AMANDA - A Distributed System for Aircraft Design

Hans-Peter Kersken¹, Andreas Schreiber¹, Martin Strietzel¹, Michael Faden¹,
Regine Ahrem⁶, Peter Post⁶, Klaus Wolf⁶, Armin Beckert⁴, Thomas Gerholt²,
Ralf Heinrich⁵, and Edmund Kügeler³

¹ DLR, Simulation- and Softwaretechnology, 51170 Cologne, Germany,

² DLR, Institute of Fluid Dynamics, 37073 Göttingen, Germany,

³ DLR, Institute of Propulsion Technology, 51170 Cologne, Germany,

⁴ DLR, Institute of Aeroelasticity, 37073 Göttingen, Germany,

⁵ DLR, Institute of Design Aerodynamics, 37073 Göttingen, Germany,

`<first name>.<last name>@dlr.de`

⁶ GMD, Institute for Algorithms and Scientific Computing, Schloss Birlinghoven,
53754 Sankt Augustin, Germany

`<first name>.<last name>@gmd.de,`

Abstract. In the AMANDA project a component-based framework for the integration of coupled technical applications, running distributed in a network, is developed. It is designed to deal with parallel and sequential programs and massive data exchange between the integrated programs. Two pilot applications will be implemented to show the feasibility of the chosen approach: a trimmed, freely flying, elastic airplane and an air-cooled turbine. Beside using the integration systems, the MpCCI¹ library is used for the coupling of the codes, each simulating single physical processes.

1 Introduction

An efficient design process for airplanes and propulsion systems requires the calculation of numerous variants for which, in the context of the product development, usually only a limited time window is available. A prerequisite for this task is the availability of high-quality procedures for the simulation of the physical processes involved, as well as their coupling, in order to be able to make predictions about physical and functional properties of the final product. In addition, a powerful soft- and hardware infrastructure must be available in order to be able to use these procedures fast and efficiently. Here a software integration system acts as an agent which manages the interaction of software tools. Otherwise the coupling of these tools would require time consuming human intervention. Additionally, it provides a uniform user interface to the integrated applications.

The AMANDA project has two main focuses: The first is to extent and enhance the software integration system TENT [9] which is a joint development of

¹ MpCCI is a trademark owned by GMD.

GMD and DLR. It was already used successfully in the project SUPEA [10] and is available as a prototype system [11]. The second focus is the implementation of pilot applications using this integration environment. Two applications have been selected:

- a trimmed, elastic, freely flying airplane and
- an air-cooled turbine.

The development of the integration framework is driven by requirements posed by these applications, e. g. the integration of parallel programs, the use of a hierarchical structure for setting up a process chain, or a control module to steer the execution of the workflow depending on internally calculated parameters.

The applications will be described in more detail in the next section, Sec. 3 gives an overview about the current state of the integration system, and Sec. 4 describes some new features of the integration system necessary for the implementation of the two AMANDA applications.

2 The AMANDA–Applications

2.1 Airplane Design

For the simulation of a trimmed, freely flying, elastic airplane the following programs have to be integrated into TENT:

- a CFD code, TAU [8] and FLOWer [4],
- a structural mechanics code (NASTRAN [7]) and a multi-body program (SIMPACK [5]), to calculate the deformation of the structure,
- a coupling module, to control the coupled fluid-structure computation,
- a grid deformation module, to calculate a new grid for the CFD simulation,
- a flight mechanics/controls module (build using an Object-Oriented modelling approach [6]), to set the aerodynamic control surface positions,
- visualization tools.

Figure 1 shows a possible scenario of the coupling. The process chain consisting of these codes is hierarchically structured. The lowest level contains the parallel CFD solver only. This CFD-subsystem consists of the flow solver itself and auxiliary programs to handle mesh adaptation. The next level, the aeroelastic-subsystem, comprises the CFD solver, the structural mechanics or multi-body code, and the grid deformation module. The data transfer from the CFD code to the structural mechanics code is performed via the MpCCI (former CoColib) [1] library which will be extended in order to deal with the interpolation used in fluid-structure interaction problems. For the solution of the coupled non-linear aeroelastic equations a staggered algorithm is implemented in the control process [2]. The highest level consists of the flight mechanics module coupled to the aeroelastic-subsystem. Each single code is additionally accompanied by its own visualization tool. The computation of a stable flight state typically proceeds as follows:

Fig. 2. Screenshot of the TENT gui for the turbine simulation described in Sec. 2.2

The CFD code TRACE [13], a 3D-Navier-Stokes-Solver for the simulation of steady and unsteady multistage turbomachinery applications, and a heat conduction problem solver (NASTRAN) are coupled to simulate the airflow through the turbine together with the temperature distribution inside the blades. For the coupling of the flow simulation and the heat conduction a stable coupling algorithm as been developed where TRACE delivers the temperatures of the air surrounding the blade and the heat transfer coefficients as boundary conditions to NASTRAN which in turn calculates the temperature distribution inside the blade. The temperature at the surface of the blade is returned to TRACE as boundary condition for the airflow.

A coupled simulation has been already realized using the MpCCI library for data exchange.

3 The Software Integration System

TENT is a framework for building and controlling complex technical workflows where the embedded application may be run on arbitrary machines in a network, i. e. the machines can be chosen depending on the application. It was developed in view of the integration of HPC applications and provides a graphical user interface to build process chains and control the integrated applications. Figure 2 show a screenshot for the simulation scenario described in Sec. 2.2. The

TENT software system consists of four different packages as shown in Fig. 3. The software development kit (SDK) summarizes all interface definitions and libraries for the software development with TENT. The base system includes all basic services needed to run a system integrated with TENT. The facilities are a collection of high-level services, and the components consist of wrappers and special services for the integration of applications as TENT-components.

3.1 The Software Development Kit

TENT defines a component architecture and an application component interface on top of the CORBA object model. The TENT component architecture is inspired by the JavaBeans specification [3]. The data exchange interface as part of the software development kit allows the transparent data exchange between two TENT-components. It can invoke a data converter, which automatically converts data between the data formats supported by the integrated tools. For efficiency reasons CORBA is used only to coordinate the data transfer, i. e. only references to the data are exchanged using CORBA itself. These references are for example path names in a file system or port numbers and IP-addresses, if socket communication is employed. The transfer of the data themselves is performed in a more efficient way, depending on the situation at hand, e. g. by socket communication or file transfer.

3.2 TENT - Base System

The TENT base system consists of the components necessary to run the integrated system of applications. For portability reasons, these components are currently implemented in Java. The Master Control Process (MCP) is the main control instance in TENT. It realizes and controls system tasks such as process chain management, construction and destruction of components, or monitoring the state of the system.

Fig. 3. Packages of the software integration system TENT.

Fig. 4. Architectural overview of TENT.

The factories run on every machine in the TENT framework. They start, control, and terminate the applications on their machine. The naming service is an enhanced CORBA naming service. The relations between the base components are shown in Fig. 4.

3.3 TENT Facilities

In order to support higher-level workflows, the system offers more sophisticated facilities, e. g. data and job management(scheduling system) servers are planned. An important facility is the data converter which acts as a filter when integrated programs with incompatible data formats exchange information.

3.4 TENT Components

All applications must be encapsulated by wrappers to access their functionality in TENT. Therefore these wrappers have to be equipped with a CORBA interface to comply to the TENT component architecture. Depending on the level of access to the program, source code, API, or file I/O, the wrapper can be tightly coupled with the application, e. g. linked together with it, or must be a stand alone tool that starts the application via system services. Depending on the wrapping mechanism, the communication between the wrapper and the application is implemented by direct memory access, IPC mechanisms, file exchange, or any other suitable communication method

4 Impacts of the AMANDA–Applications on TENT

In this section new features which have been added to the integration system due to requirements from the AMANDA-project are discussed.

4.1 NASTRAN-Co-process

Every application not especially written for the TENT-systems has to be wrapped to be accessible by the system. For codes with source code access it is now a standard task. Some codes have already been integrated by changing the main routine only. The source code had to be modified to include control and communication commands. For NASTRAN just the opposite was true, neither the source code was accessible nor does an API exist. A NASTRAN run is controlled completely by an input file. Therefore, an additional program had to be written, a NASTRAN-co-process, to hook up NASTRAN to the integration system. The co-process is responsible for setting up NASTRAN input files, parsing NASTRAN output files to extract the results, and connecting to the integration system. Because this module was developed with its integration into TENT in mind, it was implemented as a library which methods can be called from a Python script. If NASTRAN is a part of an application coupled by the MpCCI the script can handle the control of the coupled application as well.

