Engineering in the Cloud: An Engineering Software + Services Architecture Forged in Turbulent Times

Peter Williams[1], Simon Cox[2]
[1,2] dezineforce Co., United Kingdom

Abstract
This article describes both the architectural challenges that are inherent in implementing "Engineering in the Cloud" and an architecture that we call "Engineering Software + Services."

1. Introduction

Companies in all sectors are looking to cut costs, develop new revenue streams, and increase productivity. This is especially true in turbulent times. They are also looking to become more agile in a competitive landscape, reduce their time to market, create better products, and adhere to compliance requirements. This is particularly true in the engineering sector, in light of increasing globalization and the requirement to remain competitive.
This article describes the architectural challenges that are inherent in implementing "Engineering in the Cloud" and an architecture that we call "Engineering Software + Services." We illustrate this with a case study, drawn from dezineforce.com, and demonstrate how this architecture can deliver the functionality that is required by engineering companies who want to transform the way in which they work and prosper in turbulent times and beyond.

2. Engineering in the Cloud

Computationally-aided engineering has matured over many years, with computer simulations able to predict accurately the real-world characteristics of engineering designs. This can significantly reduce the need for physical testing that has traditionally been carried out at considerable expense. However, engineering companies are increasingly finding themselves having to acquire IT skills to manage and maintain these packages, as well as the complex infrastructures on which they execute.
Computation facilities in the Cloud could clearly provide many benefits. But would it really meet the needs of engineering companies, particularly in these turbulent times? It might reduce costs and make technology that had previously been available only to the largest of corporations available also to small and midsize companies. But how do we, as architects, harness the power of the Cloud to really transform the way in which engineering is conducted?
In order to affect a step change and meet the current needs of engineering companies, we believe that "Engineering in the Cloud" must also provide a level of intelligence and rich interaction that allows engineers to gain additional insights into their designs. Therefore, we define it thus:
Engineering in the cloud is a combination of cloud services and rich interactive applications that provides integrated, intelligent, self-service engineering services over and above engineering- application hosting and computation—allowing engineers to create, explore, and discover better designs faster.

3. Engineering Software + Services Architecture

Many publications exist on the challenges of building a Software + Services platform. The general concerns of security, availability, and reliability, among others, all apply as much to engineering as they do to other sectors. Instead of reiterating these more general concerns, we will focus on the less common challenges that are addressed by the "Engineering Software + Services Architecture" as a means of implementing an engineering cloud service.
There are five key, specific architectural challenges on which we will concentrate:

- Engineer interaction
- Engineering intelligence

- Engineering process orchestration
- Engineering computation
- Long-term, large-scale data management

The various subsystems in an engineering Software + Services architecture can be thought of as a number of layers and crosscutting concerns, as shown in Figure 1.

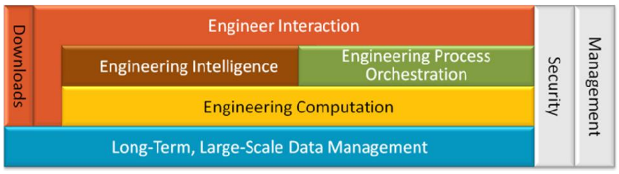

Figure 1. Generic high-level engineering Software + Services architecture

Security and management are not part of our set of five specific challenges, but they are briefly included for completeness. Within dezineforce, these challenges are implemented using the technologies shown in Figure 2.

