Digital continuity, consistency and interoperability along the product life-cycle using graph-based design languages

Dr. Stephan Rudolph, University of Stuttgart, Germany

Bio Stephan Rudolph

Stephan Rudolph

education

- Dipl.-Ing. / Dr.-Ing. / Priv.-Doz. (venia legendi) in Aerospace Engineering from University of Stuttgart, Germany

profession

- aerospace engineering faculty member, email rudolph@ifb.uni-stuttgart.de
- lectures on digital product design, art and science of systems architecting, design languages and algorithms, digital engineering, artificial intelligence
- adjunct associate professor, Swinburne University of Technology, Melbourne, Australia
- principal investigator in multiple German and European research projects
- design theory and similarity mechanics group head, academic supervisor of currently 21 PhD students
- more than 160 publications

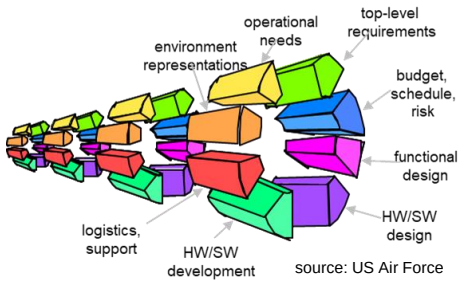
- co-founder of Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH, a small German high-tech company (see www.iils.de)

- expert on graph-based design languages

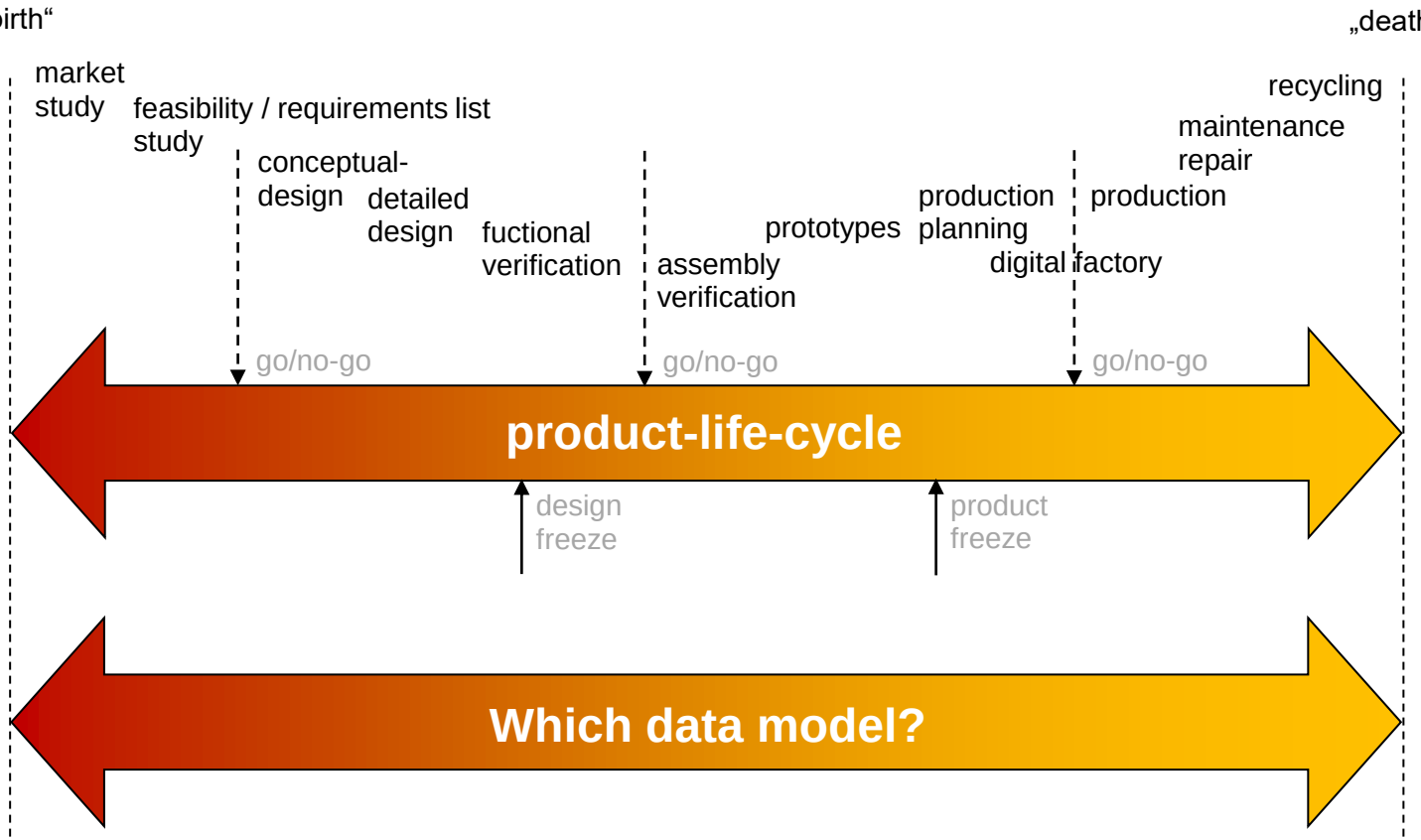
Which Data Model solves the Problem of Digital Continuity, Consistency and Interoperability?

todays deficiencies

- no/limited data consistency
- no/limited process continuity
- no/limited tool interoperability
- no/limited scope, scalability
- knowledge islands, data lakes….

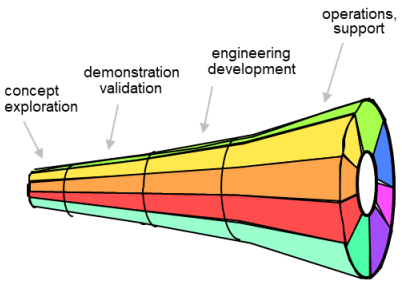


source: US Air Force



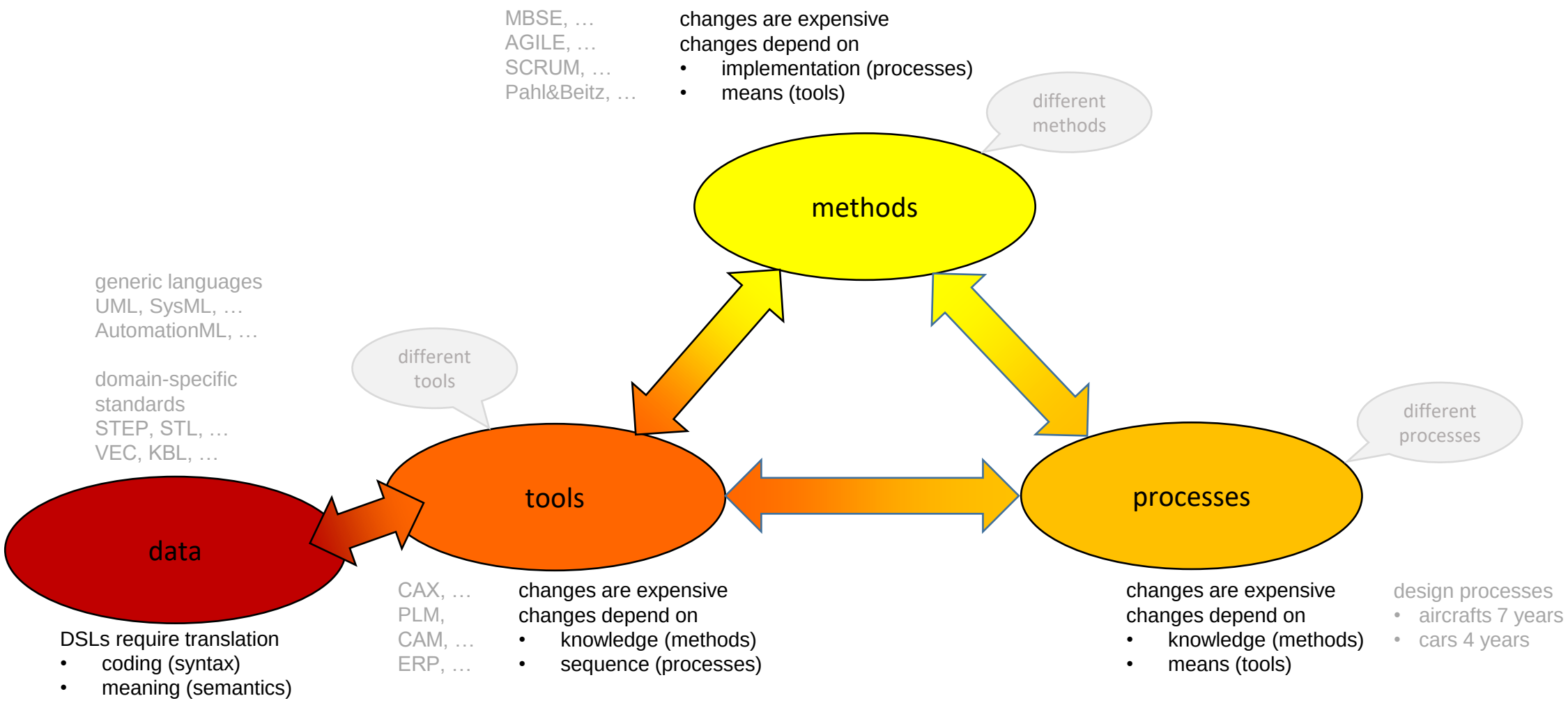
desired capabilities

- digital data consistency
- digital process continuity
- digital tool interoperability
- full scope, full scalability
- knowledge re-use, ….



