AST 2013
Workshop on Aircraft System Technologies

ARCHITECTURAL ANALYSIS OF COMPLEX SYSTEMS WITH GRAPH-BASED DESIGN LANGUAGES

Stephan Rudolph*[1], Stefan Hess[1], Johannes Beichter[1]
Martin Motzer[1] and Marc Eheim[1]

[1]Similarity Mechanics Group
Institute for Statics and Dynamics of Aerospace Structures, University of Stuttgart
Pfaffenwaldring 31, D-70569 Stuttgart, Germany

rudolph@isd.uni-stuttgart.de

Abstract
Graph-based design languages are a new way of supporting the activity of engineering design throughout the whole product life-cycle [1] with means of modern information technology. Through the machine-executed compilation of design languages in a design compiler, the automated generation of multi-disciplinary consistent models is now a reality [11, 12]. The automatic model generation relieves the engineering design teams from tedious routine works, allows know-how re-use of design knowledge by re-use of design rules and generates complex topological and parametrical system design variants by the press of a button. The work describes several important aspects in an ongoing industrial research project for engineering aircraft cabins using graph-based design languages. The aircraft cabin design problem comprises the following engineering aspects: definition of the cabin layout, generation of the 3-D geometry model, integration of air-conditioning system, power and information network architectures as well as packaging, routing and dimensioning of all involved components, while still insuring overall design model integrity and consistency.

1 INTRODUCTION

Architecting complex systems, i.e. the decision making on the architecture of aircraft cabins requires much engineering insight into the heavy multi-disciplinary coupling of the combined and interconnected sub-systems [5] and networks inside the cabin. Due to this heavy coupling inside the cabin, there existed in the past a certain tendency to stick within the “experience space” of (more or less well) known solutions, since within the built architectures there exists an (more or less sophisticated) established way how to deal with the couplings and to resolve it in several design steps.

As a direct consequence, the potential of new architectural solutions in a complex system design environment is difficult to assess *a priori*, since for an equally in depth assessment it would be required to elaborate all architectural alternatives with the same qualitative and quantitative level of engineering analysis as is available *a posteriori* for the cabin architectures built in the past. However, an in depth investigation of several competing alternatives using conventional engineering analysis using human labor techniques is economically not feasible. As a result, the potential of new cabin architectures is neither fully explored nor tested. Instead, conventional solutions which may not represent the optimal solution anymore are repeatedly built.

In order to overcome this kind of highly unsatisfactory deadlock situation, graph-based design languages have been conceived to solve this kind of complex systems engineering problem. Graph-based design languages relieve the design engineering teams by automatic model generation from tedious routine works, allows know-how re-use of design knowledge by re-use of design rules and allows very easily topological and parametrical design variations. Since faster time-to-market, higher product quality and doing things already first time right are key success factors, there exists great interest in new knowledge-based engineering tools in the aerospace industry (i.e. The Boeing Company [3], Rockwell Corporation [8], Airbus Industries [5] and others). In this respect graph-based design languages can be interpreted as a modern, rule-based extension of well-established engineering tools and practices.

This report illustrates several important aspects of a research project between an aircraft manufacturer and the Similarity Mechanics Group at Stuttgart University regarding engineering complex systems such as an aircraft cabin. The aircraft cabin design problem comprises among others [5] the following engineering problem aspects: definition of the cabin layout, generation of the 3-D geometry model, integration of the air-conditioning system, power and information network architecture, packaging and routing of all involved physical components while still insuring overall design model integrity and consistency.

The report is structured as follows: after the introduction in chapter 1, the representation and knowledge processing mechanisms of design languages are described in chapter 2. The main steps of aircraft cabin design are described and illustrated in chapter 3. The result discussion and summary closes the paper in chapter 4.

2 KNOWLEDGE REPRESENTATION AND PROCESSING IN GRAPH-BASED DESIGN LANGUAGES

Graph-based design languages are a novel means of supporting the activity of engineering design throughout the whole product life-cycle with new knowledge processing technologies borrowed from engineering, computer science and artificial intelligence. From *engineering science* stems the *systematic design methodology* as introduced by Pahl and Beitz in the early 1970s [7]. In this methodology, product *requirements* are systematically transformed into abstract product *functions*, from there transformed into solution *principles* and finally transformed into product *equipment*.

