2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences

Scientific Data and Knowledge Management in Aerospace Engineering

Andreas Schreiber, Jens Rühmkorf, Doreen Seider
Simulation and Software Technology
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Linder Höhe, 51147 Cologne, Germany
{*andreas.schreiber,jens.ruehmkorf,doreen.seider*}*@dlr.de*

***Abstract*—In aerospace engineering, simulation is a key technology. Examples are pre-design studies, optimization, systems simulation, or mission simulations of aircrafts and space vehicles.**

These kinds of complex simulations need two distinct technologies. First, highly sophisticated simulation codes for each involved discipline (for example, codes for computational fluid dynamics, structural analysis, or flight mechanics) to simulate the various physical effects. Secondly, a simulation infrastructure and well-designed supporting tools to work effectively with all simulation codes.

This paper focuses on the infrastructure and the supporting tools, especially for managing both the data resulting from large-scale simulation and the necessary knowledge for conducting complex simulation tasks. Examples of recent developments at the German Aerospace Center in the fields of data and knowledge management to support aerospace research by e-Science technologies are presented.

***Keywords*-data management; knowledge management; grid computing; aerospace; concurrent engineering; e-science**

I. Introduction

Aerospace engineering deals with the design of aircrafts and space crafts. Examples are manned or unmanned and civil or military aircrafts as well as satellites or other space vehicles such as landers or re-entry vehicles.

The design of all these crafts involves at least two different development phases: First, the pre-design phase for rough estimation of the overall behaviour; secondly, the high precision simulation which often includes full mission simulations. These design and simulation tasks are usually conducted by large teams of engineers from different disciplines. Therefore, a key technology in aerospace research and development is the availability of a supporting software environment and work methodologies like concurrent engineering.

Also, many design steps in aerospace engineering include high-resolution parallel simulation on supercomputers using sophisticated numerical algorithms and optimized codes. Examples are the complete simulation of all flow phenomena throughout the entire flight envelope including the multidisciplinary simulation of all involved disciplines of space and aerospace vehicles or the multidisciplinary optimization of the overall aircraft design as well as the design of major parts, such as turbine engines.

Goals of simulation in aerospace engineering are to analyze the aerodynamic and aeroelastic behaviour of the aircraft and its parts and the numerical prediction of the aircraft performance before the first flight. Other industrial sectors have similar objectives, for example the ship building or the automobile industry.

These kinds of complex simulations need two distinct technologies: first, highly sophisticated and optimized numerical simulation codes for each involved discipline (for example, codes for computational fluid dynamics (CFD), structural analysis, or flight mechanics); secondly, an efficient simulation infrastructure and well-designed supporting tools which can be categorized into:

- Distributed computing and Grid environments,
- data and workflow management tools,
- knowledge management and documentation tools, and
- pre/post processing and visualization software.

Figure 1 shows a generalized overall view with all tools and technologies. Usually, all these tool are integrated by using a suitable integration framework. Integrated environments allow the easy use of these tools and add additional functionalities.

The rest of this paper presents recent developments at the German Aerospace Center (DLR) and is organized as follows. Distributed computing and data management tools are described in Section II. In Section III, an integrated environment for workflow management is described. Section IV describes current knowledge management activities. Finally, a summary, concluding remarks, and future directions are given in Sections V and VI.

II. Distributed Computing and Data Management

Grid Computing [1] is an important technology for utilizing large distributed computing and storage resources and for collaboratively working between different institutions. The two major Grid middlewares are the Globus Toolkit [2] and UNICORE [3]. Both provide support for secure seamless access to distributed resources, authentication and authorization, file transfer, and data management. They have client Application Programming Interfaces (APIs) for enabling applications to use the Grid infrastructure.

For aerospace engineering, a concrete Grid infrastructure is the German AeroGrid described in the following.

