO, complemented by a summary of the related tools and techniques developed under the umbrella of the UK e-Science programme that we have explored in support of the engineering process. The main contributions of this paper are: (i) a description of the optimization workflow that has been developed in the TAVERNA workbench, (ii) a demonstrator of a structural optimization process with a range of tool options using common benchmark problems, (iii) some reflections on the experience of software engineering meeting mechanical engineering, and (iv) an indicative discussion on the feasibility of a ‘plug-and-play’ engineering environment for analysis and design.

Keywords: multi-disciplinary design optimization; Web services; workflows; semantic web

1. Introduction

(a) The engineering perspective

Complex engineering design is realized through the combined efforts of a number of specialist design teams with discipline-specific skills, tools and knowledge. The competitive environment of the engineering industry demands an optimum design through continuous improvement in performance and economy of design. However, due to the interdependent nature of disciplines, achieving an optimal design can be difficult and slow as each specialist team often lacks a direct understanding of the global consequences.

Multi-disciplinary design optimization (MDO) has been defined as ‘methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines’ (Sobieszczanski-Sobieski & Haftka 1997). Research in MDO has received

* Author for correspondence (jap@cs.bath.ac.uk).

One contribution of 16 to a Theme Issue ‘Crossing boundaries: computational science, e-Science and global e-Infrastructure II. Selected papers from the UK e-Science All Hands Meeting 2008’.
increasing interest as it aims to reconcile potential conflicts by treating the design as a whole, taking account of the interdependency of design disciplines. Two categories of methods have emerged: single- and multi-level formulations. The former typically uses a single optimizer which is applied to the entire MDO system. While this approach may be considered the more intuitive, the typical size and highly complex nature of the problems are usually unsuitable for a single optimizer. Conversely, the multi-level approach decomposes a design problem. Both analysis and optimization are carried out for each subsystem and the interactions between subsystems are considered by a system-level optimizer to determine the solution (Martins & Marriage 2007; Yi et al. 2008).

In the modern engineering environment, the multi-level formulation is often the preferred approach. There are several reasons for this, but two key factors are: (i) the decomposition of a large problem into a number of subsystem optimizations is naturally suited to parallel and distributed computing, and this can significantly reduce computational costs to realistic levels; and (ii) as the structure of engineering organizations typically reflects disciplines and each discipline team works largely independently, the multi-level approach is somewhat akin to the current design industry, and hence more suited to industrial practice.

The multi-level formulation considers the highly coupled interdependency of multi-disciplinary design criteria at system level. Therefore, it follows that the optimum solution can be sensitive to the system architecture and implementation (Alexandrov & Kodyalam 1998; Brown & Olds 2006). There have been limited studies that provide a generalized understanding of various methods and their performance, and the MDO implementations have been specific to design problems and applications. As the design evolves and the tools are modified, it requires a significant modification to the MDO framework and the suitability of the modified architecture for the given problem cannot be guaranteed.

Additional difficulties arise in the changing environment of engineering industry, where designs are increasingly carried out in various geographical locations on heterogeneous platforms. This calls for a more flexible and easy-to-understand MDO system for integrating both legacy codes and proprietary software with a range of model representations and fidelity (Giesing & Barthelemy 1998).

(b) From e-Science to e-Engineering

A key aim of e-Science activity has been to ease accessibility to data and computational resources, wherever they might be, and however they might be described. In technological terms this translates to workflow design and enactment on the one hand, and service description and discovery on the other. Active research initiatives on service discovery can be seen particularly in the chemistry and bio-informatics communities, e.g. BioMOBY at www.biomoby.org offers interoperability between biological data hosts and analytical services, BIOCATALOGUE at www.biocatalogue.org provides a curated catalogue of Life Science Web Services. The MYGRID project at www.mygrid.org.uk has demonstrated how bio-informatics research can be assisted (Oinn et al. 2004) through the automated identification of Web services and a visual programming environment—the TAVERNA toolkit (Oinn et al. 2006)—for the construction and execution of workflows. Association of semantic information with services
and its subsequent discovery is being enabled by tools such as the semantics-enabled extension of UDDI, GRIMOIRE (Fang et al. 2008), and the general purpose, extensible brokerage framework, KNOOGLE (Chapman et al. 2007)—both OMII projects (see www.omii.ac.uk)—while Seekda at www.seekda.com offers a text-based searcher for Web services and the FETA (Lord et al. 2005) component of TAVERNA is developing a semantic search interface. In previous work, we made some steps towards the semantic description of mathematical services (Caprotti et al. 2004), which is particularly relevant for engineering, but in many fields effective domain-specific description is an emerging topic.

