Design languages
for multi-disciplinary architectural synthesis
and analysis of complex systems in
the context of an aircraft cabin

Priv.-Doz. Dr.-Ing. Stephan Rudolph
Dipl.-Ing. Peter Arnold
Dipl.-Ing. Marc Eheim
Dipl.-Ing. Stefan Hess
Dipl.-Ing. Martin Motzer
Dipl.-Ing. Marius Riestenpatt genannt Richter
Dipl.-Ing. Jens Schmidt
Dipl.-Ing. Roland Weil

Similarity Mechanics Group
Institute for Statics and Dynamics of Aerospace Structures (ISD)
University of Stuttgart, Germany
Why
using graph-based design languages?

How
using graph-based design languages
aircraft cabin (3D geometry and 3D network routing)

More
digital factory, fault-tree analysis, …
fiber-reinforced structures

Future
cooperation possibilities (EU-and national projects)
workshop in Stuttgart March 2015
research question

“universal”
design theory

synthesis

...analysis

representation evaluation

...theory still unknown, but....

...arbitrary choice for aircraft design?

evaluation needs:
- objective
- holistic

synthesis needs:
- multiple domain
- multiple designs

aircraft design just special case
research assumption

“universal”
design theory

synthesis

analysis

L-systems

shape grammars

graph-based design languages

"generic"
design language
approach

representation

evaluation

similarity mechanics

synthesis advantages:
- domain independent
- design independent

evaluation advantages:
- at runtime
- error-free
design languages

- **vocabulary components**
- **rule set**
- **building rules**

**design language**
(vocabulary + rules = grammar)

**syntax**

**semantics**

**pragmatics**

- technically "meaningful"
- context "adequate"

- combinatorial "possible"

- interpretation → formalisation
- instantiation → abstraction
system of systems

model a ➔ model b ➔ model d ➔ model e ➔ model c

domain modeling (yellow) vs. domain coupling (red)
"The" problem

market study feasibility / requirements list study conceptual design detailed design functional verification assembly verification prototypes production planning digital factory production planning digital factory production maintenance repair recycling

"birth" go/no-go go/no-go go/no-go

go/no-go

Goal: Support of all engineering-activities along the product life-cycle by appropriate methods and tools

product-life-cycle

design freeze product freeze
Goal: Support of all engineering-activities along the product life-cycle by appropriate methods and tools

Graph-based design languages in UML
evolution of **programming languages**

- productivity, quality
- und security by
  - code expansion
  - code re-use
  - code generation

- with capability of
  - abstraction
  - hierarchization
  - modularization
  - visualization of information

- design of complex
- SW/HW-systems

---

**abstraction**

- high

---

**modeling languages**

- public class Flugzeug {
  - private String name;
}

---

**object-oriented languages**

- int main(int argc, char **argv)
  - { fprintf(stdout, „Hello“); }

---

**procedurale languages**

- MOVF id1, R1
- ADDF R2, R1

---

**assembler languages**

- machine code
  - 00101011 01010111
  - 01010101 10101011

---

adapted from Gruhn et al. (2006)
The evolution of design languages involves productivity, quality, and security improvements over time. Design languages support abstraction, hierarchization, modularization, and visualization of information. With capability of:
- code expansion
- code re-use
- code generation

Design languages are adapted from Gruhn et al. (2006) and include:

**Assembler languages**
- MOVF id1, R1
- ADDF R2, R1

**Machine code**
- 00101011 01010111
- 01010101 10101011

**Procedurale languages**
- Int main(int argc, char **argv)
- { fprintf(stdout, „Hello“); }

**Object-oriented languages**
- Public class Flugzeug {
  private String name; }

The design of complex SW/HW-systems with capabitlity of engineering extensions:
- rule execution
- constraint processing

Design languages with international, vendor-independent format and development tools:
- machine code
- assembler languages
- procedurale languages
- object-oriented languages

With productivity, quality, and security by:
- code expansion
- code re-use
- code generation

The time evolution of design languages is shown from 1950 to 2020.
class decomposition
UML class diagramm

- holistic aircraft description
  - topology
  - parametrics
  - geometry
  - physics (constraints)
graph-based design language (in UML)

- design process (generic)
  
  - vocabulary
  - rules

- program

- model

- simulation

- evaluation

- necessary initial effort

- (iterative design loop)

- vocabulary (as UML classes)
  
  - Aircraft
  - Fuselage

- rules (as UML model-transformations)

- production system (as UML activity diagram)
  
  consists of a sequence of design rules
design language

- vocabulary
- rules

production system

typically model-to-model transformations

design language
(definition and programming)

design language
(compilation and execution)

design graph

typically model-to-text transformations

CAD
- CATIA V5
- SOLID WORKS
- VRML
- ...

MBS
- ADAMS
- ...

FEM
- ANSYS
- NASTRAN
- MSC Laminate
- Modeler, …
- FLUENT
- STAR-CD
- STAR-CCM+ …

CFD
- MATLAB/SIMULINK
- ...
- ESATAN-TMS
- ...