In *computer science*, the idea of a sequence of mappings was abstracted and adapted for code generation purposes. The philosophy underlying the *model-driven architecture* (MDA) which advocates a sequence of *model-transformations* for code production starts now dominating software engineering projects. In the *artificial intelligence* field, automated *constraint processing* techniques have been developed which help to maintain parameter information consistency over a constraint set representing intercoupled models. In the area of *systems engineering*, ways of dealing with couplings in complex systems by means of modern *model-based systems engineering* (MBSE) methods represents one of the most up-to-date trends in the field.

Graph-based design languages merge all these principles from the aforementioned disciplines into a generic design methodology for engineering complex systems which guarantees a consistent design information representation. Graph-based design languages are hereby inspired by our natural languages, where the design *vocabulary* (i.e. the language words) and the design *rules* (i.e. the building laws) define a so-called language grammar [4]. This means that a permissible sentence in this language (i.e. a permissible vocabulary combination) represents a valid engineering design. Through the machine-executed compilation of such a graph-based design language in a design compiler [2], a very powerful framework for engineering design can be established.

2.1 Vocabulary

As an information representation format for graph-based design languages the unified modeling language (UML) [6] is used. UML allows a flexible, easy to convert in other formats and extendible engineering information representation in an internationally standardized, non-proprietary and open format. Furthermore, a multitude of free, open-source and commercial software tools for editing, storing and displaying UML are available. In terms of an engineering design language, the individual vocabulary is represented as UML classes. The information associated with a specific vocabulary is stored as attributes and methods of that class. The existing (static) interdependencies between the vocabulary are stored in the UML class diagram using principles from object-oriented modeling such as information hiding, encapsulation and inheritance.

2.2 Rules

In a graph-based design language as it is used in this work, the (dynamic) dependencies between different design vocabulary is expressed in form of a rule following an *if*→*then*-scheme. In UML, such a rule can be represented as a *model-to-model* (M2M) or a *model-to-text* (M2T) transformation, depending on whether the outcome remains a UML model or is converted into another string-based text format. The left-hand side (LHS) as *if*-part states the precondition(s), while the right-hand side (RHS) as *then*-part states the postcondition(s) of the model transformation.

The power of the graphical rule format for engineering design knowledge is twofold: First, the rule is a unified scheme for topological as well as parametrical model modifications. Second, the graphical representation of UML offers an intuitive way to express the topological changes in a graph-based instead of a string-based format.

2.3 Production Systems

A sequence of rules can be stored in a production system (in UML called activity diagram). These production systems may be hierarchically nested (i.e. that a production system may call during its execution another production system) and allow the structuring of rules into different modules. Each production system has a unique start and end point of the execution. Rules can therefore very easily grouped according to the semantic meaning of their content. Whole chunks of design rules which represent whole design activities (such as packaging a given set of geometric objects in a building space or the routing of a given set of cable connections in a predetermined routing space) can thus be clustered and grouped together in a meaningful way.

The approach of graph-based design languages offers the great advantage that a production system (i.e. the design rules contained therein) can be easily modified and re-executed automatically, thus resulting in a different kind of product architecture. Due to the fact that besides the changed rule the remaining design rules remain unaffected, the resulting different aircraft architectures are built qualitatively and quantitatively with the same level of engineering detail. This makes the subsequent engineering simulation analysis and design evaluation of alternative competing aircraft cabin architectures straightforward, sound and robust.

2.4 Processing

The compilation of the graph-based design language results internally in a central and consistent model (i.e. the so-called design graph) from which disciplinary views can be generated automatically via interfaces to dedicated engineering analysis tools. The existence of a central model reduces the amount of necessary bidirectional interfaces (also called plugins) from $n(n-1)$ to n [9]. Furthermore the guarantee for consistent models is computationally easier to maintain if all models are generated from one central model using constraint processing mechanisms [10].

Currently, the following interfaces in the design compiler [2] are used for engineering analysis: a CAD interface for CATIA V5, a CFD interface to FLUENT and a company-owned legacy code, a control software interface to MATLAB/SIMULINK, a computer-algebra system (CAS) interface to MATHEMATICA and an interface for MS-EXCEL. Further analysis tools could be added through additional interfaces in the design compiler in an easy and straightforward manner.