The objective of the work reported here is to investigate the applicability of the TAVERNA workbench to a classical engineering optimization problem. We chose to use TAVERNA because it has been used for a diverse range of domains including medicine, astronomy, social science and music, as well as bio-informatics. It is clear from the degree of up-take that the use of tools to support the authoring and enactment of workflows has been beneficial to many different communities by (i) enabling access at a distance to resources, and (ii) allowing the researcher to focus on how to combine those resources. There are many parallels to be found in the engineering design process: (i) workflows capture common, even standardized processes, (ii) resources are not necessarily co-located, and (iii) much tedious effort is spent copying and transforming data output by one analysis as input for another analysis. It is notable that commercial software packages are starting to exploit the workflow concept in their interfaces but, by being essentially closed systems, have the effect of locking users into particular products.

By building a system based on workflow, we are able to delegate file manipulation and house-keeping details to the enactment engine. Furthermore, from an engineering design perspective, it is now feasible to deploy many more analysis codes and with equal ease integrate them into workflows and visualize the results. We now proceed to discuss the proof-of-concept demonstrator.

2. Services and workflows

(a) Structural optimization algorithm

The design problem considered for the demonstrator study is a material distribution problem in a continuum such as figure 1. The continuum domain is represented by Ω enclosed by a boundary Γ, with body forces f, the boundary traction t on Γ, and support on Γu. The design variable x represents the existence of material which is either present or absent, x ∈ {0, 1}, as shown in figure 1.

The formulation of the optimization problem is to minimize the total compliance subject to the equilibrium and volume constraints. The implementation typically employs the existence of finite elements as design variables. A common approach to this discrete problem is to relax the design variables to 0 < x ≤ 1, but to penalize the intermediate values by power-law (Bensøe 1995). The optimization problem can therefore be written as

\[ \min C(x) = \sum_i x_i^p u_i^T k u_i \text{ subject to } KU = F \text{ and } V(x) \leq V_0, \]

where C denotes total compliance; u_i and k are the elemental displacement and stiffness, respectively; p is the penalization power; K is the global stiffness.
The optimization problem can be characterized by several parameters: (i) structural, i.e. the material properties, boundary conditions, loading and design domain size, and (ii) optimization, i.e. the penalization power, volume fraction and convergence criteria.

Optimization begins by discretizing the design domain continuum using finite elements and the design variables \( x \) are the continuous variation of the existence of an element, sometimes referred to as artificial density of an element. A finite-element analysis is usually employed to compute the nodal displacements, which in turn are used to determine the sensitivities required for optimization. The optimization uses the method of moving asymptotes (MMA) optimizer (Svanberg 1987), which works by updating the elemental artificial densities. This process is repeated until the convergence criterion is met. This process is formalized by algorithm 1.

