Deutscher Luft- und Raumfahrtkongress 2011
DocumentID: 241461

TOWARDS EFFICIENT DEVELOPMENT METHODS FOR AIRCRAFT SYSTEM MODELS

Steffen H. Prochnow
steffen.prochnow@bauhaus-luftfahrt.net

Bauhaus Luftfahrt, Lyonel-Feininger-Straße 28
80807 Munich, Germany

ABSTRACT

Designing and developing new aircraft systems is time-consuming and expensive. An essential support in development of aircraft system models is the visualization of aircraft system properties. The model-based design supports this visualization. However, the modeling tools available so far do not support the modeling process in detail. They are rather limited in integration of different system models and in representation of complexity. This paper analyzes possible causes and resumes present results and approaches from different aspects in aircraft design and related topics.

1 GRAPHICAL MODELING OF AIRCRAFT SYSTEMS

System development based on graphical approaches has become standard practice in the last years' design for some aspects of aircraft system models. These graphical approaches enable users of graphical modeling tools to easily explore aircraft system models, analyze and calculate the aircraft system parameters. In today's development process there exist different graphical and non-graphical modeling tools for modeling different aspects of aircraft systems, *e. g.*, model development tools for structural and functional aspects, aerodynamics, compartments, electric systems, control, etc. General modeling approaches use currently used textual programming languages such as *e. g.*, *Java* or *Simula*, which provide internal program logic only [1] or also concepts like inheritance to the modeler [2]. For specific disciplines often specialized tools for modeling and simulation are used, as *e. g.*, *GSP* [2] for gas turbines, *AVL* [3] and *Tornado* [4] for aerodynamics, and *e. g.*, specific tools for aircraft configuration optimization [5] and for interdisciplinary optimization [6]. These tools generally provide a very restricted graphical interface. It often represents static modeled data as CAD model or simple representations of simulation results with no or marginal possibilities of interaction. Other tools focus more on the structure, together with the according parametric and component model, as *e. g.*, *Catia V5* [7] and the *Pacelab Suite* [8]. Such tools are intended to develop graphical views. Therefore they make extensively use of the what-you-see-is-what-you-get (WYSIWYG) paradigm. However, since the today's development of aircraft models is an distributed process there is no generic framework that integrates all development aspects of aircraft models.

Using development paradigms established so far often results in complex models that are distributed in different modeling tools with different modeling languages and modeling objectives. As a consequence the overall system model is difficult to comprehend and maintain. Moreover, potential errors can be very subtle and hard to locate in such complex systems for the human beholder. This severely comprises the practical use of graphical system models. We also argue that today's paradigms for editing, visualizing and simulation aircraft system models have not progressed significantly since the first perception of system models. The usefulness of graphical models depends to a large extent on their readability, which is the capability of the drawing to convey its meaning quickly and clearly.

In the following we analyze methods to support the easy development and understanding of complex aircraft system models. We focus on methodologies, which seeks to support the system developer in modeling, simulating and comprehending complex aircraft system models. Central to our analysis of modeling methods and techniques is a human centric modeling approach, which identifies weaknesses and deficiencies of today's aircraft systems modeling approaches.

The use of an integrated system architecture can be seen as an enabling method that allows the integration of different modeling objectives (structure, function, aerodynamics, etc.) into one integrated structure. Important advantages of an integrated structure are the overall analysis of the modeled system and ensuring the overall system consistency. The benefit of an integrated system model is that different additional resulting property views on the system can be automatically derived (*e. g.*, flight behavior, timing, etc.) This enables the system modeler to easily explore and inspect the overall system wrt. different technical aspects.

In this paper we will also examine the what-you-see-is-what-you-get (WYSIWYG) paradigm in contrast to the textual model development paradigm. Graphical system models can be seen as a very powerful concept for the development of aircraft system models because graphical models may be more convenient to browse. However, compared to textual entry, they are rather cumbersome to construct and maintain, as designers spend a significant fraction of their time with tedious drawing and layout chores. We will analyze the textual entry wrt. its modeling efficiency and give a perspective on how to benefit from both - the advantages of the textual and the graphical modeling paradigm.