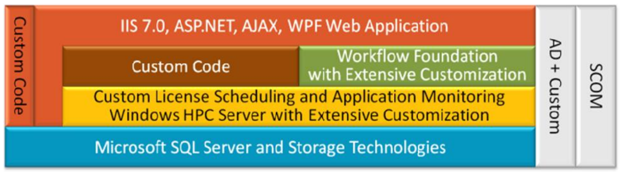
Custom Code
IIS 7.0, ASP.NET, AJAX, WPF Web Application
Custom Code
Workflow Foundation with Extensive Customization
Custom License Scheduling and Application Monitoring
Windows HPC Server with Extensive Customization
Microsoft SQL Server and Storage Technologies
AD + Custom
SCOM

Figure 2. The dezineforce high-level engineering Software + Services architecture

3.1 Engineer Interaction

Engineering processes might take hours, days, or weeks to complete, and engineers must remain in full control of the processes throughout their execution. Therefore, it is essential that they can see into the processes to understand the progress and current status of their exploration. This ranges from understanding which loop the process is in, right down to inspecting the individual output files of currently executing jobs. The ability to detect and react quickly to issues reduces wasted compute cycles and elapsed time, which reduces costs and improves time to market.

These processes create very large volumes of data, with tens of gigabytes for a single run being fairly normal; hundreds of gigabytes are not unheard of. In order to increase engineer productivity, the architecture must allow engineers to conduct intelligent and powerful searches and interrogate results without having to download the complete data set. The service must provide tools to allow engineers to view summaries of the data and relevant sections of the output files. These tools can be embedded in the service or provided as rich client tools.

Within the dezineforce service, engineer interaction is implemented using Microsoft Internet Information Services (IIS) 7.0, ASP.NET, and AJAX—providing a good experience, with broad reach. There is also a rich 3-D visualization tool that is implemented as a Windows Presentation Foundation (WPF) browser application to view and interact with the output of the optimization process. This technology was selected because it provides a rich, hardware-accelerated, interactive 3-D experience within the browser—without the need to install software explicitly—and was a good match for the existing Microsoft .NET skill sets of the team.

The use of WPF allows for rich rendering of the mathematical models by using lighting effects and camera positioning to see details that otherwise are difficult to bring out. For example, lighting effects are particularly effective in side-on views of complex surfaces, as shown in Figure 3.

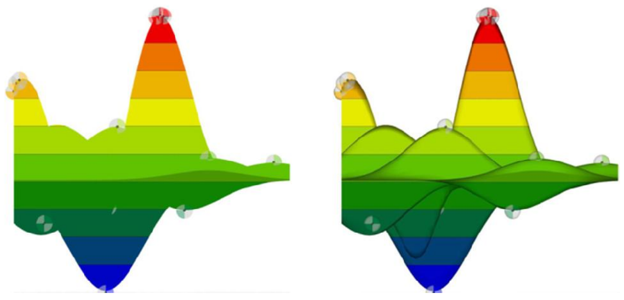

Figure 3. Ambient light (L) and directional light (R)

Because many design engineers already have high-performance CAD workstations, the ability to leverage their hardware capabilities allows for rich interaction with the models, while still providing high- quality visuals.

3.2 Engineering Intelligence

During the design process, an engineer is faced with a great deal of choices, requirements, and constraints that are often at odds with each other—requiring a set of trade-offs to be made.

As the number of variables increases, it is not feasible for engineers to explore all possible design options exhaustively. It is also unrealistic to expect them to discover counterintuitive designs without a thorough systematic investigation—particularly, on new types of design of which there is little prior industry knowledge.

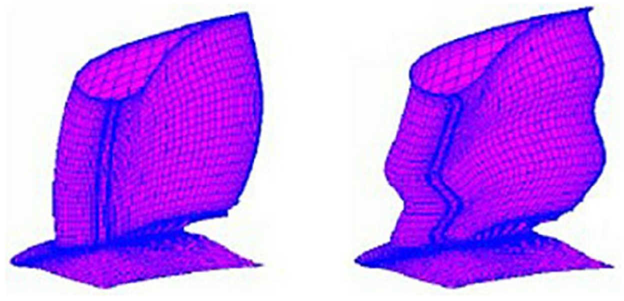

Figure 4. Intuitive guide vane (L) and non-intuitive (but superior) guide vane (R)

Figure 4 shows two guide vanes (situated at the inlet of a jet engine). The non-intuitive design is significantly better, but it is very unlikely to have been arrived at by designer judgment and hosted computation alone.