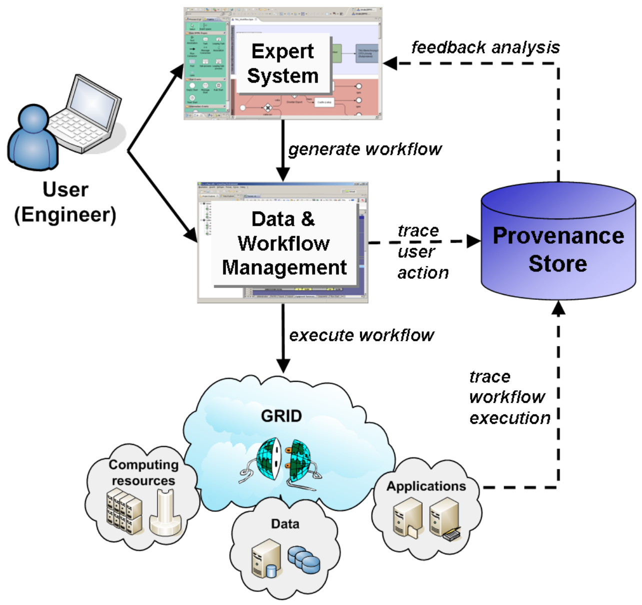


Figure 1. Generalized information science view about tools and technologies for arbitrary aerospace simulations.

A. AeroGrid

The AeroGrid aims at providing an efficient Grid based working environment for the aerospace research community [4]. The AeroGrid environment allows cooperation in research and development projects with virtual organizations, the use of always up-to-date program versions, data, and compute resources across all locations, and to document and trace the detailed history of a process that leads to a certain result (*Provenance*, see Section III-B). AeroGrid allows to use distributed compute resources by industrial and academic users, to collaborate during the design of new engine components, and to further develop CFD codes cooperatively. AeroGrid now employs the Grid middleware UNICORE 6. It provides a meta data service, a provenance service, and two different user interfaces (the data management client DataFinder and a Web Portal).

B. DataFinder

The DataFinder [5] is an easy-to-use data management client for Grids that primarily targets the management of scientific technical data. Data can be attached with meta data that is based on a free-definable data model to achieve data structuring and ordering. Moreover, the system is able to treat large amounts of data and can be easily integrated in existing working environments. The system is based upon a client-server-architecture and uses open, stable standards. One of its main features is the extensibility using Python scripts. The server side stores the meta data as well as the system configuration. The standardized protocol WebDAV is used by the client to access data and meta data on the server. Data can be stored on the same server as the meta data or separated from it. The concept of separated meta data and data storage allows the flexible usage of heterogeneous storage resources (for example, FTP, GridFTP, Amazon S3, Storage Resource Broker, or WebDAV). DataFinder can be customized to the specific application scenario, for example, with extensions of the graphical user interface to simulate turbines (Figure 2).

C. Data Organization and Meta Data

The DataFinder can be used as a simple client software for just storing data in Grids, but even more important is it's ability to structure existing data and combine it with meaningful meta data (*data model*). The customizing steps usually include an extensive requirements analysis for the different engineering tasks which leads to a logical view to all data files involved.

The meta data ("data about other data") describes data (*files*) and annotates collections of files (*directories*). Usually, administrators define required meta data, which must be specified, and users are free to add additional meta data. The meta data itself can have different types, for example strings, numbers, lists, images, or links.

