# Advances in Generalization and Decoupling of Software Parts in a Scientific Simulation Workflow System

Arne Bachmann, Markus Kunde, Markus Litz, Andreas Schreiber
*German Aerospace Center*
*Simulation and Software Technology*
*Cologne, Germany*
{Arne.Bachmann, Markus.Kunde, Markus.Litz, Andreas.Schreiber}@dlr.de

***Abstract*— Scientific simulation workflows today consist of a pool of simulation models of different domains that are linked together. In the past this was often done with highly specific connections between the simulation models, e. g., batch-scripts or use of commercial integrated systems prescribing certain procedures. This strong coupling led to several problems like the non-reusability of a simulation model in other contexts or other software environments. To address this situation a concept called *Chameleon* was developed to provide a general decoupled approach between the models. The *separation of concerns* principle was applied to disconnect the models, their data and a underlying simulation framework as clearly as possible. The *Chameleon* ideas have been realized on top of the integration frameworks *ModelCenter* and *Remote Component Environment*. The feasibility and the advantages of this concept will be pointed out in this paper. After discussing our experiences with drawbacks and merits of the currently used commercial framework and the transition to an open-source framework we give an outlook on future topics, which are relevant for a simulation software integration in scientific collaboration on a daily basis.**

***Keywords*—Scientific simulation; integration; workflow; collaboration; framework.**

## I. Introduction

Interdisciplinary collaboration in scientific simulation, modeling and exploratory research is a common activity among scientists in universities, companies and federal institutions. The requirements for this kind of locally distributed cooperations between researchers from very diverse fields of science are well-known [1], [2], and many software systems try to fulfill them and provide researchers with tools such as integrated environments, simulation data browsers and provenance explorers to deal with the associated technical challenges in scientific knowledge accumulation.

To our knowledge, there is no all-in-one solution available yet, but rather a lot of specialized and/or experimental academic tools, and a huge amount of commercial products that often fail to address the specific needs of scientific data management, flexibility to change workflows at any time, real-time monitoring and powerful post-processing facilities. Therefore, aside from similar solutions for this vast software field like Keppler & Taverna [1], [3], iSight [4], and many more, also at the German Aerospace Center (DLR) efforts were undertaken to create a suitable system, originally in the field of aircraft predesign, later adapted for use in more loosely related fields such as assessment of air traffic climate impact in the DLR-project *climate-compatible air transport system (CATS)*, innovative concepts in engine design in *Evaluation of Innovative Turbo Engines (EVITA)*, and of national and international air traffic modeling *(IML2)*. The software system created at DLR and presented in this paper certainly does not want to be the future all-in-one solution described above, but strives to reach a new level of conceptual and implementation abstraction and decoupling of its inter-component relations.

As already presented in previous papers [5], [6], there has been software development going on at DLR to provide researchers in aerospace-related fields with tools necessary to run distributed simulations and experiments on top of the existing commercial integration framework *ModelCenter* [7]. Use cases in scientific software integration were already presented elsewhere [8], [9]. In this paper we present a generalization of the simulation environment architecture we developed over the last few years.

The paper is organized as follows: Section II gives an overview about what we call the *"Chameleon-principle"* and shows the advantage of using a generalized and decoupled approach in scientific simulation workflow systems. Section III describes the original simulation environment set-up we used and outlines the main advantages and challenges with regards to the tenets of the *Chameleon* principle. In Section IV the new simulation environment is introduced with its (dis-)advantages regarding *Chameleon*. Section V draws a conclusion including a brief evaluation of current outcomes and how to proceed with further development to steadily increase the usefulness and universality of our software design for researchers and engineers.

## II. The Chameleon simulation environment

In short, *Chameleon* is the name of a concept for simulation and scientific software integration environments, targeted at experimenting and collaborating in interdisciplinary scientific simulation, modeling and assessment.