(These designs are the result of a study that was carried out for Rolls-Royce using the optimization toolkit that is incorporated in the dezineforce service. They are included courtesy of Professor Andy Keane [University of Southampton] and Rolls-Royce.)

Achieving these high-performance designs requires far more than just raw computation and automation. It requires a layer of engineering intelligence, while keeping engineers in control—guiding them through this maze in a systematic, efficient, and informative way.

This approach can reduce costs significantly, because it requires an order-of-magnitude fewer simulations. Given finite computing resources, this also reduces time to market. The ability to discover non-intuitive and counterintuitive designs leads to better products and allows engineering consultancies to offer new services to their clients—thereby, generating new revenue streams.

dezineforce provides a layer of intelligence through the use of advanced Design Search and Optimization (DSO) algorithms. This optimization suite was developed over many years and informs the process orchestration. A detailed explanation of these algorithms is beyond the scope of this article (see the "Resources" section for further reading).

The output of these computational methods can be used by an engineer to visualize the design trade-offs. In this real-world example of a simple problem, it can be seen that with very few simulations (indicated by the colored spheres in Figure 5), the optimization algorithms have predicted the shape of the surface and directed the process orchestration to focus new simulations where the better designs are to be found—in this case, near the lowest point.

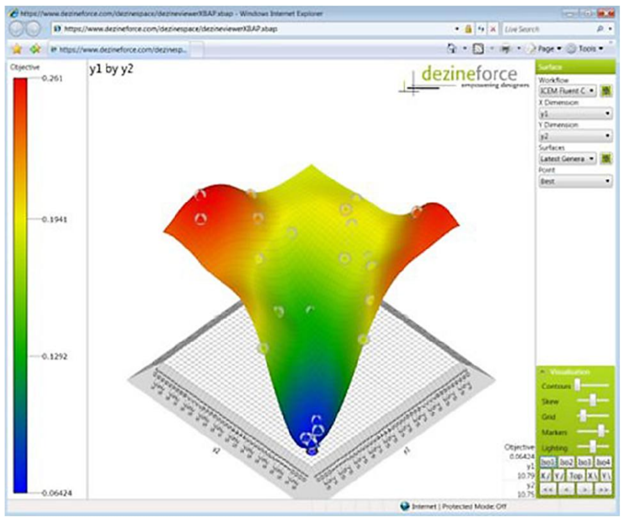

Figure 5. Surrogate model displayed in a WPF browser application

In the second example, which is shown in Figure 6, we use a complex mathematical function to provide a more challenging problem. Note how few simulations (colored spheres) have been carried out; yet the existence of peaks and troughs is predicted with remarkable accuracy.

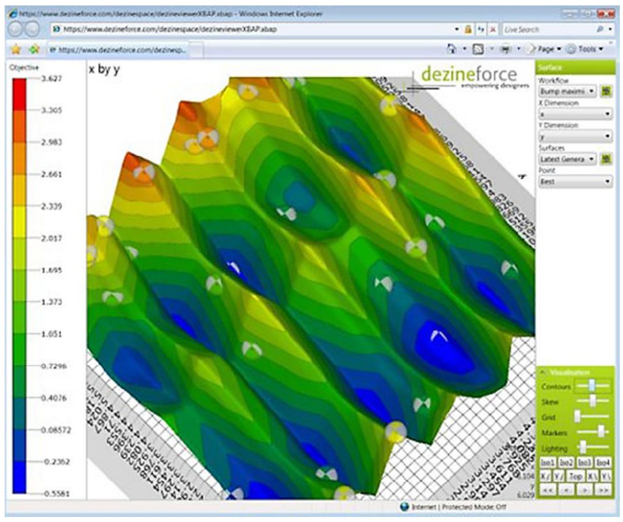

Figure 6. Surrogate model accurately predicting peaks and troughs with very little information

This enables engineers to gain insights into their designs, with far fewer resource-intensive simulations, and find high-quality non- intuitive and counterintuitive designs. This results in better products, with reduced costs and reduced time to market.