(b) Workflow construction

TAVERNA can discover and use both local programs and deployed Web services as components in workflows. All the components reported here were published as Web services and are thus potentially re-usable by others. A TAVERNA workflow that implements algorithm 1 is shown in figure 2. The algorithm and the workflow evolved iteratively through a process of collaborative authoring, each functioning as a boundary object between the domains of the participants, and resulting in the identification of five major components for the workflow: (i) design initialization, (ii) analysis of the current design, (iii) sensitivity analysis, (iv) optimization and design update, and (v) the convergence test. For the analysis step, three finite-element analysis programs were available:

(i) Computational finite element (CFE), an ‘in-house’ legacy C code which was designed specifically to undertake the analysis step of the optimization process. This is deployed as a Web service using the cSOAP toolkit (van Engelen & Gallivan 2002) and demonstrates the capacity to publish C/C++/Fortran codes as Web services.
Algorithm 1. The structural optimization process

\begin{algorithmic}
  \State \textbf{input}: a problem independent component PI
  \State \textbf{input}: a problem dependent component PD
  \State \textbf{output}: optimization process control parameters PO
  \State \textbf{var}: the value of PD at step \( i \), denoted PD\(_i\)
  \State \textbf{var}: the value of PD at step \( i-1 \), denoted PD\(_{i-1}\)
  \State \textbf{var}: the value of PD at step \( i-2 \), denoted PD\(_{i-2}\)
  \State \textbf{var}: the strain energy of the model, denoted SE
  \State 1 PD\(_i\) \leftarrow \text{Setup} (PD, PO)
  \State 2 PD\(_{i-1}\) \leftarrow PD\(_i\)
  \State 3 PD\(_{i-2}\) \leftarrow PD\(_i\)
  \State 4 repeat
  \State 5 SE \leftarrow \text{Analyse} (PI, PD\(_i\), PO)
  \State 6 \frac{dSE}{dx} \leftarrow \text{Sensitivity-analysis} (PO, SE, PD\(_i\))
  \State 7 PD\(_{i-2}\) \leftarrow PD\(_{i-1}\)
  \State 8 PD\(_{i-1}\) \leftarrow PD\(_i\)
  \State 9 PD\(_i\) \leftarrow \text{Optimize} (PD\(_i\), PD\(_{i-1}\), PD\(_{i-2}\), PO, PI, \frac{dSE}{dx}, \frac{d^2SE}{dx^2})
  \State 10 until \text{Coverged?} (PD\(_i\), PD\(_{i-1}\), PO)
  \State 11 return PD\(_i\)
\end{algorithmic}

(ii) A commercial package, ANSYS, as an example of proprietary software with a license requirement (ANSYS 11.0SP1). To incorporate ANSYS into the workflow a macro was written in the ANSYS parametric development language that executes the required analysis. Additionally, a pre-processing step—implemented as a TAVERNA shim (see §4.4 of Oinn \textit{et al.} (2006)), i.e. a service whose purpose to carry out some minor operation to establish compatibility between two other services—converts the input data file into the format required by ANSYS. The interface is generated by the SOAPLAB2 (Senger \textit{et al.} 2003, 2008) Web service deployment tool and demonstrates the creation of services from command-line driven engineering analysis tools.

(iii) A Matlab script (Matlab r2003a). The script was specifically written to execute the required analysis. This service is also deployed using SOAPLAB2 and demonstrates the means to publish services built on widely used engineering scripting software.

All the analysis programs accepted the same input files with the same format and produced output files in a consistent format, making them completely interchangeable. Each analysis program was limited to a two-dimensional static linear elastic analysis of a rectangular domain of square elements of varying density. The inputs and outputs for all workflow components are summarized in table 1. All data are in plain text file format except for the convergence test output, which is a binary number.
3. Results

In this section, we use two popular structural benchmark problems and solve them using the optimization workflow described in §2b. The two problems are a short cantilever beam and a Messerschmidt–Bölkow–Blohm (MBB) aircraft floor beam (Bensøe 1995). Each structure was discretized with square elements of unit area. Both optimization problems were run using all available analysis programs and the optimization parameters were the same for both problems: $p=3$; $V_0=0.4$; convergence criterion $=0.01$.

Figure 3a depicts the structural design environment for the popular cantilever beam of aspect ratio 1.6, with one edge clamped and a central vertical load applied on the other side. The optimum solution was obtained after 42 iterations as shown in figure 3b–e. This is typical of the solutions obtained in existing literature, thus validating the optimization algorithm implemented as a workflow. The second test case is the MBB beam, which is a simply supported beam of aspect ratio 6 with a central vertical load. The results obtained were in each case the well-known optimum solution.