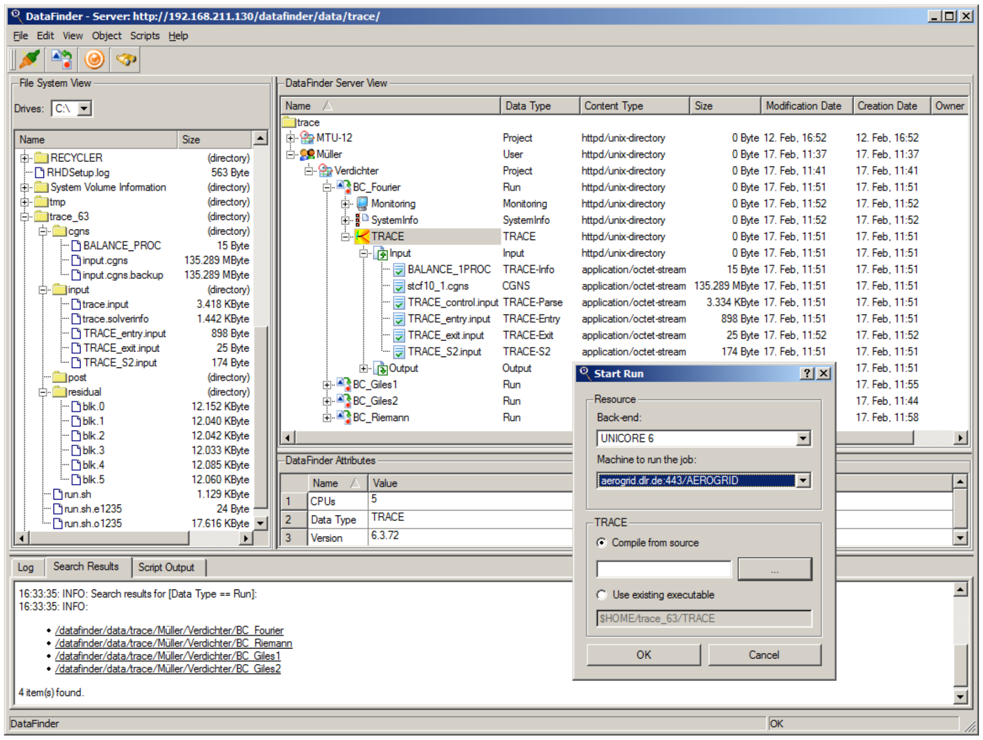
Based on the meta data, users can do a complex search to find existing data files. Since searching for data typically consumes a significant amount of the engineer's working time, an efficient search method makes possible to work more efficiently.

III. Integrated Environments and Workflow Management

Lightweight tools such as DataFinder are suitable for specific tasks with fixed workflows or with focus on data management. For that, the effort to customize the tool and to integrate it into the working environment is very low. For more complex simulation tasks involving large dynamic workflows, a more general tool should be used, for example the Reconfigurable Computing Environment (RCE).

A. Reconfigurable Computing Environment (RCE)

RCE is a service oriented software framework to manage collaborative engineering processes. It hides the complexity of heterogeneous and distributed IT systems behind a common user interface and hereby enforces secure and uniform access of data and services. RCE is application independent and easily adaptable to a variety of application domains. It is based on OSGi (Open Services Gateway initiative), the industry standard for modular dynamic Java applications, therefore RCE is platform independent and can be used on most architectures. The seamless integration of Grid-technologies into RCE allows the transparent access to resources. RCE can be easily extended by application specific services which are added and managed as *plug-ins*, the central mechanism known from the Eclipse universe. Non-Java code, like C or FORTRAN, can be integrated via


DataFinder - Server: http://192.168.211.130/datafinder/data/trace/
File Edit View Object Scripts Help
File System View
Drives: C:\
DataFinder Server View
DataFinder Attributes
Log Search Results Script Output
DataFinder
OK

Name	Size
RECYCLER	(directory)
RHDSetup.log	563 Byte
System Volume Information	(directory)
tmp	(directory)
trace_63	(directory)
cgns	(directory)
BALANCE_PROC	15 Byte
input.cgns	135.289 MByte
input.cgns.backup	135.289 MByte
input	(directory)
trace.input	3.418 KByte
trace.solverinfo	1.442 KByte
TRACE_entry.input	898 Byte
TRACE_exit.input	25 Byte
TRACE_S2.input	174 Byte
post	(directory)
residual	(directory)
blk.0	12.152 KByte
blk.1	12.040 KByte
blk.2	12.042 KByte
blk.3	12.033 KByte
blk.4	12.085 KByte
blk.5	12.060 KByte
run.sh	1.129 KByte
run.sh.e1235	24 Byte
run.sh.o1235	17.616 KByte