3.3 Engineering Process Orchestration

An engineering process can be as simple as a single simulation. More likely, however, it is made up of several different types of computation in a defined sequence, which may have some form of parallelism.
The data from one simulation will often have to be passed to the next simulation and possibly modified in some way. A more complex systematic exploration will involve multiple loops that have a wide degree of parallelism, which results in many threads of execution.
The architecture must be able to coordinate and monitor these parallel computations, reacting to their state changes in an appropriate manner. It must ensure that each computation executes within its own private area, within the correct security context. It must scale to support a large number of simultaneous processes in a way that is fair to its multiple tenants and provides a good user experience.

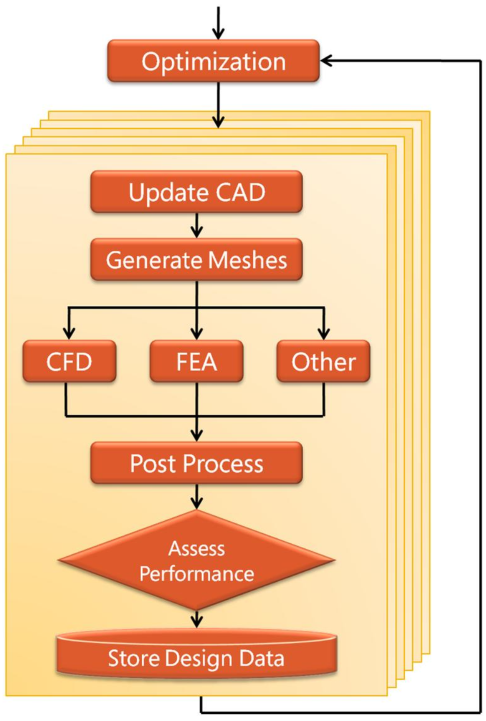

Figure 7. High-level schematic of simple optimization workflow

Figure 7 shows a high-level schematic of a simple optimization workflow.
This level of automation can greatly reduce the amount of engineer time that is required to set up and launch the simulations— reducing costs, increasing productivity, and reducing time to market. It is also a significant step towards compliance, because it ensures a consistent, well-defined, and repeatable process that is fully audited.
Within the dezineforce implementation, Windows Workflow Foundation (WF) forms the basis of process orchestration. Its ability to support episodic execution—and thereby load and unload workflows dynamically from memory—allows a large number of long-running processes to be serviced with relatively little overhead. This holds true, even when an individual workflow has several hundred parallel branches running concurrently. It also has many extensibility points that have allowed it to be heavily customized to meet requirements.

3.4 Engineering Computation

The architecture must provide powerful engineering tools and high-performance and high-throughput computing. This reduces the capital expenditure of engineering companies in terms of both software licenses and hardware. It also gives them access to a wider variety of applications, which allows them to take on new engineering challenges and open up new revenue streams. The ability to scale on demand also improves their agility.
Given that an individual engineering process can execute a large number of simulations in parallel and that many engineering processes can run concurrently, there is a need for the architecture to coordinate the use of scarce resources. These are typically software licenses and hardware.
Hardware scheduling ensures that the compute nodes that are used to execute the engineering calculations are loaded efficiently. This is typically carried out by a job scheduler that distributes load across a single compute cluster. License scheduling ensures that software licenses are used effectively across all compute clusters.
Process orchestration must work with the hardware scheduling and license scheduling to submit the right jobs at the right time to the right cluster, based on current load conditions and license availability. It should also take into account job priorities and fairness.
To work efficiently, the architecture must support two forms of multi-tenancy. Multi-tenancy is often taken to

mean supporting multiple organization or tenants on the same physical infrastructure. Although this is true for engineering Software + Services, it is also highly desirable to run different combinations and versions of engineering applications on the same compute nodes at the same time for different tenants.