Graph-based design languages are also a means to generate models for virtual reality (VR) automatically. VR as a technique for immersive visualization of complex 3-D geometry and post-processing of complex engineering analysis results is therefore ideally suited as a *back-end* for the visualization of computational results. Since design languages are represented graphically in UML, the "programming" of graph-based design languages could also be done graphically using the VR as the programming *front-end*. In the near future, graph-based design languages may thus offer the potential to fully support complete iterative design cycles (i.e. design optimization loops) in a VR environment including interactive rule modifications, renewed model generation and simulation analysis leading to an updated design evaluation.

3 AIRCRAFT CABIN DESIGN

The design of an aircraft cabin involves many disciplines. As the current state-of-art it can be mentioned that each discipline (industrial design, continuum mechanics, fluid mechanics, electronics and so on) understands its own (disciplinary) view on the system. The multi-disciplinary integration of all these different models into one single and consistent "master model" is currently in industry an unsolved open problem.

Graph-based design languages handle the consistency problem quite differently: Instead of trying to solve the integration problem by merging different models into one single master model, all different disciplinary models are automatically generated from a central model and are for this reason consistent by definition.

The price for this achievement is the formulation of aircraft design knowledge in form of a design language in UML format. As far as the engineering is concerned, the design of an aircraft cabin involves (among many others which have to be omitted here for the reason of space) the following important steps:

- definition of the cabin design variants and their boundary conditions
  (plane dimensions, number of frames, definition of crown area and so on)
- definition of cabin architecture and layout
  (number of classes, number of seats, definition of various ratios)
- definition of networks and routes
  (the network is generated through an algorithm, while routes which are intentionally pre-determined are read in from a database)
- definition of network components and equipment
  (currently the equipment is read in from a database)
- definition of the network
  (an algorithm is used to connect equipments appropriately. The corresponding connection information is stored in a dedicated UML class diagram)
- routing of electrical cables in 3-D
  (routing includes arbitrary routing space boundaries and collision testing in 3-D)
- integration of routing result into 3-D geometry model
  (through the 3-D model can be navigated in a virtual environment)
- results computation and architecture trades
  (for comparison of design variants, the weight of the cables or the diameter of the routes in the cross sections at the frame positions is calculated and exported to MS-EXCEL or CATIA V5 for further processing)
- design modifications
  (changing the input and re-executing the above processes allows the exploration of the design space through a comparison of two or more variants)

In the currently ongoing industrial research project, several smaller design languages for different purposes are combined together into one large design language to generate the aircraft cabin models. This combined design language contains all the necessary design information in an (re-)executable form.

Figures 1 and 2 show exemplarily how the geometry of the fuselage is built in an incremental way. In order for the rule in figure 1 to be executable, an instance `plane` has to exist. In this case the design rule creates the instance `spant0` and links it to

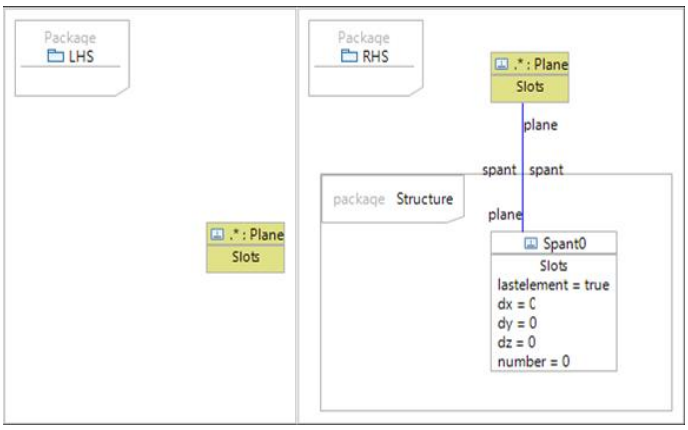

Figure 1: Design rule: Adding `spant0` *to* `plane`

the instance `plane`. Similarly, figure 2 shows the design rule how further instances of the class `spant` are added using a counter supposed it is the instance with the slot `lastelement=true`. In this way, a linked list of aircraft frames (in german *Spant*) is created. The hierarchical and modular decomposition of the overall aircraft cabin