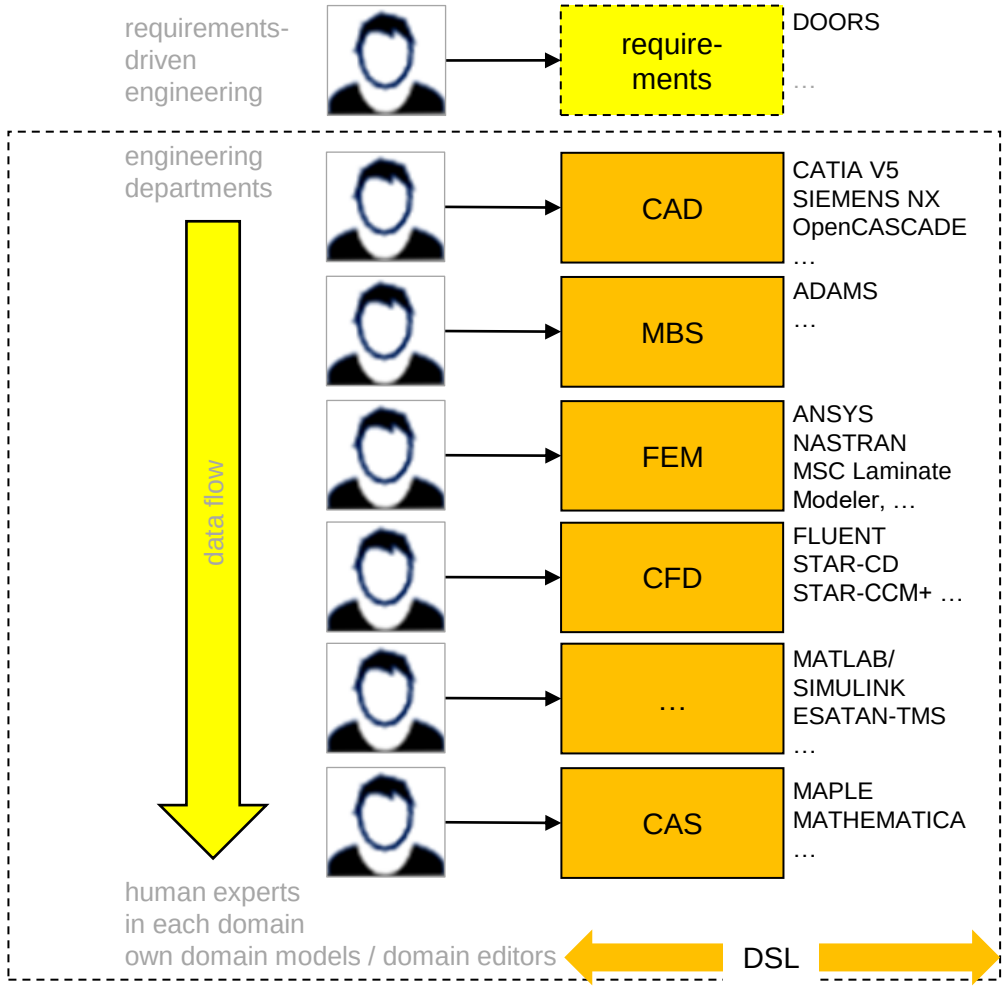
source: US Air Force

Engineering Problem Analysis (1 of 2)



Engineering Problem Analysis (2 of 2)

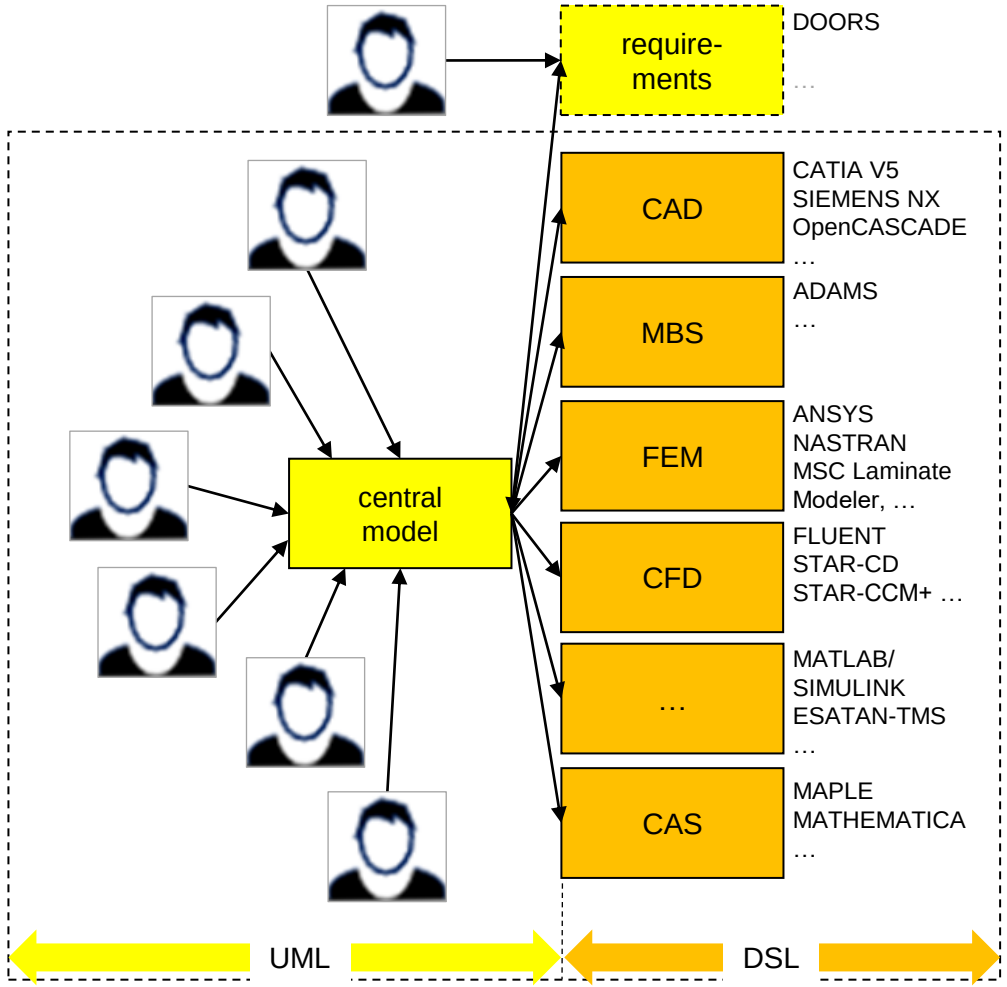
disadvantages/deficiencies (from computer science perspective)
- existing multitude of domain editors → no single source of truth
- data flow in iterative design loops → revision management
- design loops/iterations → manual rework
- models in DSLs → vendor lock-in



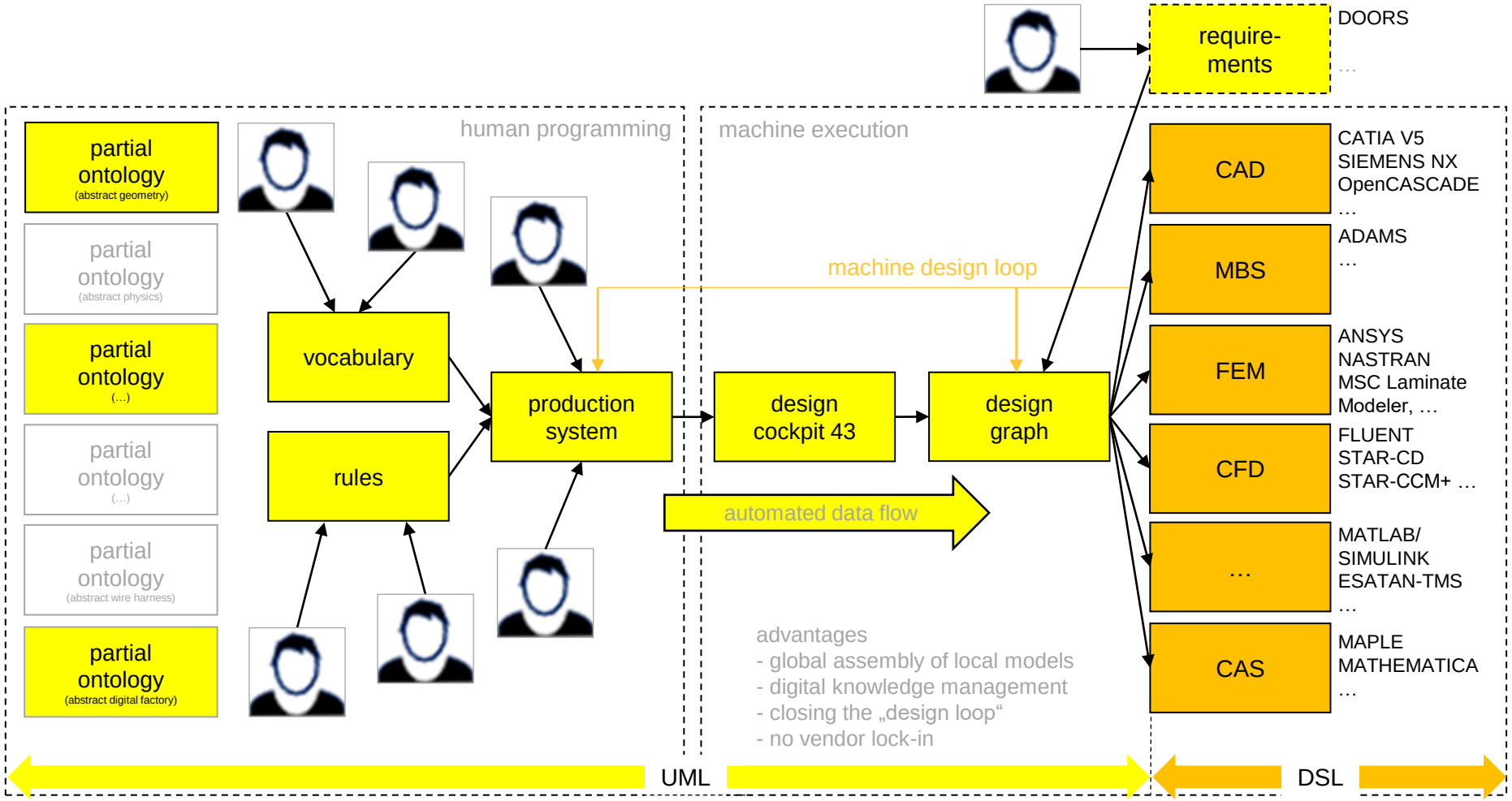
Engineering Solution (1 of 2)

advantages
- engineering data consistency through central model mapping
- use of generic object-oriented modeling language (here: UML)

disadvantages
- engineering design information flow needs to be reorganized
- engineers must be trained in new way of thinking / working

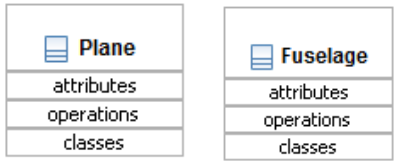


Engineering Solution (2 of 2)