1.1 Modeling Aircraft System Models

The complexity of aircraft system models results mainly from the diversity and the number of system components. Moreover the system interdependencies and the concomitant functional dependencies (dynamics) increase the overall system complexity. A human modeler can capture the structure of a graphical system model by the means of sequential traversing. First of all the modeler observes an entire graphical system model without any details. Thereupon, the system diagram is recursively refined by inspection of its sub-systems. However, the comprehension of a system under optimization (or a system under simulation) needs observation over a period of time. An aircraft system model generally consists of many interacting sub-systems. This leads to complex system dynamics and makes comprehension more difficult than conventional sequential composed system models that have only one locus of interest. An established approach for design and analysis of aircraft models is the creation of interdisciplinary and graphical system models, which easies the system validation, simulation, and automatic coherence analysis before a concrete prototype is developed and tested. Here different kinds of analyzes can answer some questions about the modeled system, *e. g.*, regarding composability of system components, structural validity and system behavior. An explorative system visualization and simulation is indispensable for a general comprehension of the system and system behavior. There are accepted paradigms and tools for visualization. However, they are very limited regarding the offered (static) views and usability and barely scalable for complex systems.

1.2 System Views

In the today's aircraft development process graphical system models are an established and accepted technique for definition and documentation of interdependencies of system sub-components as compositional element of the entire aircraft system. However, the views on sub-systems are generally represented as static (and inflexible) views, which limits the human analysis and understanding. Tufte [10] states that the problem is the low information density of the so far established visualization approaches. Hence, only small parts of the entire system can be visualized, even if an equivalent textual representation of certain model views needs less than one sheet of paper. However, with textual representations it is hard for the human modeler to establish a relation to the graphically modeled system. Another disadvantage of textual model representations is the loss of visual experience of a graphical model and its dynamics. The traceability of systems and their dynamics can be distinguished into two questions:

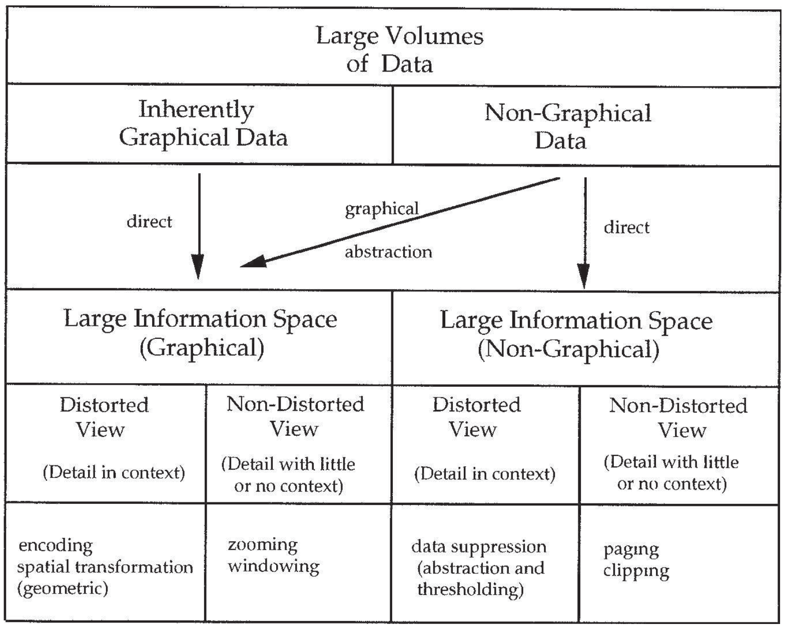

Large Volumes of Data
Inherently Graphical Data
Non-Graphical Data
direct
graphical abstraction
direct
Large Information Space (Graphical)
Large Information Space (Non-Graphical)
Distorted View (Detail in context)
Non-Distorted View (Detail with little or no context)
Distorted View (Detail in context)
Non-Distorted View (Detail with little or no context)
encoding spatial transformation (geometric)
zooming windowing
data suppression (abstraction and thresholding)
paging clipping

Figure 1: Classification of presentation techniques of large information sets (Leung and Apperley [9])

1. How does a modeled system behave?
2. Why does a modeled system behave the way it actually behaves?

For answering the first question it is less important to know details of a model. This particularly concerns human modelers, which developed themselves and being already familiar with the model. A good overview of the entire system is much more important. However, for answering the second question the human modeler needs detailed knowledge about particular sub-systems. A consolidated view indicates the need for information reduction about the system structure and to combine it with comprehensive information about system dynamics (*e.g.*, a system configuration, which can be reached during simulation.)