Name	Data Type	Content Type	Size	Modification Date	Creation Date	Owner
trace						
MTU-12	Project	httpd/unix-directory	0 Byte	12. Feb, 16:52	12. Feb, 16:52	
Müller	User	httpd/unix-directory	0 Byte	17. Feb, 11:37	17. Feb, 11:37	
Verdichter	Project	httpd/unix-directory	0 Byte	17. Feb, 11:41	17. Feb, 11:41	
BC_Fourier	Run	httpd/unix-directory	0 Byte	17. Feb, 11:51	17. Feb, 11:51	
Monitoring	Monitoring	httpd/unix-directory	0 Byte	17. Feb, 11:52	17. Feb, 11:52	
SystemInfo	SystemInfo	httpd/unix-directory	0 Byte	17. Feb, 11:52	17. Feb, 11:52	
TRACE	TRACE	httpd/unix-directory	0 Byte	17. Feb, 11:51	17. Feb, 11:51	
Input	Input	httpd/unix-directory	0 Byte	17. Feb, 11:51	17. Feb, 11:51	
BALANCE_1PROC	TRACE-Info	application/octet-stream	15 Byte	17. Feb, 11:51	17. Feb, 11:51	
stcf10_1.cgns	CGNS	application/octet-stream	135.289 MByte	17. Feb, 11:51	17. Feb, 11:51	
TRACE_control.input	TRACE-Parse	application/octet-stream	3.334 KByte	17. Feb, 11:51	17. Feb, 11:51	
TRACE_entry.input	TRACE-Entry	application/octet-stream	898 Byte	17. Feb, 11:51	17. Feb, 11:51	
TRACE_exit.input	TRACE-Exit	application/octet-stream	25 Byte	17. Feb, 11:52	17. Feb, 11:52	
TRACE_S2.input	TRACE-S2	application/octet-stream	174 Byte	17. Feb, 11:51	17. Feb, 11:51	
Output	Output				17. Feb, 11:51	
BC_Giles1	Run				17. Feb, 11:55	
BC_Giles2	Run				17. Feb, 11:44	
BC_Riemann	Run				17. Feb, 11:58	

	Name	Value
1	CPUs	5
2	Data Type	TRACE
3	Version	6.3.72

Start Run

Resource

Back-end: UNICORE 6

Machine to run the job: aerogrid.dlr.de:443/AEROGRID

TRACE

[✓ Compile from source]

Use existing executable: $HOME/trace_63/TRACE

OK Cancel

```
16:33:35: INFO: Search results for [Data Type == Run]:
16:33:35: INFO:

    • /datafinder/data/trace/Müller/Verdichter/BC_Fourier
    • /datafinder/data/trace/Müller/Verdichter/BC_Riemann
    • /datafinder/data/trace/Müller/Verdichter/BC_Giles1
    • /datafinder/data/trace/Müller/Verdichter/BC_Giles2

4 item(s) found.
```

Figure 2. Simulation of turbines in Grids with DataFinder.

provided code wrapping technologies, which are designed to integrate existing code very easily. It provides services for data management, data access, or workflow management. A workflow in RCE consists of plug-ins (i.e., application specific services). Plug-ins can have different requirements on the resources they are running on. Some have a high demand for computational power, others need a huge storage capacity. RCE is realized as a distributed system. Due to this fact it is possible to run each plug-in of a workflow on the resource which meets its needs best. Just a RCE instance has to be installed there.