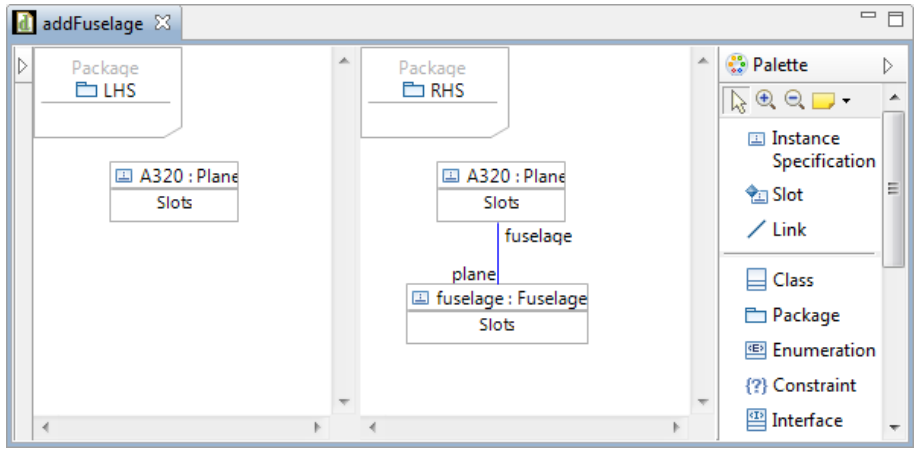


Graph-Based Design Languages (in UML)

- vocabulary (UML classes)

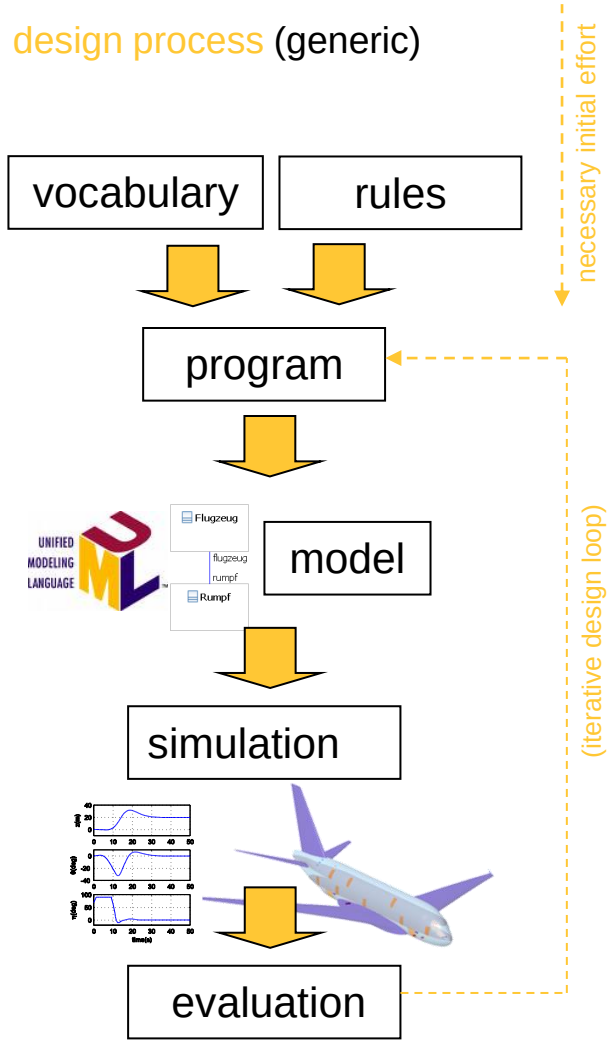


- rules (UML model-transformations)



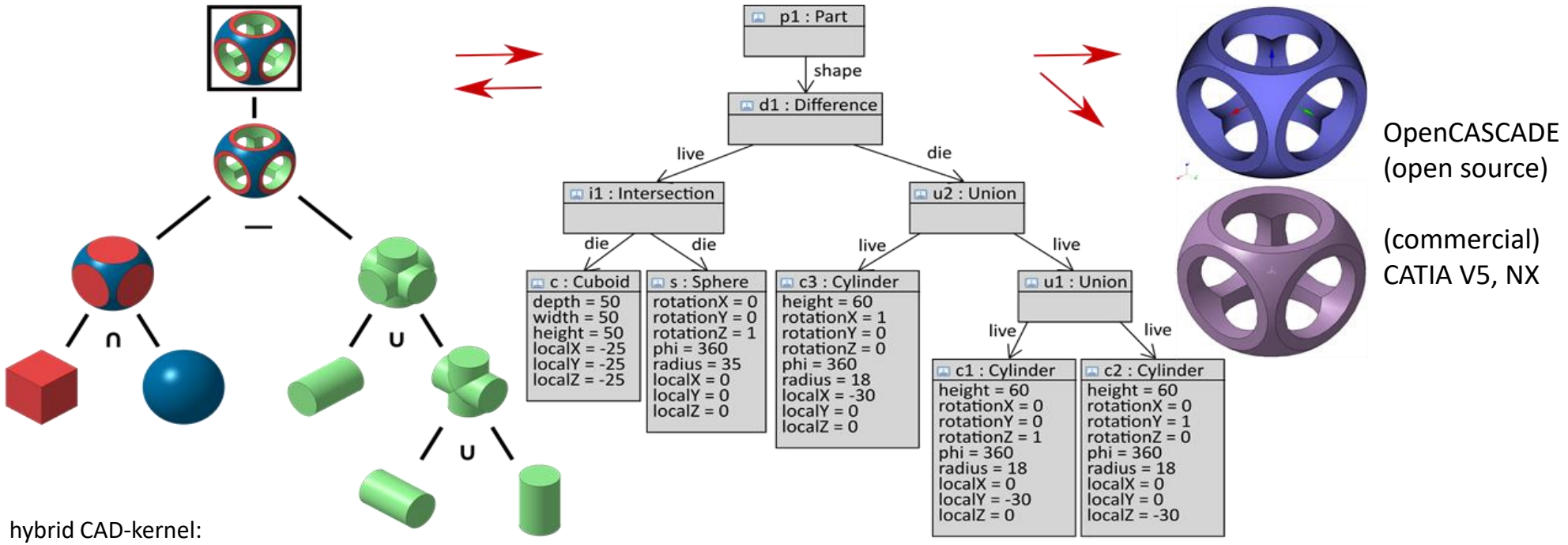
- production system (UML activity diagram) consists of a sequence of design rules

- design process (generic)



Graph-Based Design Languages (Abstract Geometry, Part 1)

CSG (constructed solid geometry)



hybrid CAD-kernel:
- CSG (complete)
- BREP (work in progress)

Graph-Based Design Languages (Abstract Geometry, Part 2)

BREP (boundary representation)

points

point number	X	Y	Z
1	4	0	0
2	-1	3	0
3	-1	-3	0
4	4	0	4
5	-1	3	4
6	-1	-3	4

lines

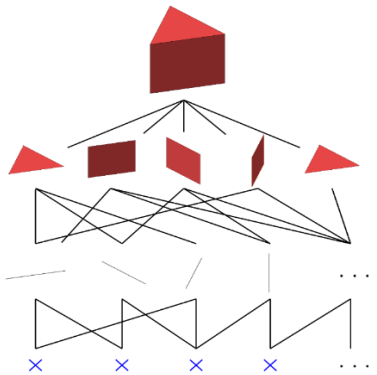
edge number	vertex 1	vertex 2
1	1	2
2	2	3
3	3	1
4	4	5
5	5	6
6	6	4
7	1	4
8	2	5
9	3	6

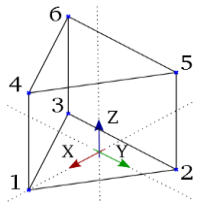
surfaces

surface number	edge sequence
1	1 2 3
2	4 5 6
3	1 8 4 7
4	2 8 5 9
5	3 7 6 9

volumes

volume number	surfaces
1	1 2 3 4 5

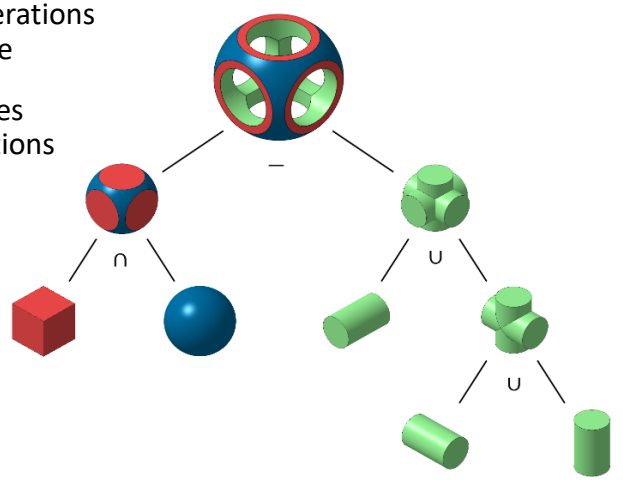


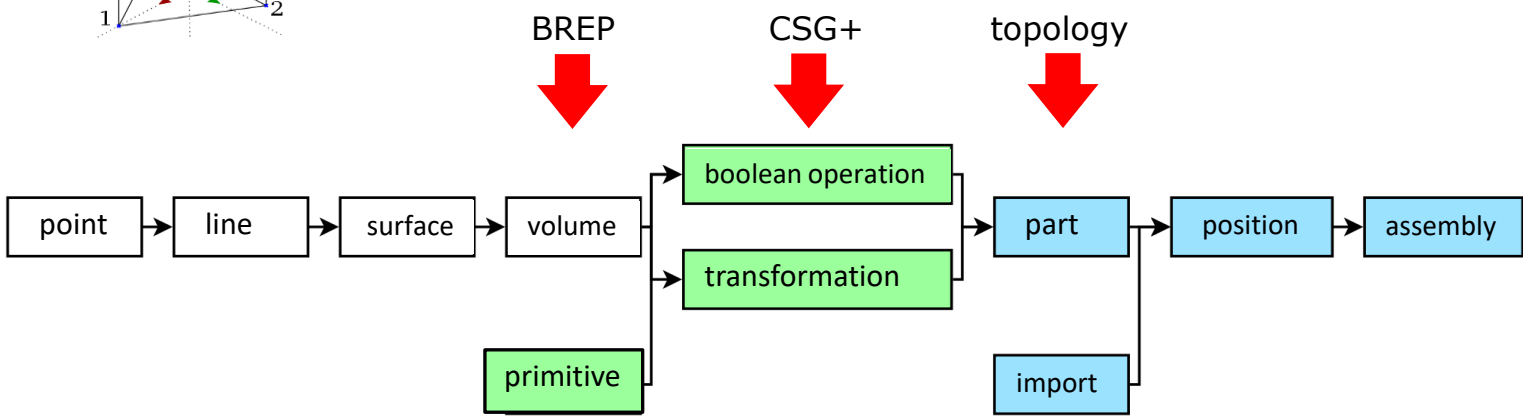


hybrid CAD-kernel:
- CSG (complete)
- BREP (work in progress)