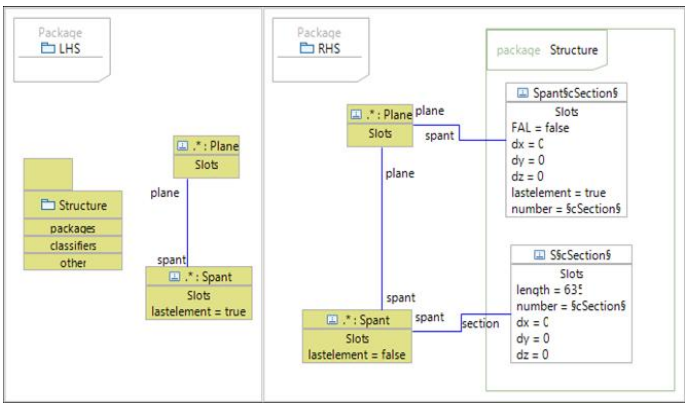

Figure 2: Design rule: Adding `spant0` *to* `plane`

design task allows the encapsulation of individual design tasks into specialized design languages. By combining the cabin layout language with the geometry generation language, a 3-D model of the aircraft cabin can be generated. Adding the routing language to the network generation algorithm allows the integration of the electrical networks into the crown area of the aircraft. This will be shown in the following sequence of figures 3 to 6 which show the intermediate results of the different steps.

Generating a class layout for an aircraft cabin is a task with involves the concurrent satisfaction of many constraints. For the multidisciplinary integration problem of the necessary networks for air, power and data into the crown area of the aircraft the cabin layout represents a significant part of the boundary conditions definition.

Figure 3 shows a simple layout editor which allows to manipulate the seat layout in an aircraft cabin. In fact, the layout of an aircraft cabin consists of about one hundred design rules which have been omitted here for the reason of space. The cabin layout shown here implicitly defines many of the connection coordinates of the starting and ending points of the networks and is therefore only shown to illustrate the decomposition of a complex design process into individual design steps.

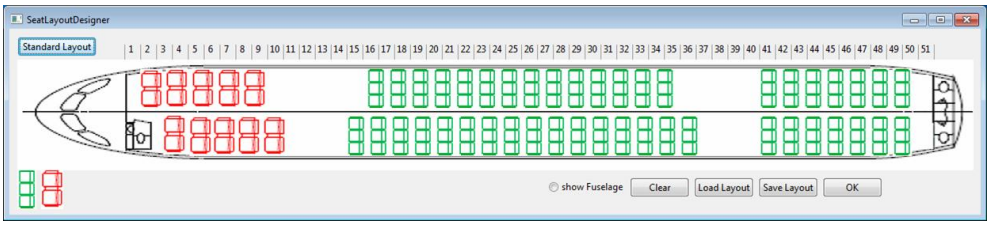

Figure 3: Screenshot of interactive cabin layout editor

Since networks typically posses an infrastructure of internal (re-)distribution nodes and interconnections, the determination of an "optimal" network represents a non-trivial optimization problem. As a result, the positions of all components (shown as different colors) of the internal network nodes are shown in figure 4.

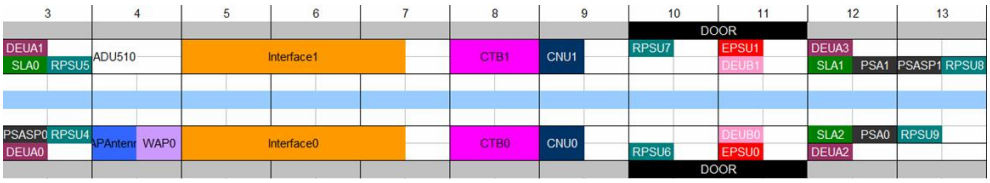

Figure 4: Network components placement in aircraft crown

Not shown in figure 4 is the position of the primary energy supply unit. Its position may be changed during the design, since the overall properties of the network (e.g. cable length, weight) may depend on the placement of this unit.

Together with the information from the cabin layout (see figure 3), from which a complete 3-D geometry model can be generated and the placement information of the network equipment including the harness information (see figure 4), the integration of the networks for power and information can be started. In a first step, the generation of the 3-D geometry model is shown on the left hand side in figure 5. From this 3-D geometry model, the routing space on the right hand side in figure 5 is automatically constructed.