In the following, we distinguish between static views on a system model, which are not related to a certain system configuration and dynamic views with an underlying concrete system configuration, as a result of a system simulation run. A sequence of dynamic views will be treated as an animation of the actual system behavior. Moreover, we distinguish static system views for a given sub-system (that always represent the same information in the same fashion) and flexible system views that adapted the selection and representation of the underlying system information. Furthermore, we distinguish between local views and global views. The local views represent certain parts on its own and separated from the given context, meanwhile global views comprise the representation of the entire view. Conventional visualization paradigms are limited to local static views.

2 VISUALIZATION OF LARGE INFORMATION SETS

The visualization of large information sets coincides with a certain display area. This area generally limits the representable model information in parts of the modeled system or the level of modeled details. There exist a number of dynamic representations, which will be presented in the following. Generally there are two classical visualization approaches:

Zoom: enlarge optical or graphical section of relevant areas, and

Explode or hide: modify the visibility of hierarchical sub-systems.

Figure 1 shows a general classification overview of representation algorithms of visual information.

2.1 CONVENTIONAL ZOOMABLE USER INTERFACES

The mostly used techniques for diagram visualization is the combination of zooming and panning. The

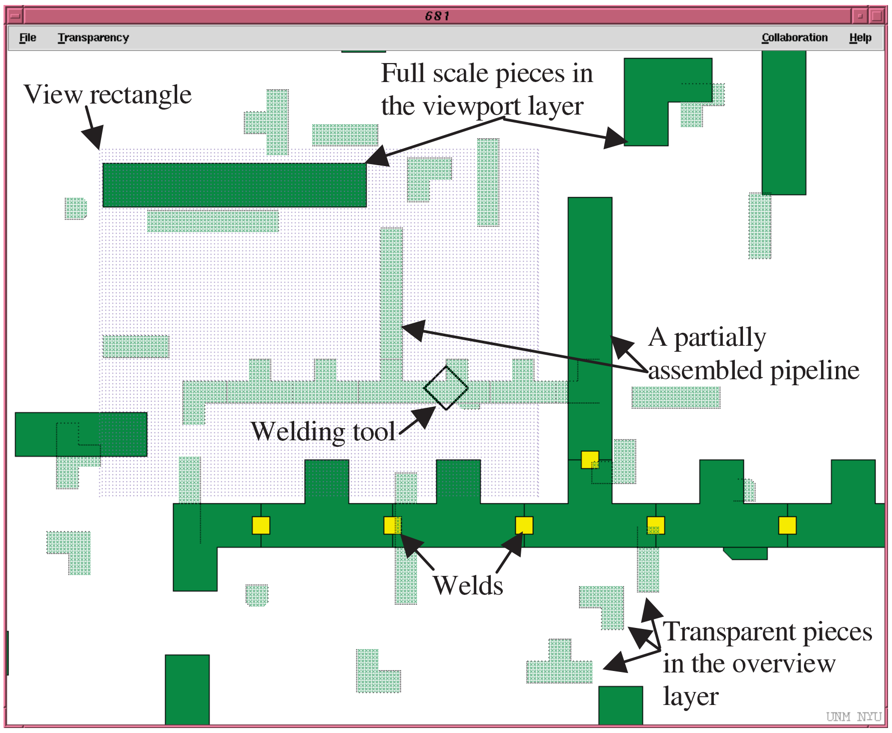

681
File
Transparency
Collaboration
Help
View rectangle
Full scale pieces in the viewport layer
A partially assembled pipeline
Welding tool
Welds
Transparent pieces in the overview layer
UNM NYU

Figure 2: Overview layer (Cox et al. [11])

usage of this technique restricts the representation to a rectangular section. The rectangle size and position is relatively determined to the diagram with help of zoom tools, scroll bars and other graphical tools. The graphical user systems, which make use of this visualization paradigm are called *zoomable user interfaces*. However, this techniques have some technical drawbacks:

- As a consequence of limitation to a rectangular section the context information drops out. Hence, the human modeler is unable to observe what section is presented and how it interacts with the rest of the graphical system.

- The larger the shown section the larger the level of information detail is represented. The extension of the visual area reduces the size of graphical elements and the clarity. Hence, it is harder to recognize graphical elements.

- After only few navigation steps the human beholder is lost in the represented information space. This increases the difficulty to find the information of interest.