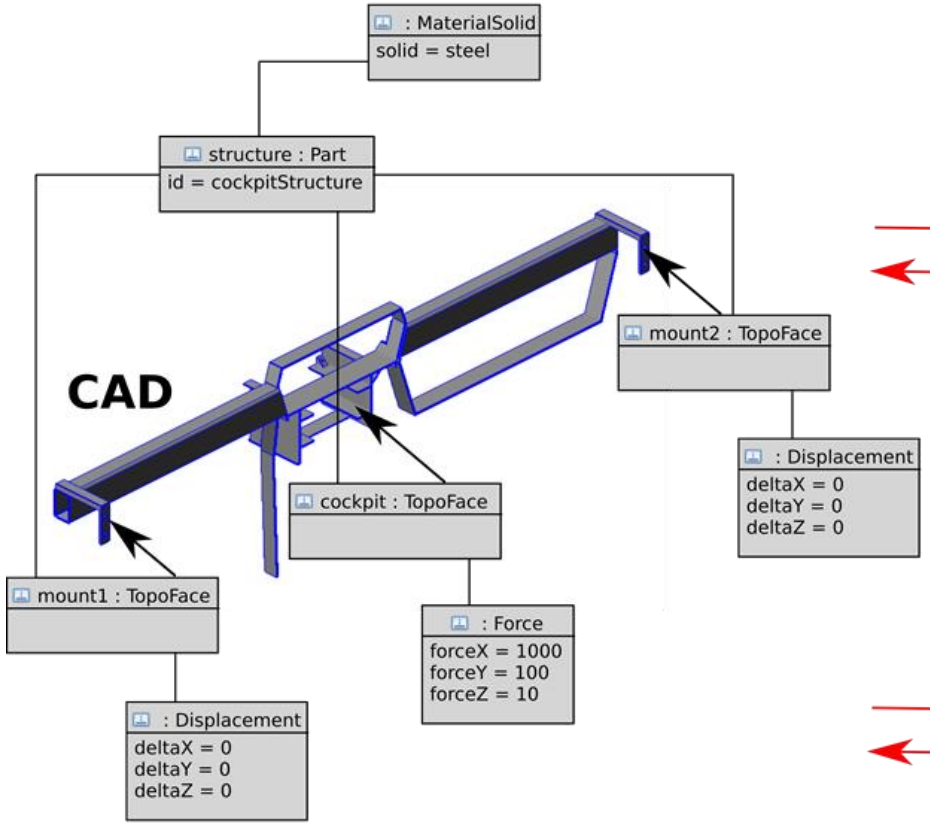
CSG (constructed solid geometry)

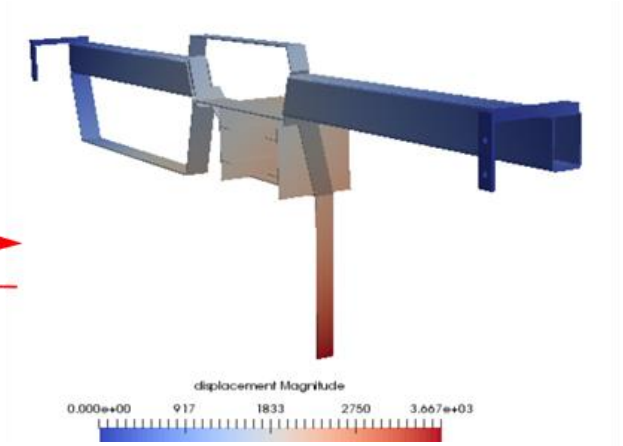
- primitives ∘ operations
  → binary tree
- leaves: primitives
  vertices: operations
  root: object



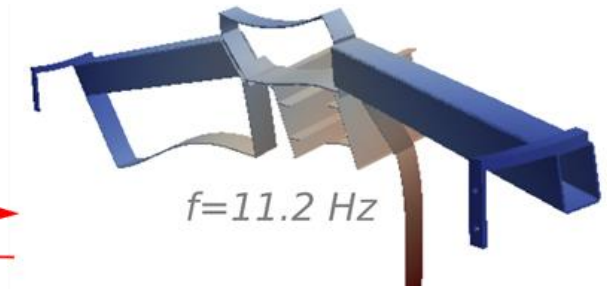


Graph-Based Design Languages (Abstract Physics)





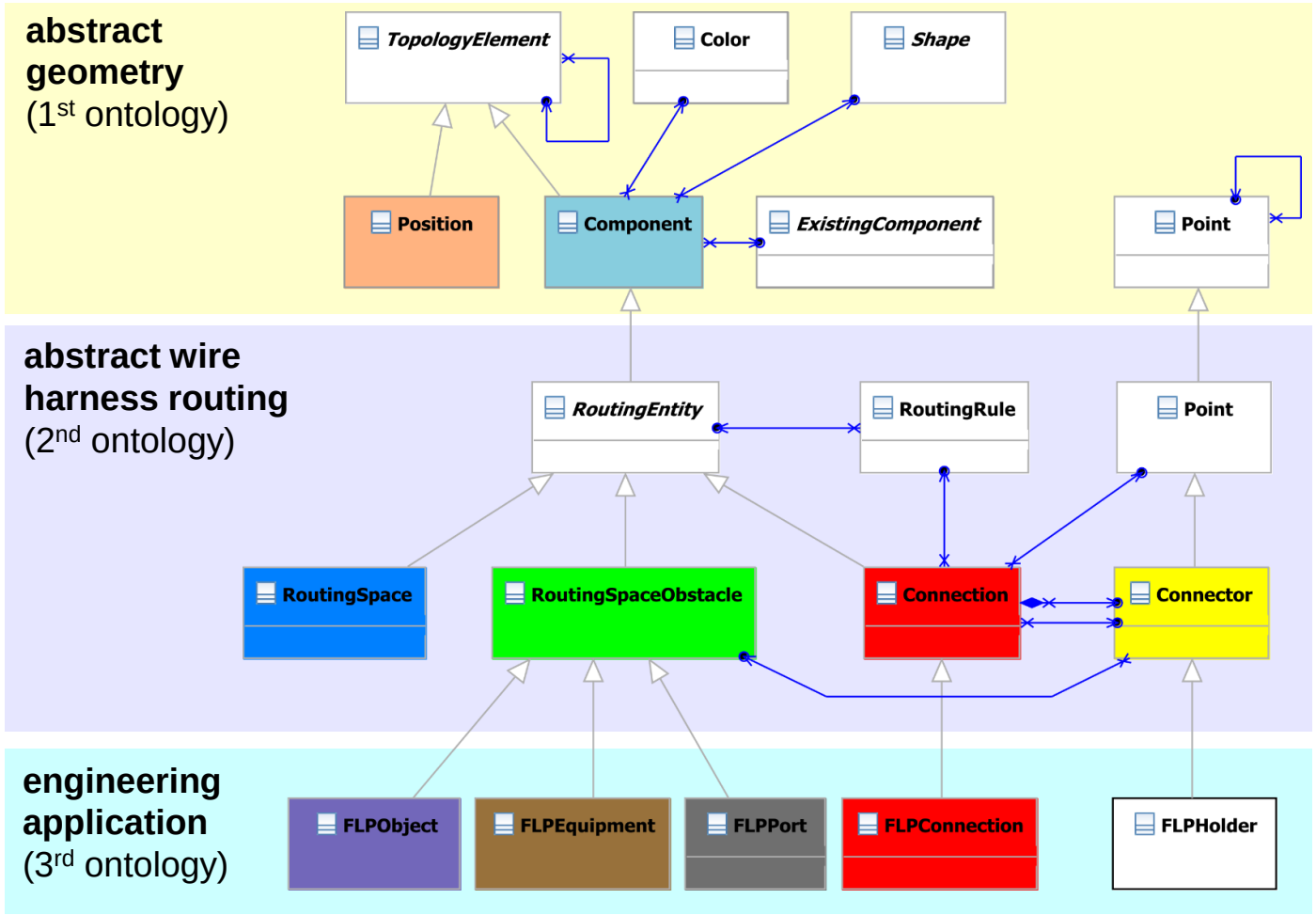
displacement / stress

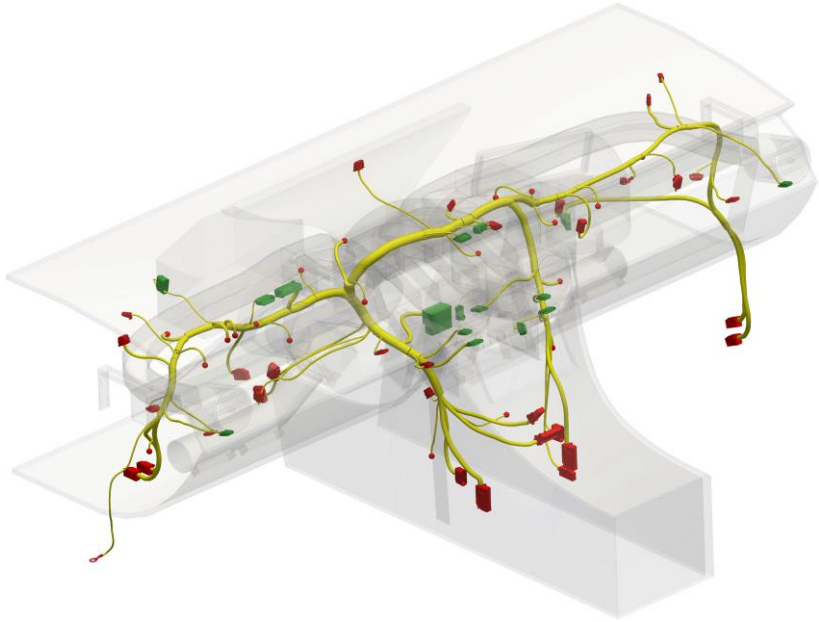


modal analysis

FEM solvers
- ABAQUS (commercial)
- ELMER (open-source)

Graph-Based Design Languages (Abtract Wire Harness and Ontology Mapping)





design languages properties
- graph-based representation
- digital consistency
- digital continuity
- digital interoperability

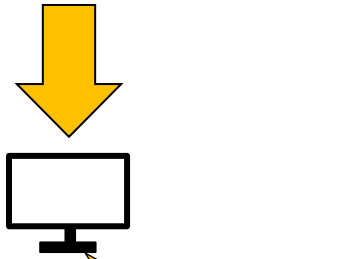
video automotive cockpit: CAD/FEM/3D wire harness routing

ontology mapping
- via UML modeling mechanisms
- via UML model transformations (M2M, M2T)