In the second step the network boxes are placed in the routing space, see figure 6 left. A routing algorithm is then used to create the cable connections according to a harness plan loaded from a database. The routing algorithm guarantees a shortest path search and includes collision testing. The routing results are shown in figure 6 right.

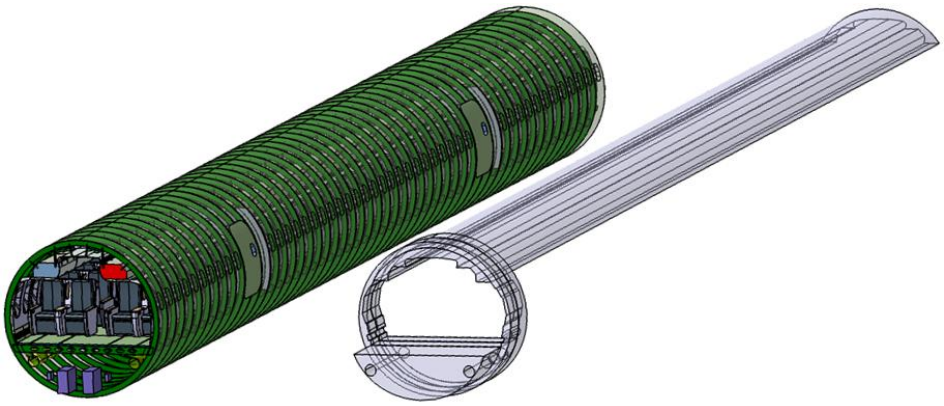

Figure 5: Extraction of routing space (right) from aircraft cabin CAD model (left)

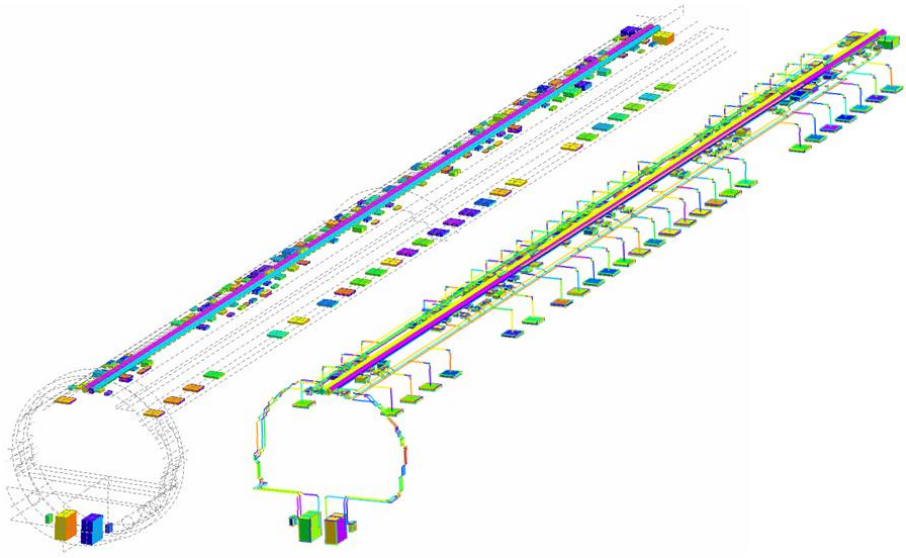

Figure 6: Network components (left) routing algorithm result (right)

From figure 6 the two network quantities total *weight* and total *network cable length* can be easily computed and may be used as performance measures for the comparison of the two generated aircraft designs. The two functionally equivalent networks do however differ in the different internal distribution infrastructure. While network version 1 has 8 internal nodes, network version 2 has only 4 internal nodes. The internal weight distribution is shown in figure 7.

Since the 3-D model in figure 6 is computationally accessible any information can be extracted. One finds that the length of network 1 adds up to $558,83$ meters while network 2 has a length of $582,50$ meters. The two equipment weights sum up to $404,22$ kg (for version 1) versus $364,22$ kg (for version 2) and the total weight of equipment and cables sums up to $963,05$ kg versus $946,72$ kg respectively.