2.2 MULTIPLE VIEWS OF DIFFERENT ABSTRACTION LEVELS

Another possibility to increase the clarity of a graphical representation is to provide the human beholder with views of different magnification levels. A special case of this principle is the representation of a less detailed overview in a separated screen area or as alternative to the working view. The overview can be represented as permanent overview layer (see Cox et al. [11] and Figure 2) or temporary context layer (see Pook et al. [12] and Figure 3) and as visual combination the working layer.

The actually selected visual section is generally marked by a rectangular border and can be modified by moving the rectangle. To adapt the visual section the human beholder can select a rectangular area in the overview layer or in the working layer. As an alternative to miniaturization of the overview layer it is also possible to re-organize the represented information. *E. g.*, a tree-structure for hierarchically structured information is an adequate visual representation with high information density. However, a disadvantage of the combination of work and overview layer is the need for more space on the screen and for mental integration of both layers.

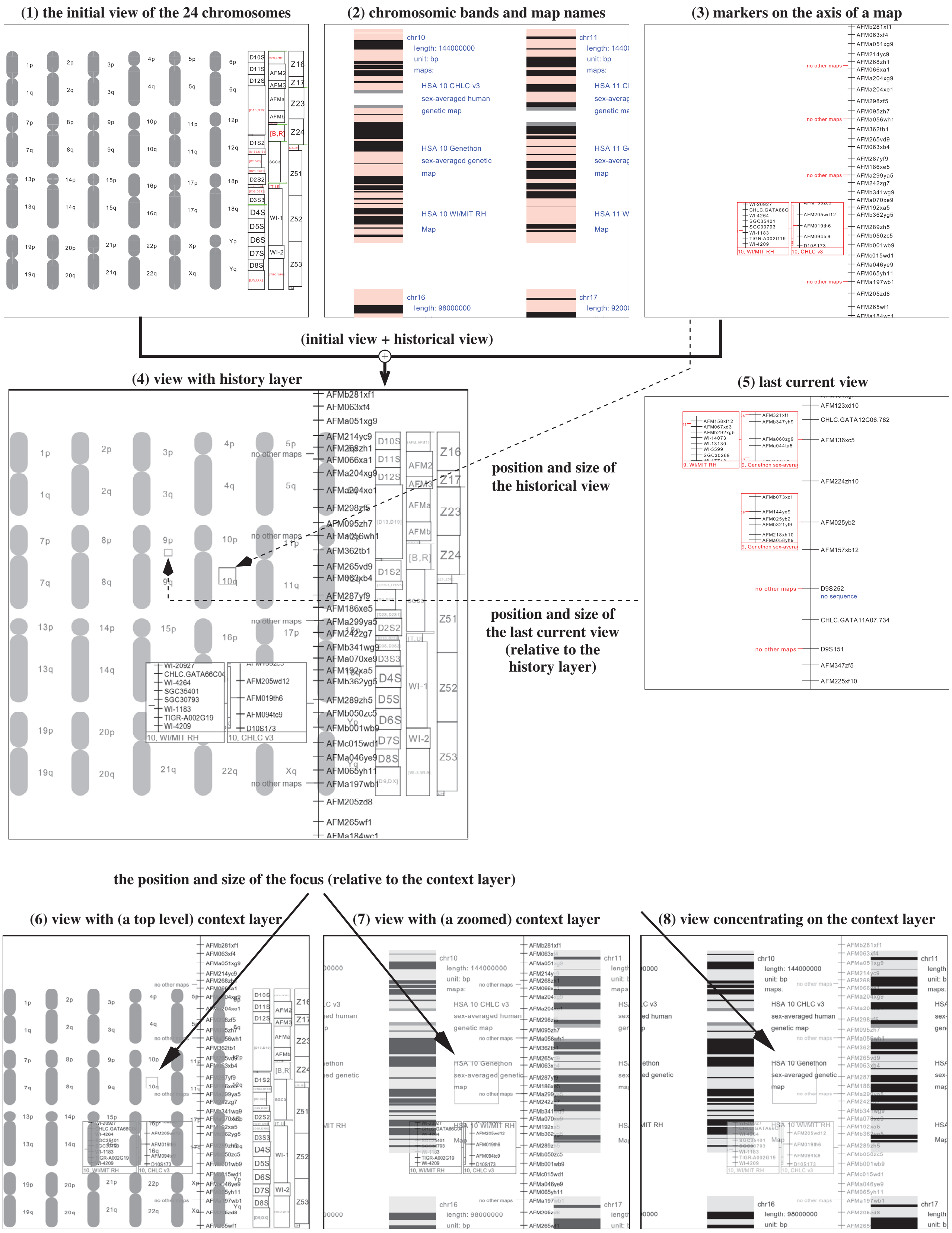

(1) the initial view of the 24 chromosomes
(2) chromosomic bands and map names
(3) markers on the axis of a map
(initial view + historical view)
(4) view with history layer
(5) last current view
position and size of the historical view
position and size of the last current view (relative to the history layer)
the position and size of the focus (relative to the context layer)
(6) view with (a top level) context layer
(7) view with (a zoomed) context layer
(8) view concentrating on the context layer

Figure 3: Context-, history and overview-Layer (Pook et al. [12])

2.3 ANIMATION OF NAVIGATION STEPS

Pook et al. [12] present one more navigation help for graphical systems that contributes to the understanding of the actual context. They propose to quickly repeat performed navigation steps (*e. g.*, zooming, minimization, moving of screen areas) on user demand. Additionally to the working layer a history layer represents the repeated tasks. By following this paradigm should ease the understanding for the modeler how the screen area selection was performed. This method can be simplified with an direct transition from the entire system view to the actually represented screen view. This variant is generally less confusing while working with graphical systems.