Graph-Based Design Languages (in UML)

human execution

human reaction time 10^{-1} [s]
versus
processor cycle time 10^{-9} [s]



machine execution

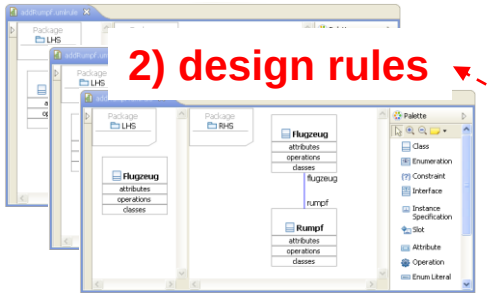
3) products

competence

- pragmatics
- capability of model synthesis and analysis with arbitrary boundary conditions

compilation process

creative act emergence



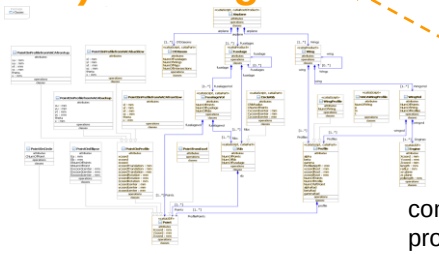
2) design rules

knowledge

- semantics
- patterns, couplings

compilation process

creative act emergence



1) ontologies

information

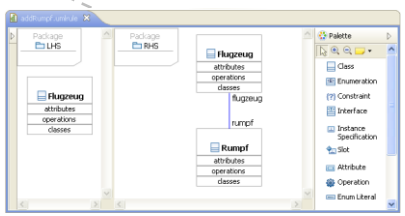
- syntax
- information representation, formats, interfaces

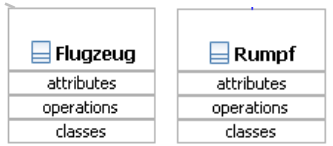
compilation process

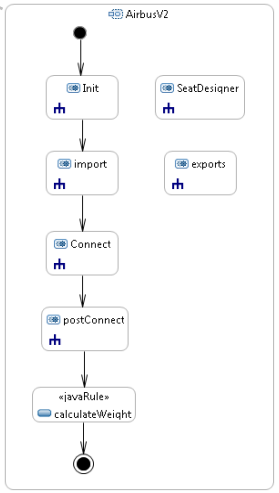
creative act emergence

raw data

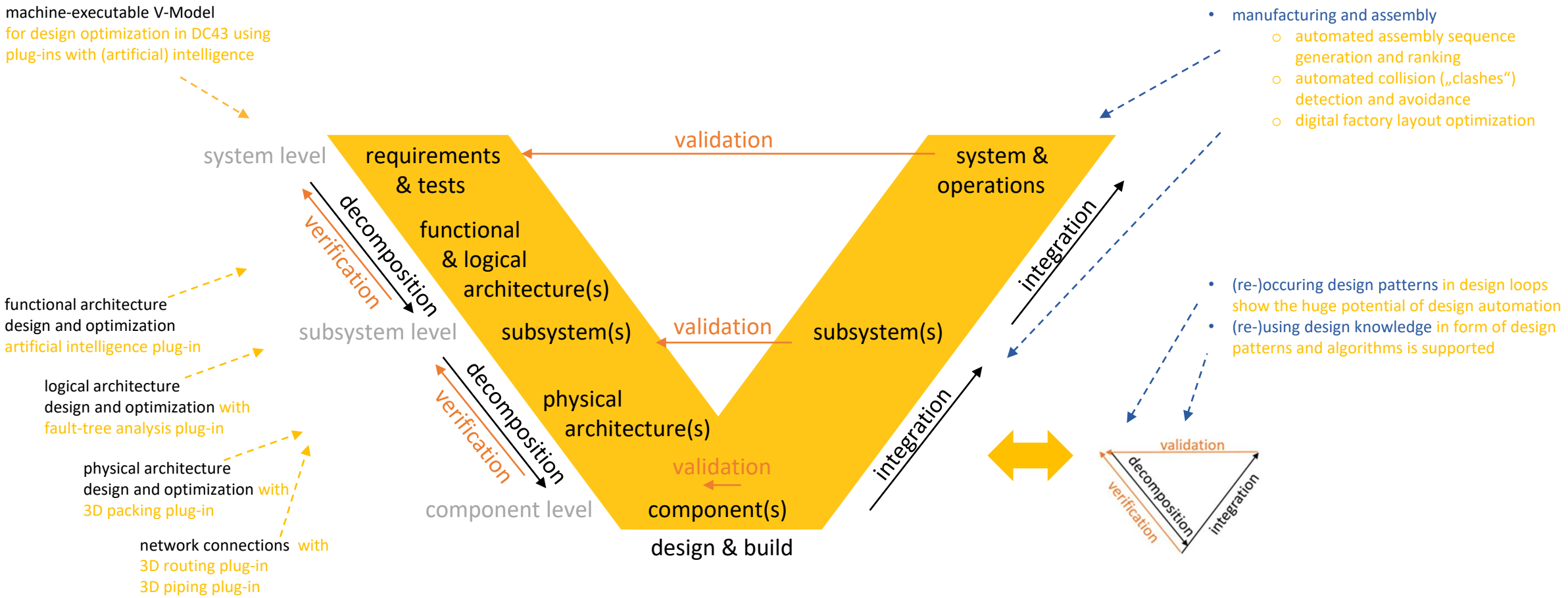
- named values: 11010101, 10111011, 01001010, 00110110, 10101000, 1111, 001, 10110101





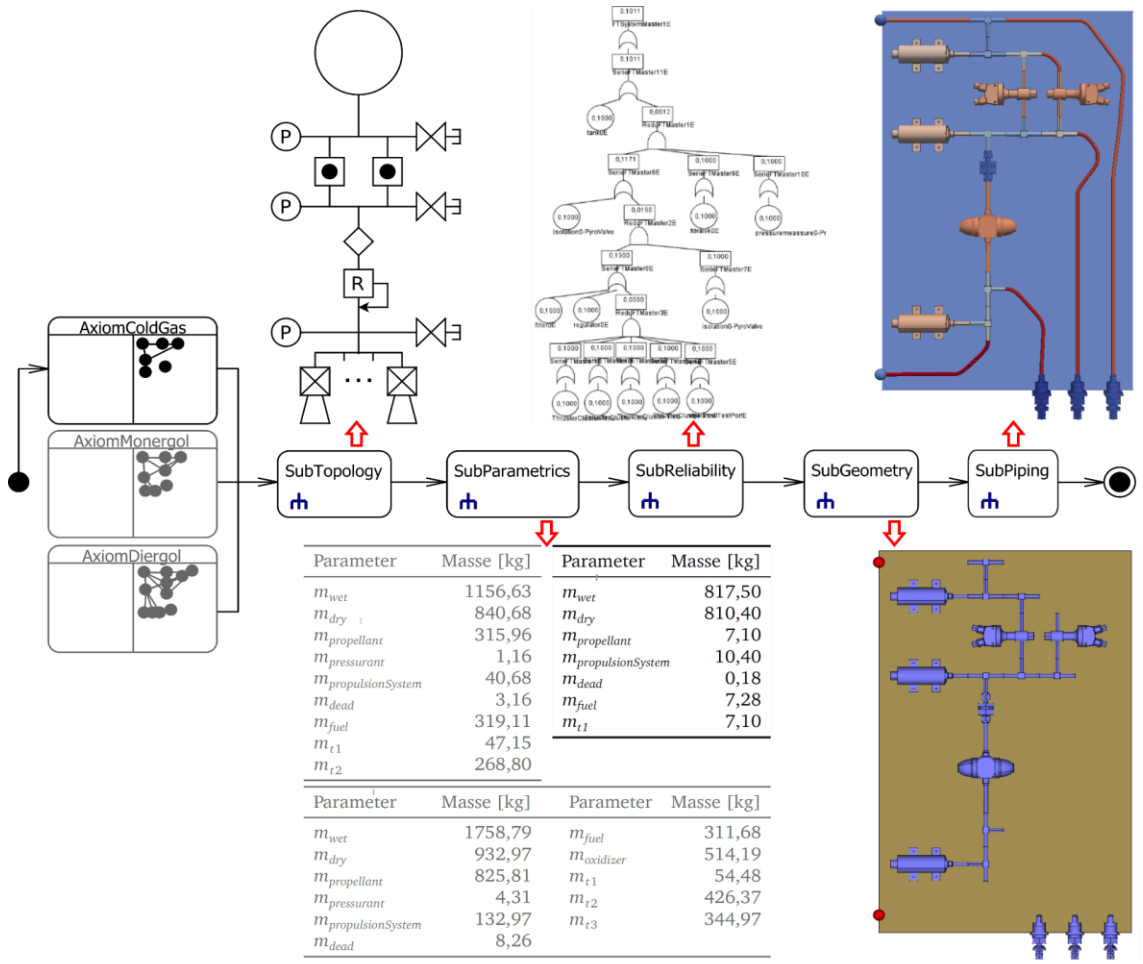


Machine-Executable V-Model of MBSE with Graph-Based Design Languages



Machine-executable V-Model of Satellite Propulsion System

requirements
- Δv
- I_{sp}

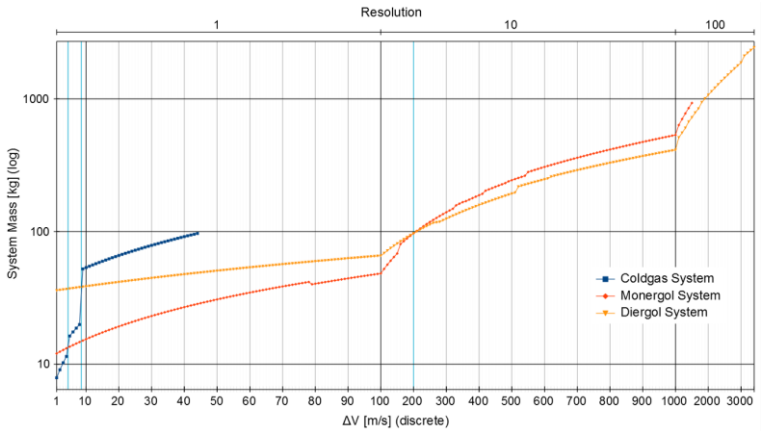


Parameter	Masse [kg]
m_{wet}	1156,63
m_{dry}	840,68
$m_{propellant}$	315,96
$m_{pressurant}$	1,16
$m_{propulsionSystem}$	40,68
m_{dead}	3,16
m_{fuel}	319,11
m_{t1}	47,15
m_{t2}	268,80