4.2 Strongly Coupled Multi-disciplinary Subsystems

In the context of AMANDA the coupling of multi-disciplinary simulation codes is a major issue. Therefore an interface to the coupling library MpCCI is provided. MpCCI coordinates the exchange of boundary informations between simulation applications working on adjacent grids. The communication between applications inside the integration system is usually controlled by CORBA mechanisms. In case of applications coupled by the MPI-based MpCCI, data are exchanged using the MPI-library without any interference of the integration system. The coupling algorithm itself, i. e. the number of time or iteration steps a code has to perform or the decision when to terminate the computation is not part of the MpCCI library. Hence, a control process is introduced into TENT to handle this task. A scripting language (Python) is used to simplify the development and the maintenance of this module. The flexibility offered by a scripting language is more important here than in other modules because setting up the coupling of two independent programs based on algorithms originally not intended for the use in coupled computation may require a lot of experimentation to find a good, i. e. converging and stable, coupling algorithm.

4.3 Hierarchical Structure

As can be easily seen in Fig. 1, without a structuring strategy it soon becomes impossible to handle the complexity of coupled problems. A design decision motivated by scenarios like this was to allow the encapsulation of existing workflows to become a single TENT-component which in turn may consist of other composite components. Each component has its own control process and by defining appropriate interfaces the complexity of a module is completely invisible to other components. The master control process is hence replaced by a set of independently working control processes in each module.

5 Conclusions

The TENT integration system allows to link software tools originally intended for stand alone use. By embedding them into the TENT framework, on the one hand, a uniform access to the tools is provided, on the other hand, data are transferred transparently between different tools without intervention from the user. The applications described show the flexibility and extensibility of the integration system. It will be used and extended in other, scientific and industrial, projects, e. g. virtual prototyping in the automotive industry [12], the simulation of a re-entry system, and combustion chamber modeling.

Acknowledgments

The work described in this paper is partially funded by the Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren (HGF) and the German ministry for education and research under grant 01SF9822.

References

- [1] R. Ahrem, P. Post, and K. Wolf. A communication library to couple simulation codes on distributed systems for multiphysics computations. In *Proceedings of ParCo99*, Delft, August 1999.
- [2] A. Beckert. Ein Beitrag zur Strömung-Struktur-Kopplung für die Berechnung des aeroelastischen Gleichgewichtszustandes. ISRN DLR-FB 97-42, DLR, Göttingen, 1997.
- [3] G. Hamilton. *JavaBeans API Specification*. Sun Microsystems Inc., July 1997.
- [4] N. Kroll. The National CFD Project Megaflow - status report. In H. Körner and R. Hilbig, editors, *Notes on numerical fluid mechanics*, volume 60. Braunschweig, Wiesbaden, 1999.
- [5] W. Krüger and W. Kortüm. Multibody simulation in the integrated design of semi-active landing gears. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference*. Boston, 1998.
- [6] D. Moormann, P. J. Mosterman, and G. Looye. Object-oriented computational model building of aircraft flight dynamics and systems. *Aerospace Science and Technology*, 3:115–126, April 1999.
- [7] Nastran. <http://www.macsch.com>.
- [8] D. Schwamborn, T. Gerholt, and R. Kessler. DLR TAU code. In *Proceedings of the ODAS-Symposium*, June 1999.
- [9] M. Strietzel, T. Breitzfeld, T. Forkert, A. Schreiber, and K. Wolf. The distributed engineering framework TENT. to be published in: Vector and Parallel Processing – VECPAR2000. Lecture Notes in Computer Science, 2000.
- [10] Supea. <http://www.sistec.dlr.de/de/projects/supea>.
- [11] Tent. <http://www.sistec.dlr.de/tent>.
- [12] C.-A. Thole, S. Kolibal, and K. Wolf. AUTOBENCH: Environment for the Development of Virtual Automotive Prototypes. In *Proceedings of 2nd CME-Congress (to appear)*, Bremen, September 1999.
- [13] D. T. Vogel and E. Kügler. The generation of artificial counter rotating vortices and the application for fan-shaped film-cooling holes. In *Proceedings of the 14th ISABE*, ISABE-Paper 99-7144, 1999.