During the DLR-projects TIVA, TIVA-II, UCAV-2010, CATS and EVITA [10] work on a generalized simulation environment began. Those efforts culminated recently in a product called "Chameleon Suite", which denotes rather a design principle than a singular software product, because there are several ways that *Chameleon* can be (and was to be) turned into executable software. Since commencing development it was determined that for the software to succeed there must be a modular and highly abstracted software design, since the projects listed above all stretched over several years and follow-up projects needed and need to be able to build upon the existing software product, while new requirements may arise at any point. The design therefore follows the principle of *separation of concerns (SoC)* to allow for later changes in any separate part without inflicting consecutive adaptations in the other parts. Figure 1 shows the general idea of separating three distinct parts of a simulation environment, namely the scientific simulation applications (from now on called "tools") with their respective specific input / output demands, the common data format *Common Parametric Aircraft Configuration Scheme (CPACS)* [11], the software libraries that can be used by framework components as well as users' applications, and the software integration framework (also called "platform"), on which our software is built on.

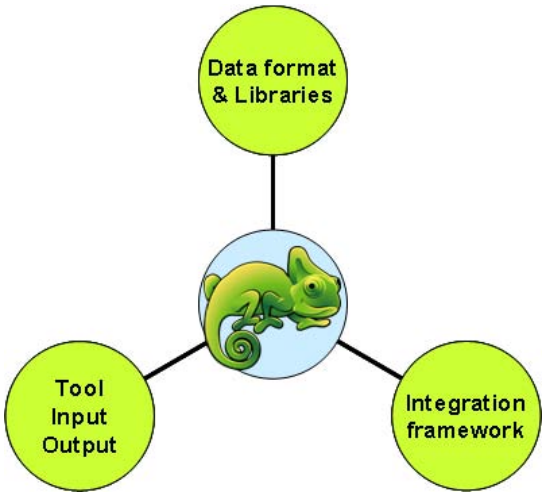


Figure 1. Separation of concerns by decoupling core components in the Chameleon simulation environment.

### A. Scientific tool integration

The integrated simulation tools need to be separated from the integration platform as well as from common functions (externalized into reusable software libraries) and the data format used to communicate between tools. Also, the input and output data formats of integrated tools may differ from the data format used to transfer information *between* tools, when they are called from a workflow engine that controls the execution flow, which is most often provided by the integration framework.

### B. The common data format

The data format and its underlying data model as the original driving core component, originally launched in the TIVA project, was specifically designed *independently* of any implementation details, to capture the structural and parametric description of aircraft configurations. The data format is wished to progress and extend independently from the integration environment on its own, although often changes arise from current project demands. Changes to the data format must only conform to the data integrity and domain constraints, but not to the technical realization of any tool using it. This way progress in disciplinary research is encouraged and not hindered by dependencies on libraries and existing integration state. Chameleon offers components to work with CPACS data; this contains importing, exporting and updating, spanning over several XML-files, including remote web-resources.

### C. The software integration framework

The software integration framework is the biggest asset but also often the most aggravating lock-in for every collaborative simulation, or other workflow environment. In order to keep even this heavy-weighing part on which everything else rests upon interchangeable, we developed all other core components in a way to stay decoupled and independent of the underlying framework. This approach seems quite bold, because frameworks differ largely in their capabilities regarding parallelization, workflow, numeric optimization, infrastructure, architecture, but also their supported data types, structures and runtime models. Regardless of this, we will show that it is possible to abstract a simulation framework from the underlying integration platform while not only retaining a useful and usable workflow and experimentation system, but also gaining all features that emerging frameworks may offer.

### D. The common software libraries

As shown in Figure 2, there are two helper libraries for data manipulation and direct data access, namely *TIXI* and *TIGL*. They have been named after the TIVA project for historical reasons.

The first supporting library that was developed was the *TIVA XML Interface (TIXI)*. Since CPACS data is completely based on XML, to simplify the access to the central data format the TIXI library has been developed to shield the application developer from dealing with the complexities of XML structure handling. Dealing with the full functional range of XML is mostly not necessary because many applications use input files based on quite simple data structures. These data structures are often single floating point or integer numbers, and their meaning depends on the exact

position of these numbers in the input or output file. More advanced types of these files are name/value pairs or lists of numbers to reflect vector or array data. On top of the full-fledged XML-library *libXML2* [12], the C-library TIXI was designed to shield the developer from XML processing when performing simple tasks like writing an integer or reading a matrix.