Each engineering application is different and reports errors in different ways: some through exit codes, some through output streams, and some by just freezing. The architecture must be flexible enough to detect errors in a variety of different ways. It must know how to terminate a rogue application cleanly and restart an application cleanly when the error condition indicates that a restart is feasible.

Within the dezineforce implementation, a license allocation, reservation, and revocation subsystem has been created as a custom application. Microsoft Windows HPC Server is used to control the scheduling of jobs onto the compute nodes. Its extensibility model has allowed it to be customized to interact with the other subsystems, so as to close the loop of process orchestration, license management, and job management. The use of heterogeneous compute nodes also has been shown to be possible in this architecture.

3.5 Long-Term, Large-Scale Data Management

As engineering processes generate very large data sets rapidly, the architecture must provide high-speed access to large storage areas. In many industries, engineers are required to keep design and simulation data for the lifetime of the products. For aviation, this is often several decades. Therefore, the architecture should allow data to be kept for extended periods of time to support compliance. If this capability is provided, checks must be in place to ensure that data has not been corrupted—with non-repudiation to ensure that changes to data are attributable.

It is common practice to use a previous design, possibly many years later, as the starting point for a new design. As well as locating the original design it must be possible for the new engineer to rapidly understand how the original design was reached, the different variants that have already been explored and how the existing design performs. This improves engineer productivity and reduces time to market. Although a UI must provide lightweight methods to explore the data without bringing it down to the client, a subset of the data often will need to be downloaded, especially if long-term data storage is not provided. The architecture must provide for high-performance, secure, and resumable downloads for very large data sets. This might be required to support compliance.

Within the dezineforce implementation, storage technology has been used to provide large-scale, high-speed file access with very high availability. It also implements local data mirroring and data mirroring between different sites to support compliance and high availability.

Microsoft SQL Server is used to store non-file–based data such as the execution history and the underlying values that are used by the optimization process to support rich querying of the data.

4. Security

Within the dezineforce implementation, general authentication is based on Microsoft Active Directory, with network-level security implemented within the switch configurations. Role-level authorization is then used to distinguish between different levels of user and their permitted actions.

Security on the compute cluster is more complex to implement. Many of the engineering applications execute by using journals (a proprietary form of script) that are capable of launching arbitrary code; therefore, it is especially important that each job be scheduled to run within a security context that has a limited sandbox. Because jobs can be submitted deep within the architecture, long after the user has left the site, it is not possible to use delegation in the normal sense.

5. Management

Within the dezineforce implementation, systems-center operations management is used to monitor all aspects of the system in conjunction with external services that monitor overall system availability.

Microsoft Windows Compute Cluster Server (today superseded by Microsoft Windows HPC Server 2008) provides its own management console, which is used to manage the compute clusters and the compute nodes within them.

6. Impact on Engineering Companies

By looking at our original customer aims of cutting costs, developing new revenue streams, increasing productivity, increasing agility, reducing time to market, creating better products, and adhering to compliance—and relating this to our five key architectural challenges—we can see a strong correlation between them, as Table 1 shows.

	Cost reduction	New revenue streams	Productivity	Agility	Time to market	Better products	Compliance
Engineer interaction			■		■		
Engineering intelligence	■	■			■	■	
Engineering process orchestration	■		■		■		■
Engineering computation	■	■		■			
Long-term, large-scale data management			■		■		■

Table 1. Mapping of architectural challenges to customer needs

The availability of engineering computation/simulation in the Cloud can reduce costs, open up new revenue streams (by providing access to capabilities that are normally beyond the reach of an organization), and provide a level of agility through the ability to ramp-up on demand. However, it is the orchestration of these computations, combined with effective engineer interaction, that will provide the real productivity improvements and significantly reduce time to market.