Parameter	Masse [kg]
m_{wet}	817,50
m_{dry}	810,40
$m_{propellant}$	7,10
$m_{propulsionSystem}$	10,40
m_{dead}	0,18
m_{fuel}	7,28
m_{t1}	7,10

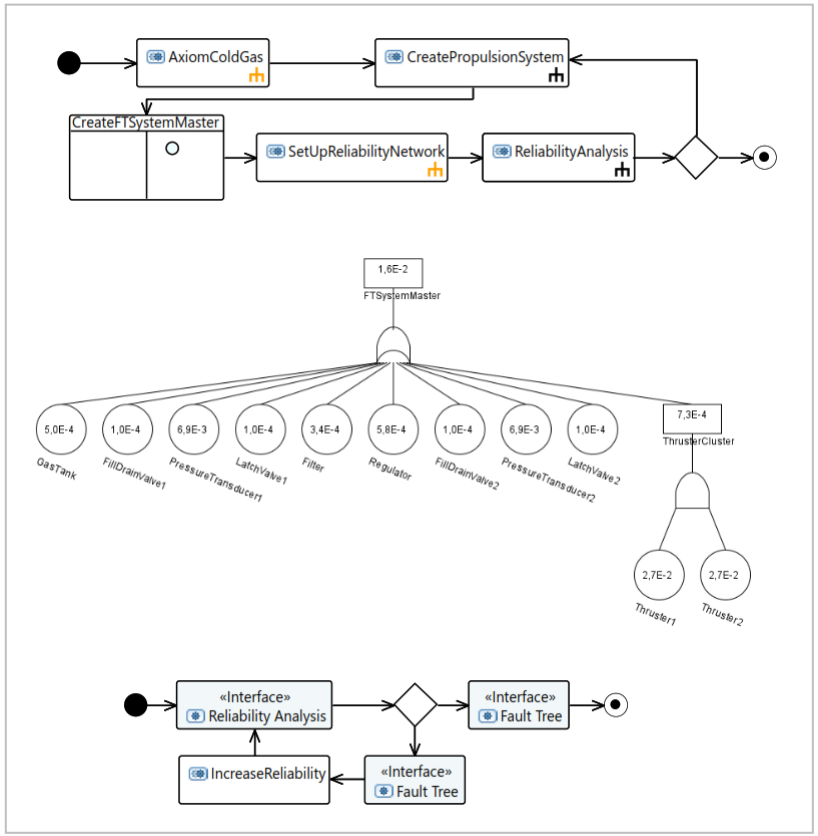
Parameter	Masse [kg]	Parameter	Masse [kg]
m_{wet}	1758,79	m_{fuel}	311,68
m_{dry}	932,97	$m_{oxidizer}$	514,19
$m_{propellant}$	825,81	m_{t1}	54,48
$m_{pressurant}$	4,31	m_{t2}	426,37
$m_{propulsionSystem}$	132,97	m_{t3}	344,97
m_{dead}	8,26		

design process automation
- functional architecture generation
- logical architecture generation
- physical architecture generation

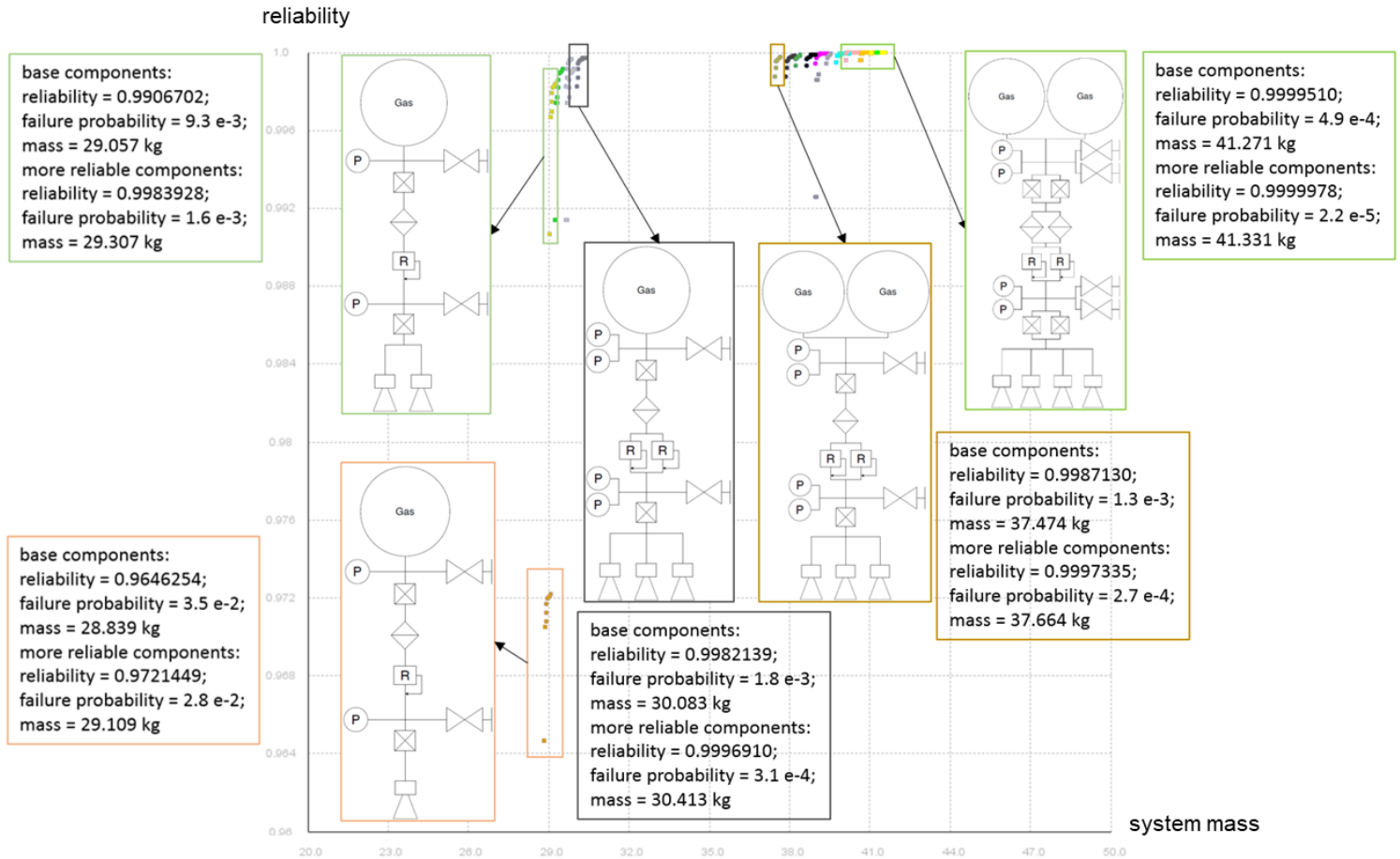


design space exploration
- technology change points
  (intersection points turquoise lines)
- technology hull

Logical Architecture Generation with FTA-Plug-In (Engineering as a Service)

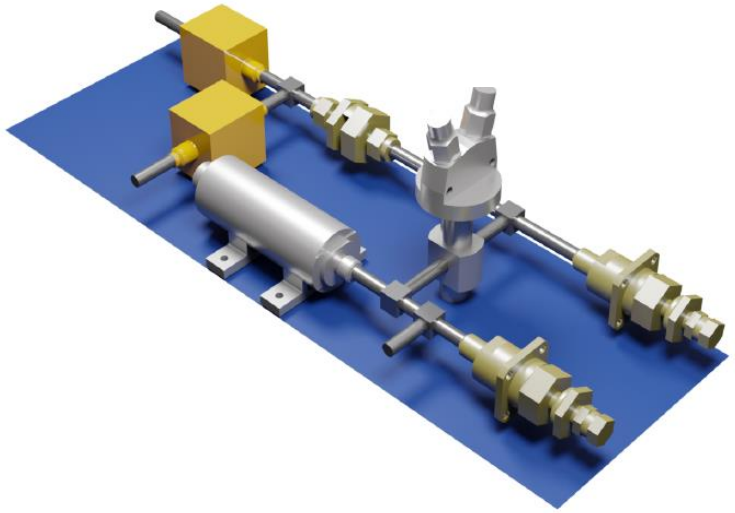


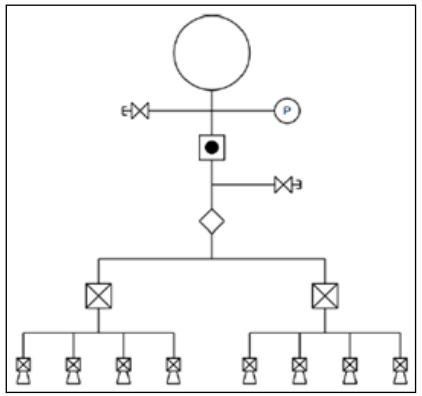
© 2020 by Marius Riestenpatt gen. Richter. Modeled, programmed and generated using graph-based design languages with fault-tree analysis (FTA)-Plugin



DoE study courtesy © 2020 by Marius Riestenpatt gen. Richter. For details see:
Riestenpatt gen. Richter, M. and Rudolph, S.: A scientific discourse on creativity and innovation in the formal context of graph-based design languages. 13th Anniversary "Heron Island" Conference Workshop on Computational and Cognitive Models of Creative Design (HI'19), Heron Island, Queensland, Australia, December 15–18, 2019.