A second dynamic library we developed is called *TIVA geometric library (TIGL)* and can be used for easy processing of geometric data stored inside CPACS data sets. With TIGL it is possible to directly execute geometric functions on fuselage and wing geometries. In the future, the functional volume of the TIGL library shall be extented to handling of other aircraft parts like for instance engine nacelles. The geometric library itself uses again the TIXI library to access CPACS data sets, while leveraging data manipulation of all supported geometry types to, e. g., build up a prevalent 3D-Model of the contained aircraft in memory. For the time being only wings and fuselages are supported, with more to come. The functional range of the library goes from the computation of surface points in Cartesian coordinates up to export of airplane geometry to different file formats (IGES, STL and VTK). Beside these computational functions, TIGL can be used to obtain information about the geometric structure itself, for instance how many wings and fuselages the current airplane configuration has.

The overall architecture regarding those mostly independent software parts is displayed in Figure 2.

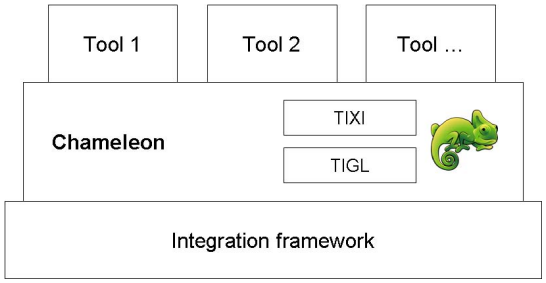


Figure 2. Coarse depiction of Chameleon's architecture.

### E. Advantages of the Chameleon approach

The general idea of our decoupled architecture is to simplify the technicalities connected with distributed computing and provisioning of highly specialized simulation tools of different scientific fields as much as possible. Therefore, the simplicity to set up and configure workflows and to introduce new tools into the environment was one of the main drivers during development.

The second most important advantage of the Chameleon approach is the additionally gained *"business"* value stemming from the decoupling principle: Every newly integrated tool that is based on the abstraction of core components as explained above instead of on legacy technology provides an instant bonus to current and future simulation projects because it is abstracted in a way that allows not only its reuse in other projects and contexts, but also in other integration frameworks and with other data formats. This is, in a nutshell, the main advantage of adhering to a strict SoC in a scientific workflow environment, that, to our knowledge, is unprecedented. Interfacing with a common data format increases interoperability and cooperations become easier to start with each newly added tool. Inclusion of many tools into one environment boosts productivity and reproducibility of results. Validation capabilities are much stronger when working with workflows instead of plain, manually set up process chains (like batch files or hand-crafted converter scripts).

Additionally to the separation of core parts in Chameleon, a host of different helper components is deployed for every Chameleon version, spanning convenient helper components for finding errors, sending email-notifications (useful in long-running workflows), graphical components for data extraction, data mapping, three-dimensional visualization plugins, script language integration, logging and more. These components cannot always be copied as they are from one Chameleon instantiation to another Chameleon implementation based on another integration framework, but they are rather provided, pre-manufactured and supported by the *Chameleon* team to comprise a comprehensive collection of tools needed to design scientific workflows, experiment with domain values and exchange data and ideas with colleagues. The usage of the libraries, of the data format, and most of the workflow features are virtually just the same, from a user's perspective, regardless of the platform used.

In the following two subsections we will present the currently supported integration frameworks that Chameleon was developed to run on, along with their respective advantages and drawbacks.