2.4 FOCUS-AND-CONTEXT REPRESENTATION

The focus-and-context representation allows to present currently interesting information in detail and to keep the relation to the context. The focus is generally presented with a high magnification factor to highlight all information details. For the visualization of the context a smaller magnification or a miniaturization is used to present as much context information as possible. The best-known realization of a focus-and-context representation is the *fish eye view*. Using a non-linear optical distortion mostly a circular area is magnified. The result is comparable with pictures resulting from a convex lens distortion or photography with an extreme wide-angle lens. Figure 4(a) shows a diagram representation with fish eye distortion. However, relevant literature emphasizes also disadvantages of fish eye views:

- Non-linear distortion representations aggravate the readability and comprehensibility of linear representations (*e. g.*, diagrams.)
- The computation effort of fish eye representations can delay the visualization and aggravate the traceability.

Sarkar and Brown [13] introduce an alternative focus-and-context method that overcomes the disadvantages of optical distortions. Their method rather preserves the character of the magnified focused objects using a graphical non-distortion magnification (see Figure 4(b)). The focus' position, size and level of detail can vary dependently on the visualization intent.

Definitions

Level of detail (LOD): *a priori* significance of an object, dependent on the selection of the visible screen area

Degree of interest (DOI): numerical value consisting of level of detail

Distance from Focus (DFF): object distance to the user focus of interest

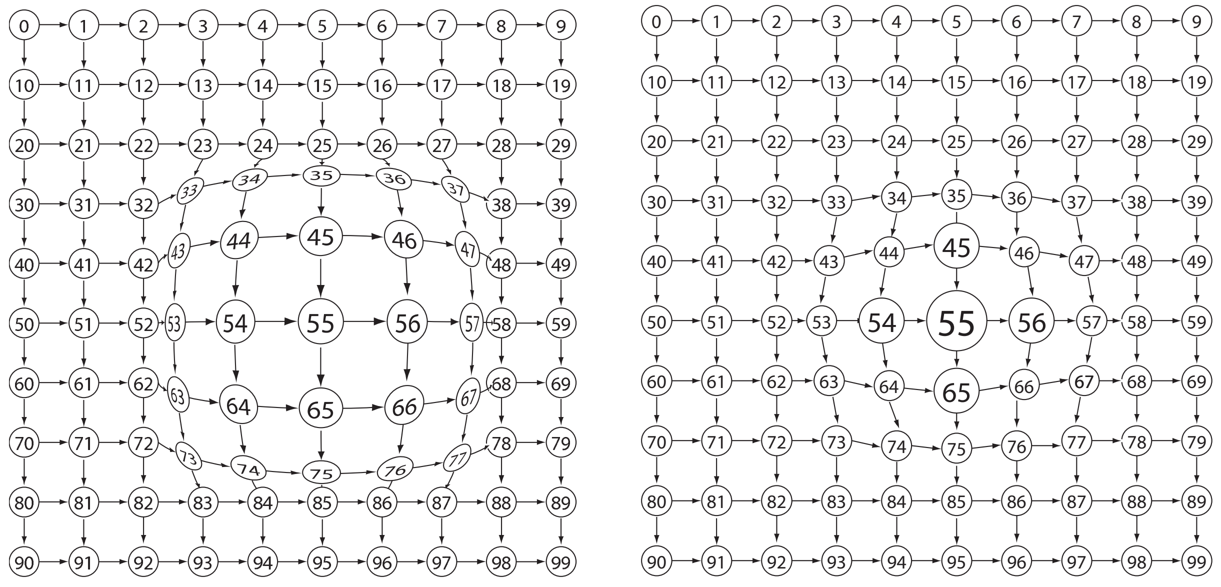
The size of a presented object depends on its original size and the distance to the focus of interest.