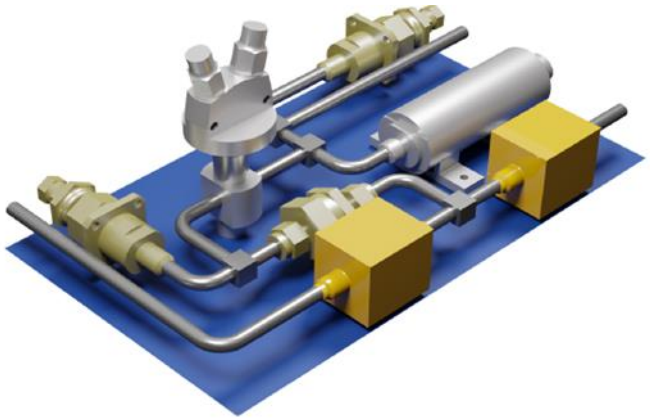
Physical Architecture Generation with 3D Packaging Plug-In (Engineering as a Service)

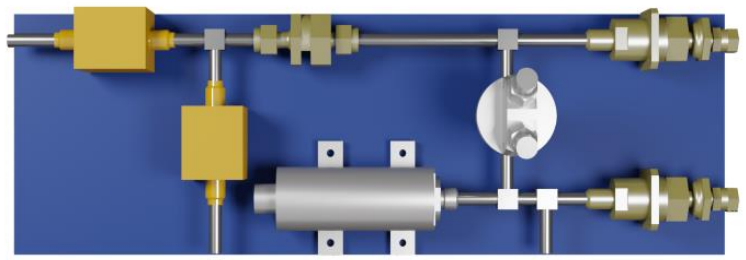




packing algorithm
maps logical architecture

to physical architecture
w/o side-constraints

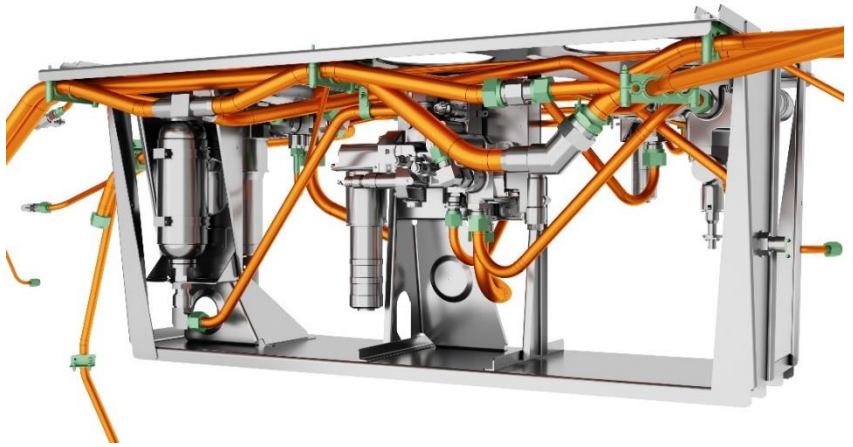


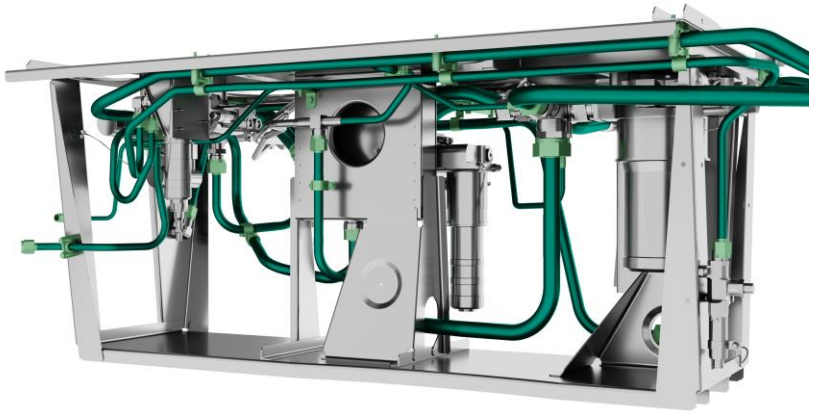


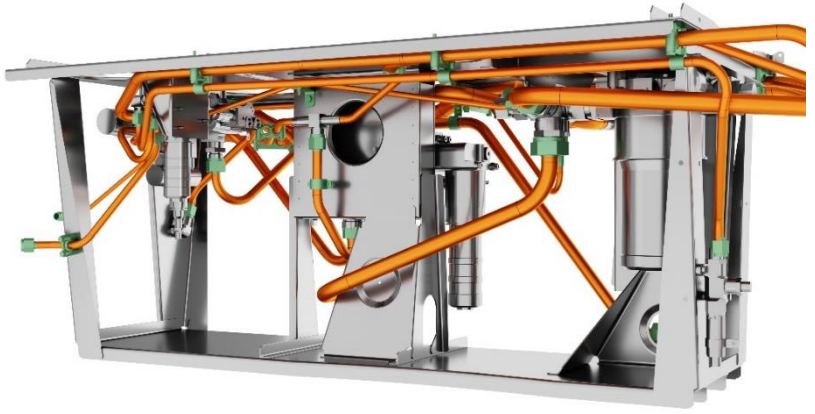
automatically generated packing
(bendings forbidden, area 457 cm^2)

automatically generated packing
(bendings allowed, area 361 cm^2)
22% space savings by 6 bends

Physical Architecture Generation with 3D Piping Plug-In (Engineering as a Service)







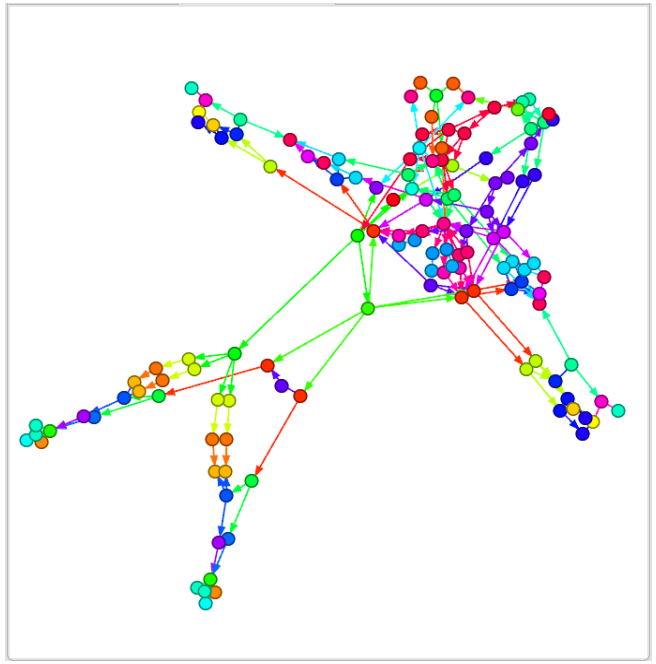
Neumaier, M.; Kranemann, S.; Kazmeier, B.; Rudolph, S.:
Automated Piping in an Airbus A320 Landing Gear Bay Using Graph-Based Design Languages.
Aerospace 2022, 9, 140.
https://doi.org/10.3390/aerospace9030140

Physical Wing Architecture Optimization System
https://cordis.europa.eu/project/id/865044

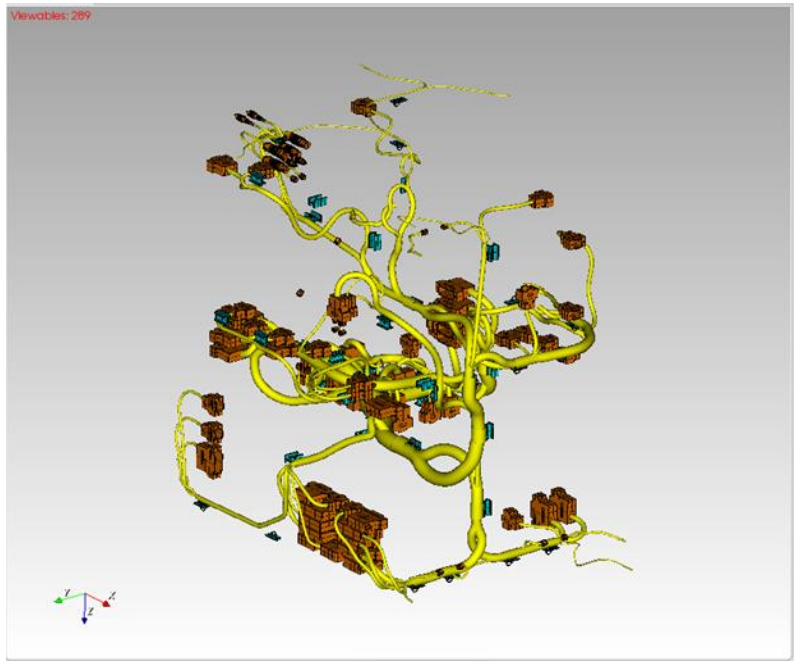
manually designed series solution A320
(pipe lengths 22.6 [m], 107 bends)

automatically generated solution
(pipes lengths 20.414 [m], 95 bends)
10% length/weight savings with 12 bends less

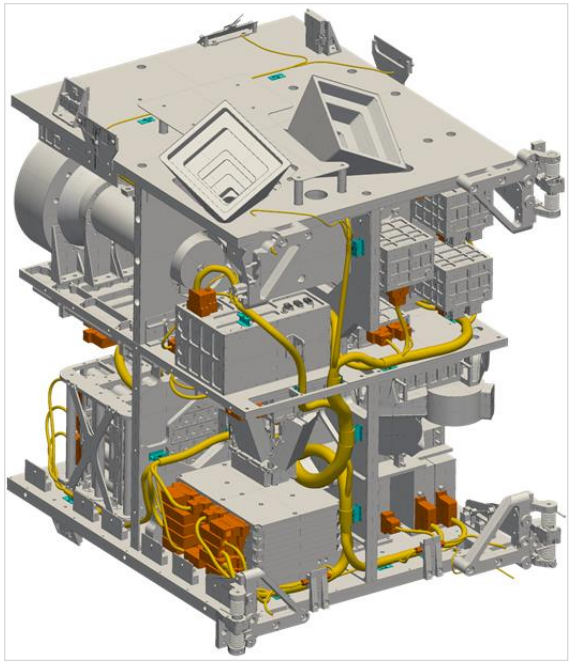
Physical Architecture Generation with 3D Routing Plug-In (Engineering as a Service)



© 2018 design graph of wire harness generated with 3D routing plug-in



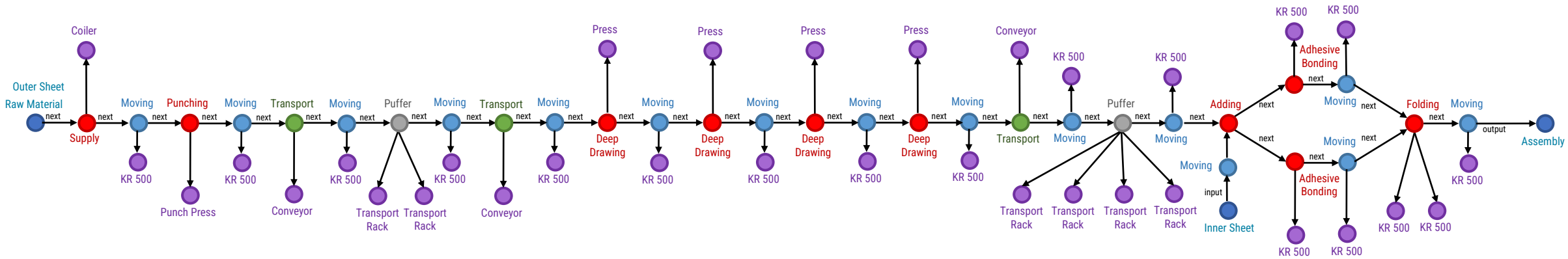
harness shown was automatically generated in 15 mins on PC with i5@3,2GHz, 32GB RAM