## III. Phoenix ModelCenter

The simulation environment *ModelCenter (MC)* is a commercial product from Phoenix Integration, a software vendor located in Philadelphia, USA. MC is based on a client / server architecture where the *ModelCenter* application itself is a client-side *Microsoft Windows* application to build and monitor simulation workflows. On the server-side an application server called *Analysis Server* is used to provide the software infrastructure for distributed execution of simulation models. Alternatively, a product called *CenterLink* can be used, which allows asynchronous and queued execution of packaged models independently of the MC client. Simulation models can be constructed as scientific workflows in MC and run either in the the integrated workflow environment or on the above-mentioned server. A scientific workflow can be either a data- or process-driven tool chain with capabilities similar to the *Business*

*Process Modelling Notation (BPMN)*. The following sections enumerate the key advantages and drawbacks of MC when used as the foundation for the Chameleon simulation environment:

### A. Advantages

- Tool based simulation approach; application wrapping
- Runtime tool execution depending on fixed number of input- and output *variables*, synchronous data transfer
- Data- and process-driven approach available
- Choice of several workflow-driving schedulers
- Many integration components included: Optimizers, Calculation software integration (e. g., , Excel, Matlab, ANSYS)

### B. Disadvantages

- Even moderately large data sizes not supported
- No integrated support for XML-based data formats (yet), although internally running on XML-technology
- No server-based GUI for the individual tools providable (availability not foreseeable in the future)
- No runtime support for wiring up complex workflows (e. g., wizard functionality)
- Hard to conceive ways on how to add support for collaboration in teams
- No possibility to add additional cross-cutting technologies like, e. g., provenance, data management, user access rights; may be included in an upcoming version

On the one hand, the listing shows that the simulation environment comes with all general functionality to work with the basic idea of *Chameleon* in combination with simulation workflows. But on the other hand, the disadvantages bar us from building a really integrated simulation environment, which addresses the needs of researchers regarding a full-fledged collaboration software. *ModelCenter* is a useful software for creation of simple simulation workflows, but a simulation environment should address not only the process, but also potentially complex and dynamic data (-management) and the collaboration issues.

## IV. Remote Component Environment

The integration framework *Remote Component Environment (RCE) [13]*, formerly known as *Reconfigurable Computing Environment*, is an open-source product from the DLR institute *Simulation and Software Technology (SC)* [14], department *Distributed Systems and Component Software (VK)* and the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI). *RCE* is a grid component framework for distributed computing based on the *Eclipse Rich Client Platform (RCP)* and provides core functions needed in a distributed environment. These features, among others, are:

- Authorization and authentification (AA)
- Detaching of running workflows from the client's GUI
- Operating system independency due to use of the RCP and Java platform (Linux, Windows, Solaris, AIX, Mac)
- Modular framework based on OSGi [15] allows for further enhancements regarding emerging technologies, e. g., provenance and knowledge management
- Inter-instance file transfer and notifications
- Visual workflow editor
- Data management
- Distributed logging view

RCE was originally designed in a project concerned with the predesign of ships in collaboration with dockyard companies. Because predesign processes for ships have a lot in common with the predesign tasks for aircraft, RCE was a natural candidate to create a Chameleon environment on. Similar to the structure of the previous section, an examination regarding advantages and disadvantages follows:

### A. Advantages

- Tool-wrapping for simulation tool integration
- Runtime tool execution based on a data-channel concept, leading to asynchronous data transfers
- Data-driven approach, allowing streaming of input and output data values
- GUIs allowed for server-side simulation tools, accessible from every RCE instance in a network
- Support for wiring up complex workflows (wizards)
- Possibility to extend the system for collaboration tasks
- Open architecture allows to add additional technologies like provenance, data management and visualization

### B. Disadvantages

- No process-driven approach available yet
- No workflow-scheduling approaches, logic is currently implemented only in the integrated components
- Young product; needs an active community, code committers and operational experience

It is obvious that *RCE* is able to execute simulation workflows as well. The main advantage of *RCE* is the flexibility of being built on top of the open source RCP and OSGi platforms and therefore having the opportunity to add collaboration topics and additional technologies to it. For institutions like DLR, working on strongly multidisciplinary projects with locally distributed teams of manifold professions there is a need for supporting and integrating new technologies in addition to create and execute workflows whenever need arises.