2.4.1 SEMANTIC ZOOMING

Semantic zooming is a method to meaningfully represent graphical models as *e. g.* graphical aircraft models. With this technique the level of detail (LOD) is adapted to the magnification factor. When the visible area contains too many graphical elements the number abstraction level will be increased to hide actually unimportant information. This technique leads to nearly constant visual complexity across multiple magnification levels. The semantic zooming makes use of the hierarchical structure of the represented information representation; more hierarchy levels will be presented with decreasing magnification level. However, the need for meaningful and aesthetic model representations requires often rearrangement and re-size of presented objects under consideration of the objects' context.

2.4.2 SEMANTIC FOCUS-AND-CONTEXT

Köth [14] provides a combination of focus-and-context representation and semantic zooming called semantic focus-and-context representation. Here, focus and context are represented at different levels of abstraction. The lower level of context abstraction is achieved by hiding details as realized with the semantic zooming technique rather than object miniaturization with semantic focus-and-context. An advantage of focus-and-context methods is the preservation of correct syntax and the distortion free representation of hierarchical diagrams. As a dynamic extension of semantic focus-and-context Prochnow and von Hanxleden [15] present a technique for efficient visualization of simulation behavior of embedded systems with Statecharts based on hiding sub-states of inactive hierarchical and uncollapsing of active states.



(a) optical magnification

(b) graphical magnification

Figure 4: Diagram representations with focus-and-context methods

3 AESTHETIC CRITERIA, COGNITIVE EXPERIMENTS AND LAYOUT CONVENTIONS

A central problem in visualization of graphical models are appropriate aesthetic criteria for a human understandable information representation. For common graph representation there exist lots of aesthetic criteria for model element placement and layout. However, assumptions on aesthetic criteria becomes even harder for aircraft models.

3.1 NORMAL FORMS

Normal forms for graphical structures describe graphical structures according to more or less accepted rules for structure and layout. They aim reduce the number of possible diagram element placements to one standardized representation, which leads to representation characteristics as, *e. g.*,

1. starting elements are placed on the upper left side,
2. connectors between graphical elements should be drawn from left to right,
3. connectors should be drawn without crossings, or
4. connectors should be drawn in clockwise direction.

Such criteria can help *e. g.*, to visually compare different graphical models. Hence, normal forms ideally combine a number of aesthetic criteria to gain on one side

1. a unique representation of graphical model elements and graphical models and on the other
2. a good readable and understandable graphical model representation.

Beside the advantages in cognition, normative aesthetic criteria can save computation time and memory since the linear representation can be calculated faster and are visualized in a more compact way than arbitrary two-dimensional model representations.

3.2 COGNITIVE EXPERIMENTS ON VISUAL LANGUAGES

The phrase "one picture is worth ten thousand words" comprises a shortened classification and evaluation of alternatives with the same meaning. The human mind can faster perceive and make sense of graphical information representations than textual or spoken representations. This is also true for specification and communication *e. g.*, in computer science. Here, classical visual examples of graphical languages are flow diagrams, Petri nets, or class diagrams in *UML* [16]. However, often some graphical specifications need more specific textual descriptions using *e. g.*, programming languages as *C* and *Java*. Generally a diagram represents a (part of a) visual description based on an underlying visual language; analogously textual descriptions base on textual languages. Often graphical languages are often seen as advantageous for

- complex structures for faster cognition and
- simple descriptions with only few graphical elements, which does not need extensive explanations.