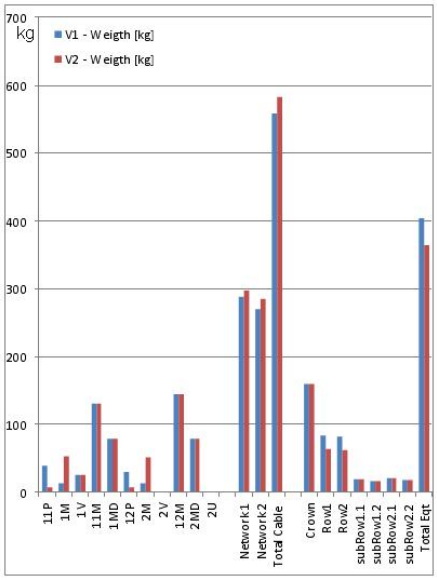

Figure 7: Weight distribution of network components and cable routes

4 SUMMARY AND OUTLOOK

A consistent central design model which covers several distinct engineering disciplines such as a CAD-model (for/from construction in 3-D), a continuum mechanics model (from/for stress analysis), a fluid mechanics model (from/for flow analysis), a control model (from/for control design) and so on may be achieved by using graph-based design languages. In this work, the generation of a consistent central model for the multi-disciplinary analysis of an aircraft cabin has been shown. The analysis included the generation of the class layout (seats, lavatories and so on) and showed the integration of several routes of a power and an information network. The routing algorithm guarantees a shortest path search and includes collision testing.

Two alternative networks with a different infrastructure for power and information distribution were automatically constructed through compilation of the design language in a design compiler. As a result, the two network quantities total weight and total network cable length were selected as performance measures for the comparison of the two generated aircraft designs. By extending this approach in the future, the *Pareto surfaces* of competing aircraft cabin architectures can be automatically constructed, thus serving the cabin architect as a solid guideline for a justified decision making in complex future multi-disciplinary aircraft cabin design scenarios.

References

[1] P. Arnold and S. Rudolph, ”Bridging the gap between product design and product manufacturing by means of graph-based design languages”, TMCE 2012 Symposium, Karlsruhe (2012)

[2] Design Compiler 43, Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH, www.iils.de, last access March (2013)

[3] J. Heisserman, R. Mattikalli and S. Callahan, ”A grammatical approach to design generation and its application to aircraft systems”, Proceedings Generative CAD Systems Symposium ‘04, Pittsburgh, PA (2004)

[4] B. Kröplin and S. Rudolph, ”Entwurfsgrammatiken Ein Paradigmenwechsel?”, Der Prüfingenieur **26**, 34-43 (2005)

[5] B. Landes and S. Rudolph, ”Aircraft cabin architectures including tolerancing using a graph-based design language in UML”, Proceedings Deutscher Luft- und Raumfahrt Kongress 2011, Bremen, 27.-29. September (2011)

[6] The Object Mangament Group (OMG), *OMG Unified Modeling Language (OMG UML)*, Superstructure V2.4.1, OMG Document Number 2009-02-02, http://www.omg.org/spec/UML/2.2/Superstructure, last access August (2011)

[7] G. Pahl and W.Beitz, *Engineering Design*. (Springer, London, 1996)
Originally published in German: G. Pahl und W. Beitz, *Konstruktionslehre*. (Springer, Heidelberg, New York, 1993)

[8] S. Reddy, K. Fertig and D. Smith, ”Constraint management methodology for conceptual design tradeoff studies”, Proceedings of 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference, paper DTM-1228, August 18-22, Irvine, CA (1996)

[9] S. Rudolph, ”On design process modelling aspects in complex systems”, 13th NASA-ESA Workshop on Product Data Exchange (PDE 2011), May 11-12, Cypress, California, USA (2011)

[10] S. Rudolph and M. Bölling, ”Constraint-based conceptual design and automated sensitivity analysis for airship concept studies”, Aerospace Science and Technology **8**, 333-345 (2004)

[11] J. Schaefer and S. Rudolph, ”Satellite Design by Design Grammars”, Aerospace, Science and Technology **9**, 81-91 (2005)

[12] S. Vogel, B. Danckert and S. Rudolph, ”Knowledge-based design of SCR Systems using graph-based design languages”, MTZ **73**, 50-56, August (2012)