Engineering intelligence further improves efficiency by focusing the computational effort on the areas that are likely to produce good designs. It is also an effective route to finding high-quality non- intuitive or counterintuitive designs. Importantly, this can be achieved while keeping the engineer in control of the process.

A recent study by Catalyzt analyzed the dezineforce solution and compared the costs against conventional design approaches, and across a wide range of engineering-design activities. This analysis covers different levels of design complexity, types of design "solvers," and areas of engineering design. Catalyzt concluded that "well- designed, well-executed SaaS services, such as the dezineforce service, can dramatically cut the costs of engineering design."

The headline results are included in the following list, courtesy of Cartezia.

The following were key areas of the examined and compared costs:

- IT costs, including:
  - Analysis
  - Application licensing
  - Computing hardware
  - System management and support
  - Setup (procurement and commissioning)
- Designer time:
  - Analysis setup and postprocessing
  - Modeling
  - Assessing scope for design improvement
  - Making design decisions
- Risk management:
  - Late-in-cycle changes
  - Recalls/warranty claims/penalties

The cost-comparison curves that are reproduced in Figures 8–10 dramatically illustrate the cost advantage of the dezineforce offering over conventional design approaches. To this cost advantage should be added the additional benefits of significantly enhanced design optimization (giving better designs) and the design flexibility that is enabled by use of a subscription-based service—with multiple tiers of subscription usage coupled with the ability to buy "top-up" design capability.

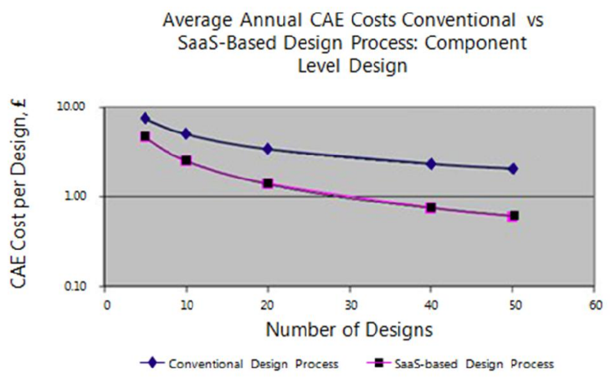

Figure 8. Component-level cost comparison

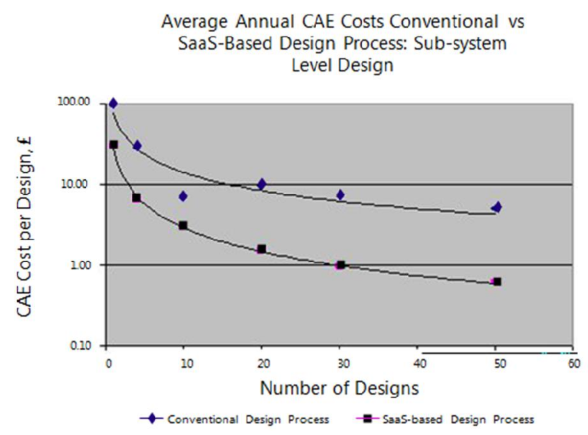

Figure 9. Subsystem-level cost comparison

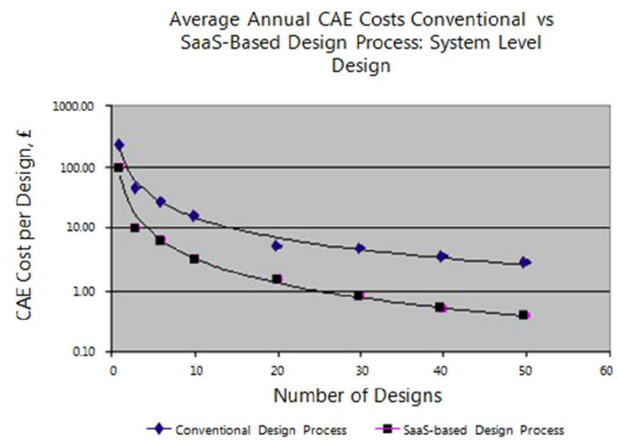

Figure 10. System-level cost comparison

A move to this new model is also a shift from capital expenditure to operational expenditure, which is particularly significant in turbulent times.