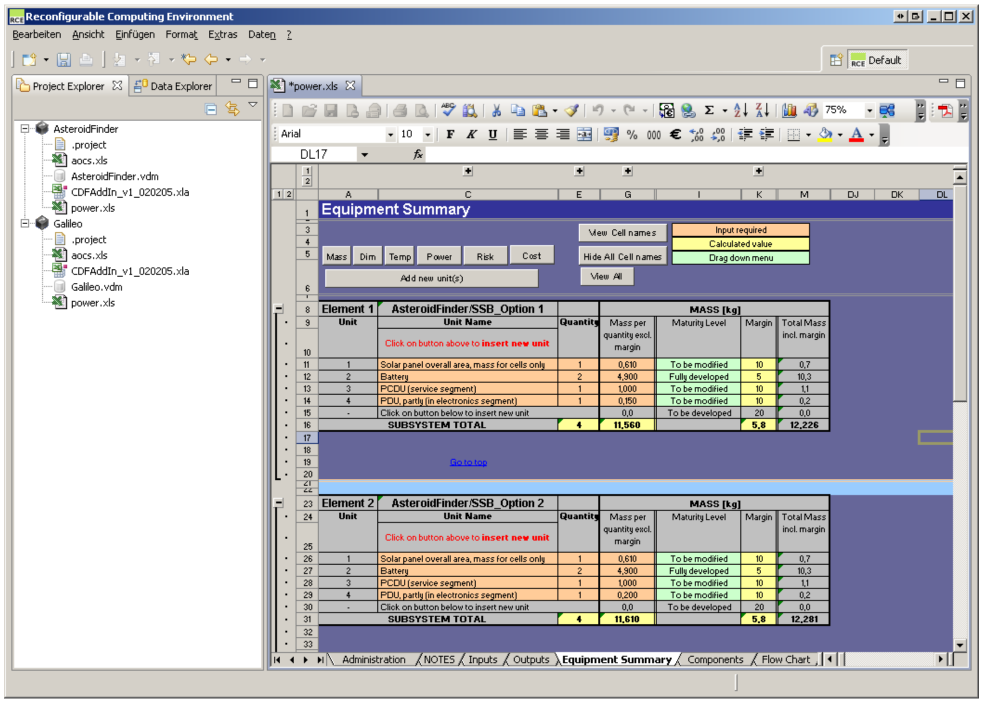
For workflows with complex, computationally intensive simulations, RCE provides a Grid interface that enables plug-ins to source computations out to the Grid. This interface is realized by integrating the Grid Application Toolkit (GAT) [6] into RCE. GAT provides a uniform API to access heterogeneous compute and data resources. By making use of adaptors it is able to support different Grid middleware (e.g., Globus or UNICORE) and resource systems by having one uniform and stable API. New technologies will be supported by adding an adaptor without the need to modify the API. This Grid interface of RCE allows workflows to be executed on the Grid. More precisely, their components (i.e., plug-ins) are enabled to run computations in the Grid. The fact that not the whole workflow is completed in the Grid has assets and drawbacks. On the one hand it is not possible to benefit from scheduling methods specialized in the Grid fields because scheduling of the whole workflow has to be done outside of it. In regard to the plug-ins it is possible to make use of Grid scheduler. On the other hand this concept allows the combination of plug-ins running in the Grid with plug-ins being executed in RCE. This leads to a very flexible and customizable design.

Since scientists in aerospace engineering are using Microsoft Excel for various tasks (for example, "small calculations" or visualizations) and many input data for calculations is stored in Excel spreadsheets, RCE has an interface to Excel. This integration allows engineers to either access data in Excel sheets or include Excel as a component into the simulation workflow.

B. *Provenance*

Data files are annotated with meta data to make an efficient search for it (see Section II-C). A similar concept applies to workflows (or computational processes), where a recorded documentation of the process execution makes possible to search for conducted workflows and to analyse the executed workflows. The detailed history of the processes as they took place is called *Provenance* information. A more general definition of Provenance is "the Provenance of some information is the history of its creation" [7].

With Provenance information, one can determine the origin of electronic data and to ensure the authenticity of results and to determine the compliance of the process that led to the data.



Element 1	AsteroidFinder/SSB_Option 1		MASS [kg]			
Unit	Unit Name	Quantity	Mass per quantity excl. margin	Maturity Level	Margin	Total Mass incl. margin
	Click on button above to **insert new unit**					
1	Solar panel overall area, mass for cells only	1	0,610	To be modified	10	0,7
2	Battery	2	4,900	Fully developed	5	10,3
3	PCDU (service segment)	1	1,000	To be modified	10	1,1
4	PDU, partly (in electronics segment)	1	0,150	To be modified	10	0,2
-	Click on button below to insert new unit		0,0	To be developed	20	0,0
	SUBSYSTEM TOTAL	4	11,560		5,8	12,226

Element 2	AsteroidFinder/SSB_Option 2		MASS [kg]			
Unit	Unit Name	Quantity	Mass per quantity excl. margin	Maturity Level	Margin	Total Mass incl. margin
	Click on button above to **insert new unit**					
1	Solar panel overall area, mass for cells only	1	0,610	To be modified	10	0,7
2	Battery	2	4,900	Fully developed	5	10,3
3	PCDU (service segment)	1	1,000	To be modified	10	1,1
4	PDU, partly (in electronics segment)	1	0,200	To be modified	10	0,2
-	Click on button below to insert new unit		0,0	To be developed	20	0,0
	SUBSYSTEM TOTAL	4	11,610		5,8	12,281

Figure 3. Design of satellites with RCE using embedded Excel spreadsheets.