## V. Conclusion and future work

The transition from a data format and tool integration in ModelCenter – which was developed over several years and in several projects as mentioned in Section I – to the new integration framework RCE, was prototypically exercised and evaluated. The software development overhead was quite small in comparison to the former projects, because

of several facts: The data format and its accessor libraries were already created, tested and broadly in use. Installation overhead is minimized, since the infrastructure is already in place and cooperation in server configuration is widely given and responsibilities are accepted. The component framework offered by the OSGi-platform reduces the overhead ("boiler-plate code") needed to create and integrate new components. The shortcomings of the existing framework were largely known and many workarounds had been in place. Therefore in the new framework we could often directly use its capabilities to reach the same goal without having to resort to a workaround. In many parts, the RCE framework had differing concepts and a different philosophy to the definition of workflow. Here the challenge lay in the fact that most logic for RCE integration goes into the components instead of the workflow engine. We hope that upcoming versions of RCE will have more powerful features to ease the developer's overhead to integrate components in a useful and usable way.

In this paper we have shown the advantage of a generalized and decoupled concept for integrating data, tools and frameworks. We demonstrate how easily our *Chameleon* principle can be adapted to the open source grid computing framework *RCE*. The reason for changing the framework is, beside the proof of generalizability, that the new framework is a more eligible candidate for a real simulation system. This is because we learned that collaboration in teams and support of users in creating a workflow is much more important than just to execute the same process over and over again without changes. The wide field of collaboration is, beside the core integration technology, the next main approach we want to address in our *Chameleon* software suite. Our future work will be about the integration of provenance information and to provide a possibility to semantic checks during creation of workflows.

## References


[1] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[2] T. Schlauch and A. Schreiber, "Datafinder. a scientific data management solution," in *PV 2007*, Oct. 2007. [Online]. Available: http://elib.dlr.de/53369[Online;2010-07-19]

[3] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1067–1100, 2006.

[4] "SIMULIA iSight website," http://www.simulia.com/academics/isight.html, [Online; 2010-07-19].

[5] A. Bachmann, M. Litz, M. Kunde, and A. Schreiber, "A dynamic data integration approach to build scientific workflow systems," in *Proceedings of the 4th International Conference on Grid and Pervasive Computing*, vol. 1, Geneva, May 4. – 8. 2009.

[6] A. Bachmann, M. Kunde, M. Litz, A. Schreiber, and L. Bertsch, "Automation of aircraft pre-design using a versatile data transfer and storage format in a distributed computing environment," in *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on*, Oct. 11. – 16. 2009, pp. 101 – 104.

[7] "Process integration & design optimization," http://www.acel.co.uk/pdfs/designsimulation/, [Online; 2010-03-19].

[8] L. Bertsch, G. Looye, T. Otten, and M. Lummer, "Integration and application of a tool chain for environmental analysis of aircraft flight trajectories," in *The 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Sep. 21. – 24. 2009.

[9] C. M. Liersch, T. Streit, and K. Visser, "Numerical implications of spanwise camber on minimum induced drag configurations," in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, Orlando, Florida, USA, Jan. 5. – 8. 2009.

[10] "DLR project websites," TIVA: http://www.dlr.de/fa/desktopdefault.aspx/tabid-1474/2079_read-3561/, TIVA-II: http://www.dlr.de/fw/desktopdefault.aspx/tabid-2959/4455_read-20454/, UCAV-2010: http://www.dlr.de/sc/desktopdefault.aspx/tabid-5141/8654_read-11626/, CATS: http://www.dlr.de/pa/desktopdefault.aspx/tabid-4618/, [Online; 2010-07-19].

[11] D. Böhnke, "Dataintegration in preliminary airplane design," Master's thesis, Universität Stuttgart, Jul. 2009, Work in progress, to be published in Sep. 2009.

[12] "libxml2," http://www.xmlsoft.org, [Online; 2010-04-01].

[13] D. Seider, "RCE homepage," http://www.rcenvironment.de, 2008, [Online; 2010-07-19].

[14] "German Aerospace Center website," http://www.dlr.de/sc, [Online; 2010-07-19].

[15] "OSGi Wikipedia entry," http://en.wikipedia.org/wiki/OSGi, [Online; 2010-07-19].