Phil. Trans. R. Soc. A (2009)
4. Discussion and related work

We have built and validated—using some well-known benchmark structures—a proof-of-concept workflow demonstrator that addresses or facilitates the points raised in Giesing & Barthelemy (1998)—specifically flexibility, provenance, multiple (consistent) models, distribution and resource brokerage—by

**Table 1.** Workflow component inputs and output.

<table>
<thead>
<tr>
<th>component</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>design variable initialization</td>
<td>structural parameters</td>
<td>initial element densities</td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td></td>
</tr>
<tr>
<td>FE analysis</td>
<td>structural parameters</td>
<td>element compliance</td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>element densities</td>
<td></td>
</tr>
<tr>
<td>sensitivity analysis</td>
<td>element densities</td>
<td>first-order derivative</td>
</tr>
<tr>
<td></td>
<td>element compliance</td>
<td>second-order derivative</td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td></td>
</tr>
<tr>
<td>optimizer (MMA)</td>
<td>structural parameters</td>
<td>current MMA variables</td>
</tr>
<tr>
<td></td>
<td>element densities</td>
<td>updated element densities</td>
</tr>
<tr>
<td></td>
<td>element compliance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>densities from previous two iterations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>element compliance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>first- and second-order derivatives</td>
<td></td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td>MMA variables from previous iteration</td>
</tr>
<tr>
<td>convergence test</td>
<td>updated element densities</td>
<td>convergence result</td>
</tr>
<tr>
<td></td>
<td>previous element densities</td>
<td>(1, converged; 0, continue)</td>
</tr>
<tr>
<td></td>
<td>optimization parameters</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Initial design domain of short cantilever beam (a) and its optimization at iteration 10 (b), iteration 20 (c), iteration 30 (d) and converged solution at iteration 42 (e).
redeploying tools conceived for e-Science, to enable more flexible and intuitive MDO processes that allow for: (i) the continuing use of favoured legacy code and prototype scripts as well as commercial software in a common framework, and (ii) the necessary variation in model representation and precision. It is interesting to observe that 10 years on from the above paper, Moore et al. (2008) make similar observations and call for the development of an open-source framework for MDO.

A notable difference between our work and several examples in the current literature on workflow in MDO is our use of an off-the-shelf framework, allowing for concentration on developing programs and wrappers, instead of developing the framework itself, as is the focus of Wang et al. (2003), Hao et al. (2004), Kim et al. (2006) and Lähr & Bletzinger (2007). Working with an existing community of (bio-informatics) users clearly underpins claims for usability and even longevity, as well as the capacity for more rapid growth, given an attractive set of services and workflows, in new domains.

Our planned next steps include evaluation of alternative workflow description languages and engines, the development of semantic descriptions of the components, and undertaking a wider range of case studies.

(a) Technical issues

Commercial software packages, such as ENGINEOUS, NOESIS and PHOENIX offer similar facilities for workflow creation, visualization and management although in each case the user is effectively limited to using the components provided with the package. The real benefit of using a workflow engine such as TAVERNA is the relative ease of using Web services, potentially leading to: (i) management and use of proprietary analysis tools by contractors, integrating them into the MDO workflow, minimizing risk of loss of intellectual property, reducing design time and improving quality control over (subcontracted) components, (ii) sharing ‘in-house’ codes with external users instead having to act as a software provider and maintainer, as well as allowing straightforward version control, and (iii) capacity for the automatic capture of component provenance information and potential for full life-cycle knowledge management.