Provenance information has many benefits for aerospace applications (see [8] for details). First, it is a clean documentation of distributed simulation workflows. It makes possible to analyze and reason all conducted computations, even after years. An example is checking for requirements which regard compliance to legal and business regulations. The Provenance information is also a proof of the results and makes them reproducible.

In practice, Provenance information is stored in a *Provenance store*, which usually is an XML data base (Figure 4). The information to be recorded depends on the application [9]. For simulations in aerospace engineering, typically the parameters of the simulated model (including the "classical" meta data), versions of the simulation codes used, information about the machines on which the codes are executed, and information about the libraries and compilers used are recorded. Important is that not only the information itself is recorded but also a description of all interactions (between users, codes, machines etc.) during process execution.

After recording the Provenance information, engineers can do queries on the Provenance store. Some example queries are:

- For a given parameter, in which simulation(s) has it been used?
- Which data has been produced in a simulation with a specific parameter?
- What simulations have been run using a given model (aircraft design)?

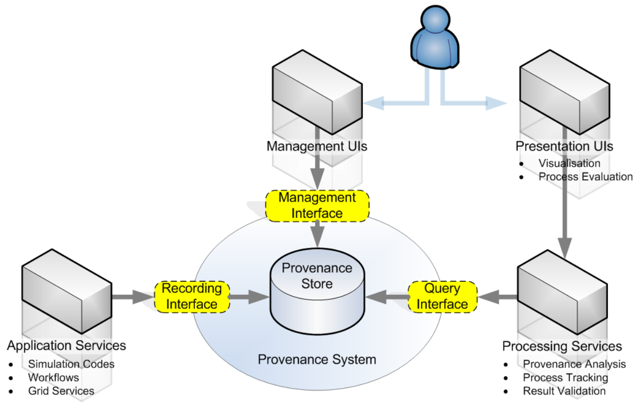
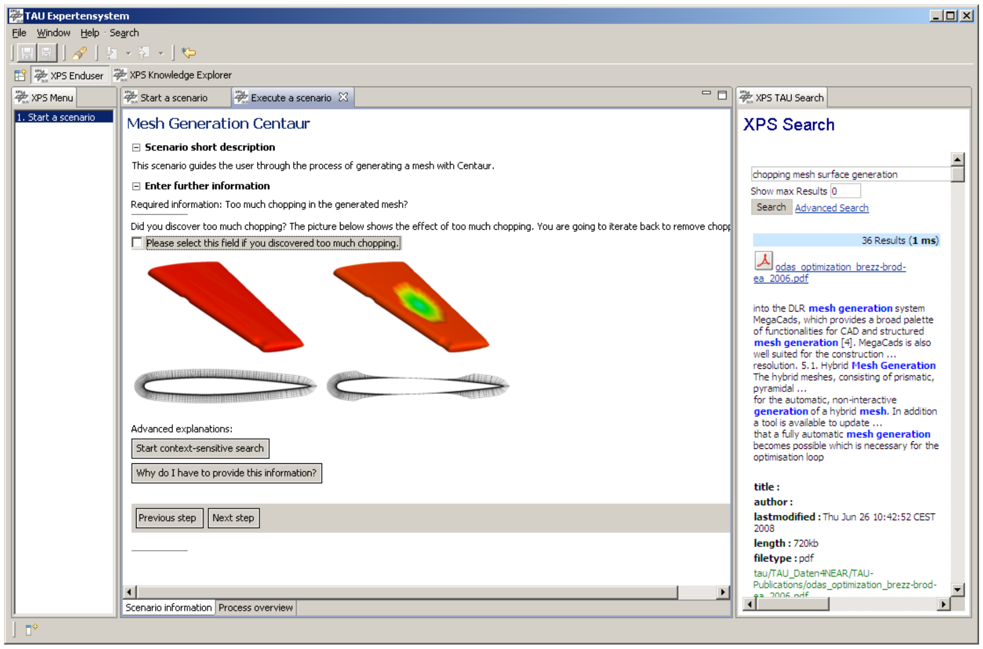


Figure 4. Architecture of the Provenance infrastructure.