CAD model courtesy © 2018 Institute for Space Systems, University of Stuttgart
with integrated wire harness automatically generated with 3D routing plug-in

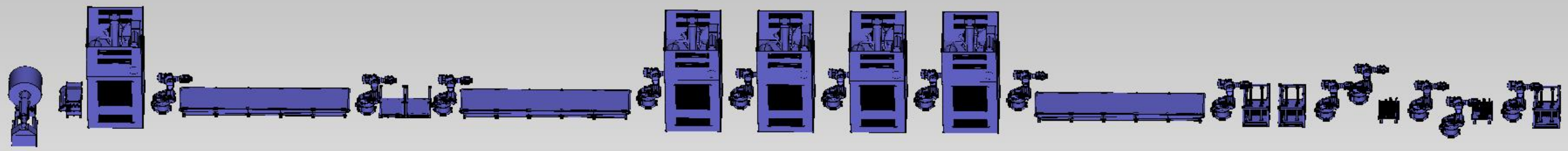
in the MANUNET project FORTIFIER (MNET20/ICT-3763)
- a data converter to STEP AP242 with EWIS content was implemented, including a mapping of VEC to STEP AP242
- a digital factory simulation for wire harness manufacturing was generated

Digital Factory Generation with Digital Factory Plug-In (Engineering as a Service)



digital factory generation using design languages

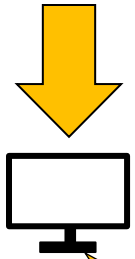
- 2D layout generation, throughput design of transfer line
- generation of digital factory simulation
- VR-capabilities using HTC VIVE



Graph-Based Design Languages

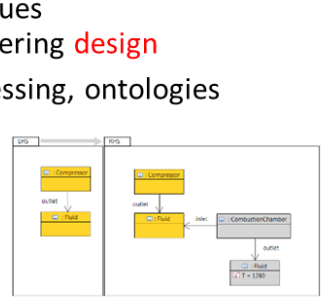
human execution

human reaction time 10^{-1} [s]
versus
processor cycle time 10^{-9} [s]



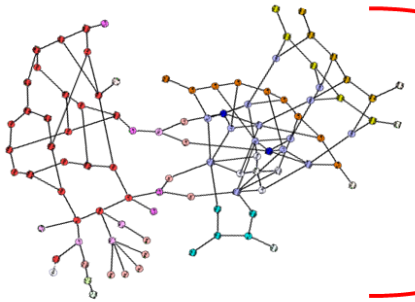
machine execution

- artificial intelligence techniques
  for creative power in engineering design
  - natural language processing, ontologies
  - graph algorithms
  - constraints processing
  - …



with

- formal language processing
  in computer science which can be
  - translated/compiled
  - traced/validated/verified
  - advanced system analysis
  - …



„design intelligence“
intelligence in form of
a search problem

„design automation“
compilation and processing
in a software platform

less cost

less time

KPIs < 50% cost
< 50% time

artifical intelligence
enhanced engineering
design automation
(intelligent MBSE)

10^{X} variants

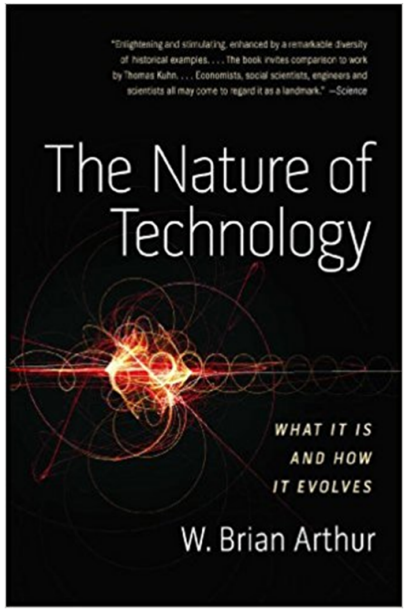
more exploration

Philosophical Foundation of Formal Languages for Design

The philosopher William Brian Arthur postulated in his book[1] *"The Nature of Technology"* (2009) in chapter 5.2 entitled *"Design as Expression Within a Language"*:

- A new device or method is put together from the available components - the available vocabulary – of a domain. In this sense a domain forms a language; and a new technological artifact constructed from components of the domain is an utterance in the domain's language.
- And it means that the key activity in technology – engineering design – is a form of composition. It is an expression within a language (or several).

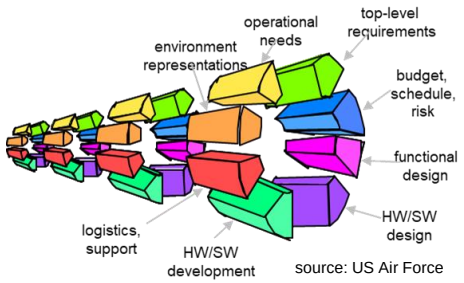
[1]Arthur, W.B.: The Nature of Technology - What It Is and How It Evolves. New York, Free Press, 2009.



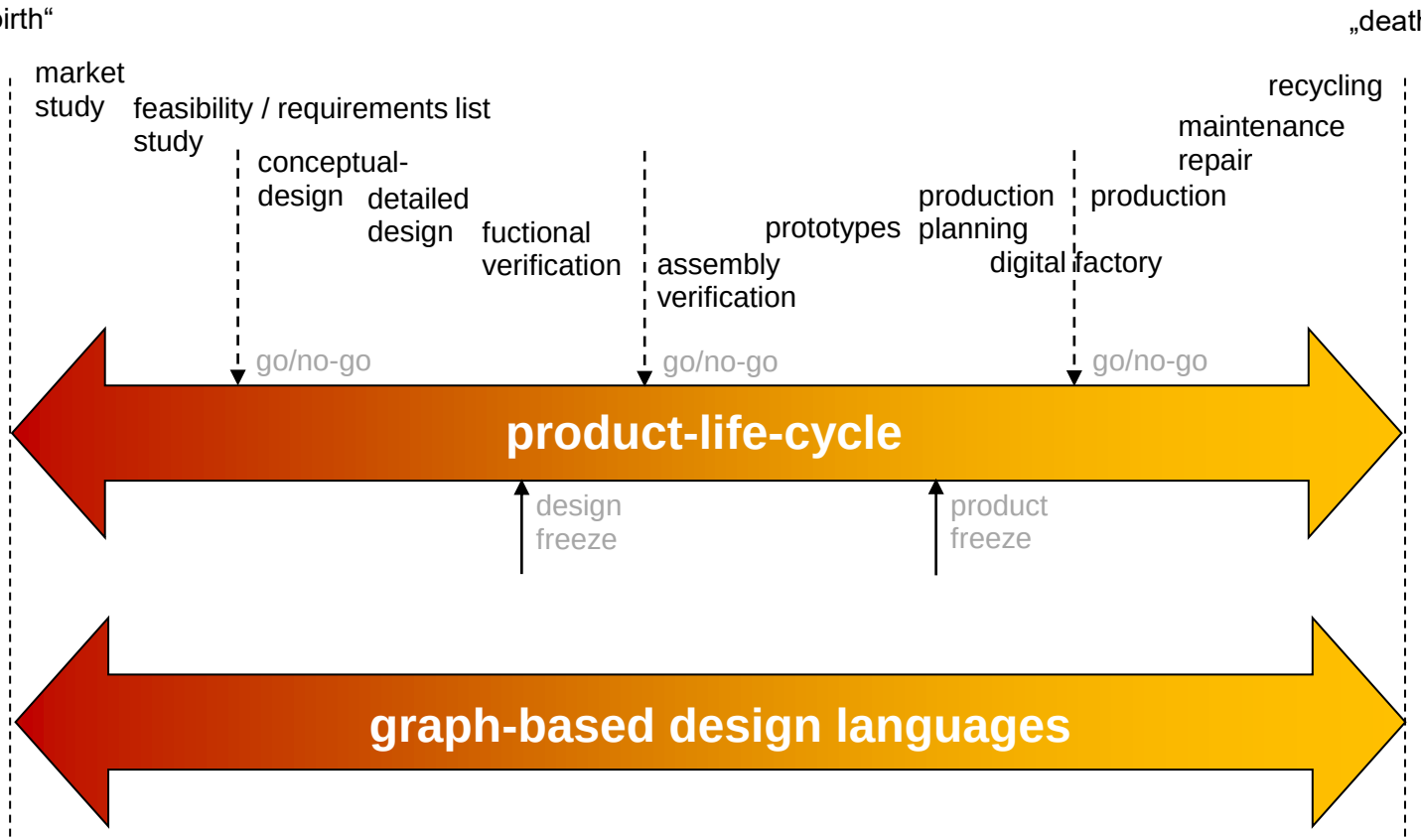
Graph-Based Design Languages provide Digital Continuity, Consistency and Interoperability

todays deficiencies

- no/limited data consistency
- no/limited process continuity
- no/limited tool interoperability
- no/limited scope, scalability
- knowledge islands, data lakes….

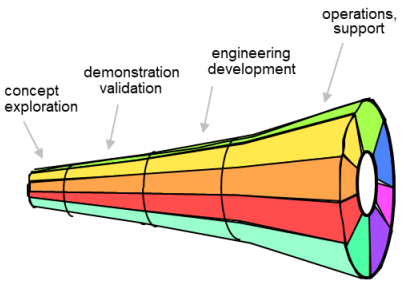


source: US Air Force



future capabilities

- digital data consistency
- digital process continuity
- digital tool interoperability
- full scope, full scalability
- knowledge re-use, ….



source: US Air Force

Sponsors

This research and development project was partly funded by the German Federal Ministry of Education and Research (BMBF) under the research program "Future of Value Creation“ (program/funding code: MNET20/ICT-3763) and is supervised by the Project Management Agency Karlsruhe (PTKA).
The responsibility for the content of this publication lies solely with the author.

Dieses Forschungs- und Entwicklungsprojekt wurde teilweise mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) im Forschungsprogramm “Zukunft der Wertschöpfung”(Programm/Förderkennzeichen: MNET20/ICT-3763) gefördert und vom Projektträger Karlsruhe (PTKA) betreut.
Die Verantwortung für den Inhalt dieser Veröffentlichung liegt allein beim Autor.