We foresee a particular benefit arising from the use of Web services for the engineering research community. There has been significant effort into the development of alternative MDO methods, but few attempts at comprehensive comparison between methods. One reason is the amount of work required to develop and implement the sizeable range of available methods on a common platform, as most MDO methods for practical problems tend to be designed only for specific applications (Alexandrov & Kodiyalam 1998; Martins & Marriage 2007). A possible solution is the concurrent publication of the article and its implementation as a Web service—an approach along these lines has been implemented by the London Mathematical Society’s Journal of Computation and Mathematics for several years (e.g. http://www.lms.ac.uk/jcm/11/lms2007-056/). Some recent publications in the MDO literature have reached similar conclusions to ourselves about the desirability of workflow approaches (e.g. Shi et al. 2005; Bereneds et al. 2008; Moore et al. 2008), but we observe that the published descriptions appear to use bespoke software rather than workflow tools.
A notable drawback of accessing commercial analysis tools as services is that it bypasses much of their value-added functionality, such as post-processing and visualization. This factor is also identified by Oinn et al. (2006) in their extensive analysis of TAVERNA for life sciences applications and holds true for the engineering sector as well.

(b) Reflections on process

Bringing together engineering codes and workflow software has been a learning process for both parties. Apart from the initial challenge of appreciating and understanding each other’s vocabulary, there have been deeper-rooted issues around the advantages (or otherwise) of bringing in another layer of software technology and the adaptation of legacy code for the new environment.

There have been several situations during this work that might be thought of as ‘cultural’ issues, but, not being sociologists, these observations should be seen as purely anecdotal and without significant foundation. One aspect that surprised the computer scientists was the apparently low importance given to making software re-usable: modifications were proposed to make components work in the particular context of use, with relatively little consideration of new environments. In other words: aspects of engineering practice are now commonplace in computer science (software engineering), but these principles have not necessarily made their way back to software development in engineering.

Much of the literature on cultural issues in engineering has addressed ethnicity or organizational factors rather than domain discipline. However, Bond & Ricci (1992) explored the ways in which different disciplines work together in the context of aircraft design. Interestingly, the conclusions they reached resonate equally well with our experience of computer scientists working with mechanical engineers. We paraphrase and summarize their conclusions here and comment upon them: (i) an inter-disciplinary project proceeds through the cooperation of specialists, (ii) each specialist has its own model (or models) of the design for various purposes—we used a shared model (the algorithm) to communicate with one another, (iii) specialists have limited ability to understand each other’s models—we now each have a limited understanding of each other’s domain, (iv) design proceeds by successive refinement of the models, which are coordinated and updated together—indeed, we prototyped the workflow and the algorithm and revised and updated them together, and (v) the design decisions, which are acts of commitment and model refinement, are negotiated by the specialists among themselves.

Finally, over and above the practical issues identified in §4a, there is a qualitative aspect that is enabled by the adoption of Web services and workflow. Much of design optimization has been based on the parametric representation defined at the initial design stage. This restricts the solution space and prevents optimization methods from exploring all potential solutions. As the design matures and the scale increases to higher levels of detail, more refined analysis and optimization methods are required and results from the previous stages and legacy systems do not always translate well, both requiring many hours of manual process and potential loss of information. Previous design decisions in one discipline may be challenged as the problem is better understood, but the consequences of change for other disciplines are less understood, thus it is simpler...
to remain at the local optimum. Furthermore, the selection of codes and numerical tools leading to local attractors may as much be a function of economic and social factors as technical suitability. However, the accessibility of a wide range of codes, capture of provenance information and the ease of trying out alternative design avenues would ease the exploration of multiple design spaces, and offers the chance to make a notable step forward in MDO.

The authors would like to thank: (i) Prof. K. Svanberg for the implementation of the MMA algorithm, (ii) the Numerical Analysis Group at the Rutherford Appleton Laboratory for the Fortran 77 HSL package MA57, and (iii) Ian Dunlop and Stian Soland-Reyes of the MYGRID team for their help with TAVERNA via the TAVERNA-users mailing list. This work was partially funded by the EPSRC Bridging the Gaps programme, through the project ‘Creativity in Systems Design: the Bath Interactive Ideas Factory’ (EP/E018122/1).

References


Phil. Trans. R. Soc. A (2009)