- What have been the initial conditions for this simulation?
- Which were the termination criteria?
- How often has the workflow been executed?
- Which specific algorithm/version has been used in the computations?
- Which were the hosts participating in the simulation run?

IV. Knowledge Management

In aerospace engineering, the efficiency of computational design packages has a direct influence on the design and development costs. Today's systems are heavily dependent on



Mesh Generation Centaur

Scenario short description

This scenario guides the user through the process of generating a mesh with Centaur.

Enter further information

Required information: Too much chopping in the generated mesh?

Did you discover too much chopping? The picture below shows the effect of too much chopping. You are going to iterate back to remove chopp

☐ Please select this field if you discovered too much chopping.

XPS Search

chopping mesh surface generation

Show max Results 0

Search Advanced Search

36 Results (**1 ms**)

odas_optimization_brezz-brod-ea_2006.pdf

into the DLR **mesh generation** system MegaCads, which provides a broad palette of functionalities for CAD and structured **mesh generation** [4]. MegaCads is also well suited for the construction ... resolution. 5.1. Hybrid **Mesh Generation** The hybrid meshes, consisting of prismatic, pyramidal ... for the automatic, non-interactive **generation** of a hybrid **mesh**. In addition a tool is available to update ... that a fully automatic **mesh generation** becomes possible which is necessary for the optimisation loop

title :
author :
lastmodified : Thu Jun 26 10:42:52 CEST 2008
length : 720kb
filetype : pdf
tau/TAU_Daten4NEAR/TAU-Publications/odas_optimization_brezz-brod-ea_2006.pdf

Figure 5. Information about mesh generation in the knowledge management tool *XPS for CFD*.

the users driving them. Identifying and sharing best practice between users is therefore crucial to reduce the variability of design quality and therefore the final cost of components produced. In large scale projects (such as airplane design) designs of components are undertaken by various consortium members from domains such as aerodynamics, flight performance, propulsion, structure mass, stability and control. For example, a simulation of a turbine engine alone with a sufficient resolution (50 million grid points) now takes about 2–3 weeks on state-of-the-art computing facilities. Therefore, best practices must be adhered to and shared among the users and experts of the fields involved. The DLR knowledge management system *XPS for CFD* addresses this issue.

At DLR, various projects activities have been undertaken to gather systematic knowledge in form of best practices and actual expert knowledge, with the aim to enable CFD users to obtain consistent high quality results with reduced uncertainty and lower costs for a wide range of aerospace flow problems. As part of these projects, this knowledge and there best practices have been integrated into the system *XPS for CFD*, thus providing the CFD user with specific guidelines, rules, standards for individual codes and algorithms, and ready to be deployed workflows.

The knowledge management system differentiates between various roles (e.g., *domain expert* that is knowledgeable in a certain field, *system manager*, or *user*). The system aims to offer a structured representation of the knowledge of a field, which is adequately easy enough to manage and allows for the later extension in different fields. The representation of knowledge with *rules* is often useful since this mimics human patterns of thoughts [10].

The implementation has been carried out on top of RCE by integrating the RETE-based [11] rule engine JBoss Drools [12].

V. Conclusion

Combining the described software tools results in a sophisticated distributed working environment for all kind of design and simulation tasks in aerospace engineering. Especially, it supports concurrent engineering where coordinated work on a common data model is important. The recording of Provenance assures traceability and gives engineers and customers confidence in the results obtained. Also, linking expert knowledge with Grid environments makes large-scale complex aerospace simulations very easily possible. This includes the use of data and meta data within the semantic context of aerospace engineers and the use of expert knowledge to generate simulation workflows being executed in a Grid.

In summary the described tools make possible for aerospace engineers to

- share their knowledge with others,
- use knowledge of others (e.g., of current and former colleagues),
- work with data (e.g., generate additional data products such as movies),
- search for data using meta data and already performed simulations using Provenance information,

- select codes and computing resources for simulations, and
- create computational workflows manually depending in their design problems.