Benveniste et al. [17] highlight for languages in computer science: *"Today, we see with some surprise that visual notations for . . . languages have found their way to successful industrial use with the support of commercial vendors. This probably reveals that building a visual formalism on the top of a mathematically sound model gives actual strength to these formalisms and makes them attractive to users."* However, visual languages have some disadvantages. As an example the non-linearity of visual diagrams aggravates their creation and especially their modification. To get an overview of the graphical system, the developer often takes resort to paper and pencil. Moreover, a complex graphical model cannot be easily printed; the print-out needs often the space of a whole desk, the wall, or the roof. With textual languages the print can be reduced to few sheets of papers. With a textual description as *e. g.*, programming languages the described elements can be easily added, moved or deleted without disrupting the document readability. With a visual diagram adding and moving of describing graphical elements is much more effort and raises the need for re-arranging some graphical elements to improve the readability, as programmer states *"I quite often spend an hour or two just moving boxes and wires around, with no change in functionality, to make it that much more comprehensible when I come back to it (Petre [18])."*

Cognitive experiments on graphical formalisms highlight some weaknesses of their readability (compare Petre and Green [19], Petre [18], Moher et al. [20]). In contrast to an artist picture the intention of a graphical modeling language is to precisely and efficiently provide the reader of a graphical system with a concrete meaning of the modeled system. A central question in working with graphically modeled systems is whether the system does provide same meaning to different model readers. Related to that is also important to make clear how much effort is needed in order to extract this meaning of the represented graphical model. Generally, written text constitutes graphics with a very limited vocabulary. The experienced reader intuitively performs an abstraction and does not recognize letters as single graphical elements with their specific individual graphical properties. Textual languages were developed since many centuries as medium for description of technical information. In contrast visual languages are comparable underdeveloped. An interesting conclusion drawn in cognitive experiments is that the comprehension of a graphical diagram is not only dependent on the graphical elements, but also from a secondary notation, *i. e.*, from the layout and further typographic annotations. Experienced readers as well as programmers use typographic and other types of annotations for efficient reading and understanding of textual programming languages. The same characteristics can help to understand visual languages. Petre and Green [19] summarize the behavior of beginners and experienced developers of graphical models as follows:

- When the both equivalent representation—graphical and textual—were provided in parallel then experienced users preferred to read the text to understand the graphics.
- It often hard for beginners to understand the essence of graphics. (In contrast to the general opinion about graphics is that their meaning is obvious.) Existent, but irrelevant connection lines often confuse beginners.
- Experts can make efficient use of their fingers to understand graphical structures.

Moher et al. [20] have analyzed the readability of different Petri net variants. They concluded that:

- the graphical variants were generally not better readable than the textual variants and
- the readability of diagrams was strongly dependent on the layout.

However, programmers stated a preference for graphical diagrams. This can be explained by the following properties of graphical languages:

- They are more rich, *i. e.*, more information are represented with less unnecessary space.

- They provide a "Gestalt" effect and efficiently visualize structures.
- They provide a higher abstraction level; hence, they focus more the expressed problem.
- They are more accessible, easier and faster to understand, and easier to remember.
- They are less formal and "not-symbolic."
- They are fun to read.

In general it seems that the illusion of a better information accessibility is more important than real improvements in information representation. The positive adapted character of visual languages can raise the reading motivation.

Petre [18] states that graphical model designs created by beginners are harder to read than those created by modeling experts. The reason is that beginners don't care so much about layout. In contrast experts try to improve the model readability by re-arranging the graphical elements during the whole modeling process. Petre [18] also states that *"It is time to recognize the impact of 'bad graphics'—of haphazard use of perceptual cues and secondary notations—mis-cueing, misleading, misreadig, and misunderstanding."* This is especially true for the design of aircraft systems, where we can find more and more applications for graphical system models. Furthermore it is stated *" It appears that graphical notations can have a greater capacity to 'go wrong' than textual notations."*

An impressive number of style guides have been developed for textual languages, *e. g.*, for writing programming languages as *C* or *Java*. Besides that, only a small number for graphical languages in computer science such as the MathWorks Automotive Advisory Board (MAAB) [21] have been developed. However, these are comparatively less precise and give only few rules for model layout. Moreover, tools for textual languages that support analyzing and formatting, and that check and/or force the use of certain modeling style guides are also available. Purchase et al. [22] performed an experiment to express the relation between the conformity to certain aesthetic criteria and the readability of *UML class diagrams*. They found that in general the diagram layout should follow the modeled meaning and semantics.