By looking at current trends, it is clear that globalization is increasing within engineering, with a desire for design teams to span multiple geographies. The provision of globally accessible, centralized computation and storage addresses many of the challenges that engineering companies face when they work globally. It allows all members of the team to run simulations, without having to provide extensive computation facilities at each site.

The ability for engineers to search and view results without having to download the full data set significantly reduces the need to move large data sets around the globe—removing the time delays and network infrastructure requirements that are inherent in doing so. The net result is that engineers in diverse geographies can both follow the progress of and control the design exploration, in near–real time, over standard Internet connections. This has the potential to support a follow-the-sun model, as offices in different time zones come online.

The instant availability and scalability of a Cloud-based service— compared to the long lead times that are involved in procuring and commissioning a dedicated engineering-computation facility—should also not be overlooked.

7. Conclusion

We have defined a class of service called “Engineering in the Cloud” based on the real-world needs of engineering companies:

“Engineering in the cloud is a combination of cloud services and rich interactive applications that provides integrated, intelligent, self-service engineering services over and above engineering- application hosting and computation—allowing engineers to create, explore, and discover better designs faster.”

We have shown that the hosting of engineering computations on its own does not satisfy this requirement, and we have put forward an architecture, called “Engineering Software + Services,” that can do so.

We have then used dezineforce as a case study to show how this architecture can—and has been—effectively created. During the case study, we walked through the key features of the architecture and described the technologies that are used to implement each of them.

We have shown how “Engineering in the Cloud,” implemented as an engineering Software + Services architecture, can affect a step change in engineering and is a reality today.

References

[1] De Souza, A., R. Harding, A.J. Keane, and S.J. Cox. "Integrated Design, Search, and Optimization for All." NAFEMS World Congress, June 2009.

[2] Forrester, Alexander, András Sóbester, and A.J. Keane.*Engineering Design via Surrogate Modelling: A Practical Guide*. Chichester, West Sussex, England: John Wiley & Sons Ltd., 2008.

[3] Keane, A.J., and P.B. Nair. *Computational Approaches for Aerospace Design: The Pursuit of Excellence*. Chichester, England; Hoboken, NJ: John Wiley & Sons Ltd., 2005.

[4] Keane, A.J., and J.P. Scanlan. "Design Search and Optimization in Aerospace Engineering." *Philosophical Transactions of the Royal Society A (Phil. Trans. A)*. Volume 365, No 1859, 2007, 2501–2529.

Peter Williams is Chief Technology Officer of dezineforce. A Microsoft Certified Architect, he is responsible for the architecture of the dezineforce "Engineering in the Cloud" implementation and for ensuring that the technology strategy supports the business strategy. Peter joined dezineforce from Microsoft Corporation, where he held a number of technical and managerial roles; more recently, he held the position of Senior Solutions Architect within the Microsoft U.K. professional services organization, where he led the development of numerous enterprise- scale solutions. Previously, he was group manager for software and infrastructure development within the Microsoft U.K. solution- development center and spent a number of years in Redmond, WA, working in the product groups.

Simon Cox is Professor of Computational Methods in the Computational Engineering Design Research Group within the School of Engineering Sciences of the University of Southampton, as well as Chief Scientist at dezineforce. An MVP Award holder, he directs the Microsoft Institute for High Performance Computing at the University of Southampton and has published over 120 papers. Currently, Simon heads a team that applies and develops computing in collaborative interdisciplinary computational science and engineering projects, such as computational electromagnetics, earth system modeling, applied computational algorithms, and distributed service-oriented computing for engineering.