The software tools support users in various steps. They

- generate workflow descriptions based on expert knowledge,
- execute workflows on distributed resources and Grids,
- extract meta data from results automatically, and
- record detailed Provenance for documentation and further analysis.

The outlined software infrastructure is also suitable for other application domains than aerospace engineering. Currently, parts of the infrastructure is being used for the automobile and ship building industry. The use for medical applications is in progress.

VI. Future Work

In the future, research at DLR will concentrate on a tighter integration of the expert system with the other tools, for example to use recorded Provenance information to generate new knowledge about simulation processes or to adapt existing rules in the expert system (*feedback analysis*).

The knowledge management system will be integrated with external data sources. Currently the expert system supports full text search in literature data bases and in documents managed by the data management tool (Figure 5). In the future, the integration with special search engines such as *Wolfram|Alpha* for receiving background information and performing sample computations will help engineers. For example, the calculation of input parameters such as coefficients related to different angles of attack for airfoils could be performed by Wolfram|Alpha and the underlying software Mathematica very easily. As an example see the NACA 12 airfoil[1].

A future extension will be the integration with Web 2.0 technologies [13] (e.g., Blogs, Wikis, or Social networks). Groups of engineers working in large teams distributed over different companies or countries share the same aspects and problems regarding communication and distribution of information as in any other social network. Web 2.0 technologies could support their work as it improves communication and sharing of knowledge.

Acknowledgment

This work has been supported by the German Federal Ministry for Research and Technology (BMBF) under Grant 01IG07006 (AeroGrid), Grant 01G07009 (PartnerGrid), and Grant 01AK800A (DGI) and by the European Community under Grant IST-2002-511085.

[1]http://www.wolframalpha.com/input/?i=NACA+12

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid – enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[2] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Lecture Notes in Computer Science*, vol. 3779, pp. 2–13, 2006.

[3] Jülich Supercomputing Centre. UNICORE. [Online]. Available: http://www.unicore.eu/unicore [accessed, July 13, 2009]

[4] German Aerospace Center (DLR), AeroGrid website. [Online]. Available: http://www.aero-grid.de [accessed, July 13, 2009]

[5] T. Schlauch and A. Schreiber, "Datafinder - a scientific data management solution," in *Ensuring the Long-Term Preservation and Value Adding to Scientific and Technical Data, PV 2007, Oberpfaffenhofen, Germany*, 2007. [Online]. Available: http://www.pv2007.dlr.de/ [accessed, July 13, 2009]

[6] G. Allen, K. Davis, T. Dramlitsch, T. Goodale, I. Kelley, G. Lanfermann, J. Novotny, T. Radke, K. Rasul, M. Russell, E. Seidel, and O. Wehrens, "The gridlab grid application toolkit," in *HPDC*, 2002, p. 411.

[7] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The provenance of electronic data," *Communications of the ACM*, vol. 51, no. 4, pp. 52–58, 2008.

[8] G. K. Kloss and A. Schreiber, "Provenance implementation in a scientific simulation environment," in *International Provenance and Annotation Workshop, IPAW 2006*, ser. Lecture Notes in Computer Science, vol. 4145. Berlin / Heidelberg: Springer, 05 2006, pp. 37–45.

[9] S. Munroe, S. Miles, L. Moreau, and J. Vzquez-Salceda, "Prime: a software engineering methodology for developing provenance-aware applications," in *Proceedings of the 6th international workshop on Software engineering and middleware*, ser. Foundations of Software Engineering. ACM, 2006, pp. 39–46.

[10] P. Jackson, *Introduction to Expert Systems*, 3rd ed. Boston, MA, USA: Addison Wesley, 1998.

[11] C. L. Forgy, *On the Efficient Implementation of Production Systems*. Pittsburgh, PA, USA: Carnegie-Mellon University, 1979.

[12] The JBoss Community Team, JBoss Drools. [Online]. Available: http://www.jboss.org/drools [accessed, July 13, 2009]

[13] M. E. Pierce, G. C. Fox, J. Y. Choi, Z. Guo, X. Gao, and Y. Ma, "Using web 2.0 for scientific applications and scientific communities," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 5, pp. 583–603, 2009.