4 SUMMARY

Graphical diagrams for specification of aircraft system models can efficiently support the aircraft system design and help to make the development process faster and less cost-intensive. The problem is that today's modeling tools don't provide efficient techniques to create and maintain graphical models. This paper provides an overview of related problems and refers to analogue problems that are already focused in computer science, *e. g.*, in *UML* diagrams. We reviewed the relevant literature and showed how problems in graphical aircraft system modeling can be tackled and how findings from domains can be adapted and used in this field. We argued that the aircraft system models could profit from he use of introduced techniques that can improve their readability, understandability, and maintainability that would finally result in improvements of the entire aircraft system modeling process.

REFERENCES

[1] A. Schneegans and O. Kranz. Flying Objects — An object oriented toolbox for multidisciplinary design and evaluation of aircraft. In *21st International Council of the Aeronautical Systems (ICAS) Congress*, Melbourne, Australia, September 1998.

[2] J. A. Reed and A. A. Afjeh. Computational Simulation of Gas Turbines: Part 1 —- Foundations of Component-Based Models. *Transactions of ASME*, 122(3):366, 2000.

[3] M. Drela and H. Youngren. *AVL 3.26 User Primer*. MIT Aero and Astro and Aerocraft Inc.

[4] Tomas Melin. *User's Guide Tonado*. Royal Institute of Technology (KTH), Stockholm, Sweden, December 2000.

[5] A. Mccullers. Aircraft Configuration Optimization Including Optimized Flight Profiles. In J. Sobieski, editor, *Proceedings of the Symposium on Recent Experiences in Multidisciplinary Analysis and Optimization*. NASA CP-2327, 1984.

[6] L. Morino, G. Bernardini, and F. Mastroddi. Multi-disciplinary Optimization for the Conceptual Design of Innovative Aircraft Configura-

tions. In *Computer Modeling in Engineering and Sciences (CMES)*, page 1.

[7] Dassault Systems. 3D CAD CAM product design software – CATIA. WWW, last visited: July 25, 2011. http://www.3ds.com/products/catia.

[8] PACE. Pacelab Suite. WWW, last visited: July 25, 2011. http://www.pace.de/products/preliminary-design/pacelab-suite.html.

[9] Ying K. Leung and Mark D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[10] Edward R. Tufte. *Visual Explanations*. Graphics Press, Cheshire, Connecticut, 1997.

[11] D. Cox, Jasdeep S. Chugh, Carl Gutwin, and Saul Greenberg. The usability of transparent overview layers. In *Companion Proceedings of the CHI '98 Conference on Human Factors in Computing Systems*, pages 301–302, Department of Computer Science, University of Calgary, 1997. ACM Press.

[12] Stuart Pook, Eric Lecolinet, Guy Vayssaix, and Emmanuel Barillot. *Context and Interaction in Zoomable User Interfaces*. ACM Press, 2000.

[13] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proceedings of the ACM SIGCHI 1992 Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

[14] Oliver Köth. Semantisches Zoomen in Diagrammeditoren am Beispiel von UML. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2001.

[15] Steffen Prochnow and Reinhard von Hanxleden. Comfortable modeling of complex reactive systems. In *Proceedings of Design, Automation and Test in Europe Conference (DATE'06)*, Munich, Germany, March 2006.

[16] Object Management Group. Unified Modeling Lanugage—UML resource page, 2005. http://www.uml.org.

[17] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The Synchronous Languages Twelve Years Later. In *Proceedings of the IEEE, Special Issue on Embedded Systems*, volume 91, pages 64–83, January 2003.

[18] Marian Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6): 33–44, June 1995.

[19] Marian Petre and Thomas R. G. Green. Learning to read graphics: Some evidence that "seeing" an information display is an acquired skill. *Journal of Visual Languages and Computing*, 4 (1):55–70, 1993.

[20] Thomas G. Moher, David C. Mak, Brad Blumenthal, and Laura M. Leventhal. Comparing the comprehensibility of textual and graphical programs: The case of petri nets. In *Empirical Studies of Programmers: Fifth Workshop*, pages 137–161. Ablex Publishing, 1993.

[21] MathWorks Automotive Advisory Board (MAAB). *Controller Style Guidelines for Production Intent Using MATLAB, Simulink and Stateflow*, April 2001. http://www.mathworks.com/industries/auto/maab.html.

[22] Helen C. Purchase, David Carrington, and Jo-Anne Allder. Graph layout aesthetics in UML diagrams: User preferences. *Journal of Graph Algorithms and Applications*, 6(3), 2002.