CAS
- MAPLE
- MATHEMATICA
- …
design loops

- design language (definition and programming)
- design language (compilation and execution)

production system

closing the "design loop" (feed-back)

design compiler 43

design graph

- CAD
  - CATIA V5
  - SOLID WORKS
  - VRML
- MBS
- FEM
- CFD
  - ANSYS
  - FLUENT
  - STAR-CD
  - STAR-CCM+
- MATLAB/SIMULINK
- ESATAN-TMS
- MAPLE
- MATHEMATICA

vocabulary

rules

manual or computer-assisted
Why
using graph-based design languages?

How
using graph-based design languages
aircraft cabin (3D geometry and 3D network routing)

More
digital factory, fault-tree analysis, …
fiber-reinforced structures

Future
cooperation possibilities (EU-and national projects)
workshop in Stuttgart March 2015
aircraft cabin (geometry + systems + ...)
system of systems

model a

model b

model c

model d

model e

http://www.rolls-royce.com

domain modeling

domain coupling
rule definition (M2M-transformation)

domain expert(s) → **rule:** add instances to an existing instance aircraft

![Diagram showing a rule for adding instances to an existing instance of an aircraft.]

domain experts → **rule:** specialization of a generic cable to a specific type (cast, slots added)

![Diagram showing a rule for specializing a generic cable to a specific type (cast, slots added).]
production system (activity diagram)

Hierarchical nesting of production systems in aircraft cabin design language
cabin layout

layout design language: calculation of layout according to requirements

design graph: data structure of the cabin layout

geometry design language: generation of aircraft cabin geometry
from cabin layout to network

creation of network interfaces depending on cabin layout i.e. seats and monuments
integration on class layer

cabin.layout class diagramm

lighting class diagramm (partially)

electrics class diagramm

routing profile
network generation and full 3-D routing

initial CAD geometry

define boundary conditions

generate routing space

meshed for later collision checks

variable obstacles

parameterized equipment boxes

build grid

collision checks

search algorithm

modified A* algorithm

post processing

collision free CAD geometry with exact cable length
design trades (2 network variants)

design trade: 8 versus 4 SPDBs
Why
using graph-based design languages?

How
using graph-based design languages
aircraft cabin (3D geometry and 3D network routing)

More
digital factory, fault-tree analysis, …
fiber-reinforced structures

Future
cooperation possibilities (EU-and national projects)
workshop in Stuttgart March 2015
aircraft design language

product line
aircraft family

A318
A319
A320
A321

detailed view
wing (topology / parameters)

(detailed view)
aircraft panel

(external)
(interior)

aircraft panel (CATIA V5)

- fuselage section
- 4 panels
- lateral and longitudinal structural connections
- riveting
- parameters:
  - nFrame=5
  - nStringer=6

-generation time approx. 15 mins
  (about 2/3 spent for CATIA V5 model)
digital factory

integrated design and generation of the digital factory along the digital design phase using graph-based design languages

simulation and analysis
- choice of tools
- choice of layout
- path generation
- and much more…
manufacturing sequence

\[
\begin{align*}
N(\text{Clip1}, \text{Niet1}) &= 0 \\
N(\text{Clip1}, \text{Niet2}) &= 0 \\
N(\text{ClipN}, \text{NietM}) &= 0 \\
N(\text{Frame1}, \text{Clip1}) &= 0 \\
N(\text{Frame1}, \text{Clip4}) &= 0 \\
N(\text{FrameM}, \text{ClipN}) &= 0 \\
N(\text{Skin}, \text{Clip1}) &= 0 \\
N(\text{Skin}, \text{Clip2}) &= 0 \\
N(\text{Skin}, \text{ClipN}) &= 0 \\
N(\text{Stringer1}, \text{Clip1}) &= 0 \\
N(\text{Stringer1}, \text{Clip2}) &= 0 \\
N(\text{Stringer1}, \text{ClipN}) &= 0 \\
N(\text{StringerM}, \text{ClipN}) &= 0
\end{align*}
\]
digital factory

- integrated design and generation of the digital factory along the digital design phase using graph-based design languages
- simulation and analysis
  - choice of tools
  - choice of layout
  - path generation
  - and much more…
Why
using graph-based design languages?

How
using graph-based design languages
aircraft cabin (3D geometry and 3D network routing)

More
digital factory, fault-tree analysis, …
fiber-reinforced structures

Future
cooperation possibilities (EU-and national projects)
workshop in Stuttgart March 2015
propulsion system (design and FTA)

- design language shown encodes „systematic german design methodology“ (Pahl and Beitz, 1972)

- Pahl and Beitz (1972) design means consistent mapping: requirements → abstract product functions → solution principles → embodiments
cooperation

How

using graph-based design languages
aircraft cabin (3D geometry and 3D network routing)

More

digital factory, fault-tree analysis, …
fiber-reinforced structures

Future

cooporation possibilities (EU-and national projects)
knowledge-based methods workshop
at Stuttgart University in March 2015
thanks / questions

…. ask / approach me for cooperation ideas…. Horizon 2020 Factory of the Future (FoF program)…

… you provide the application know-how
… we do the processing

…and the PhD candidates of the „Similarity Mechanics Group“
Dipl.-Ing. Peter Arnold
Dipl.-Ing. Marc Eheim
Dipl.-Ing. Jürgen Freund
Dipl.-Ing. Stefan Hess
Dipl.-Ing. Martin Motzer
Dipl.-Ing. Marius Riestenpatt genannt Richter
Dipl.-Ing. Jens Schmidt
Dipl.-Ing. Roland Weil