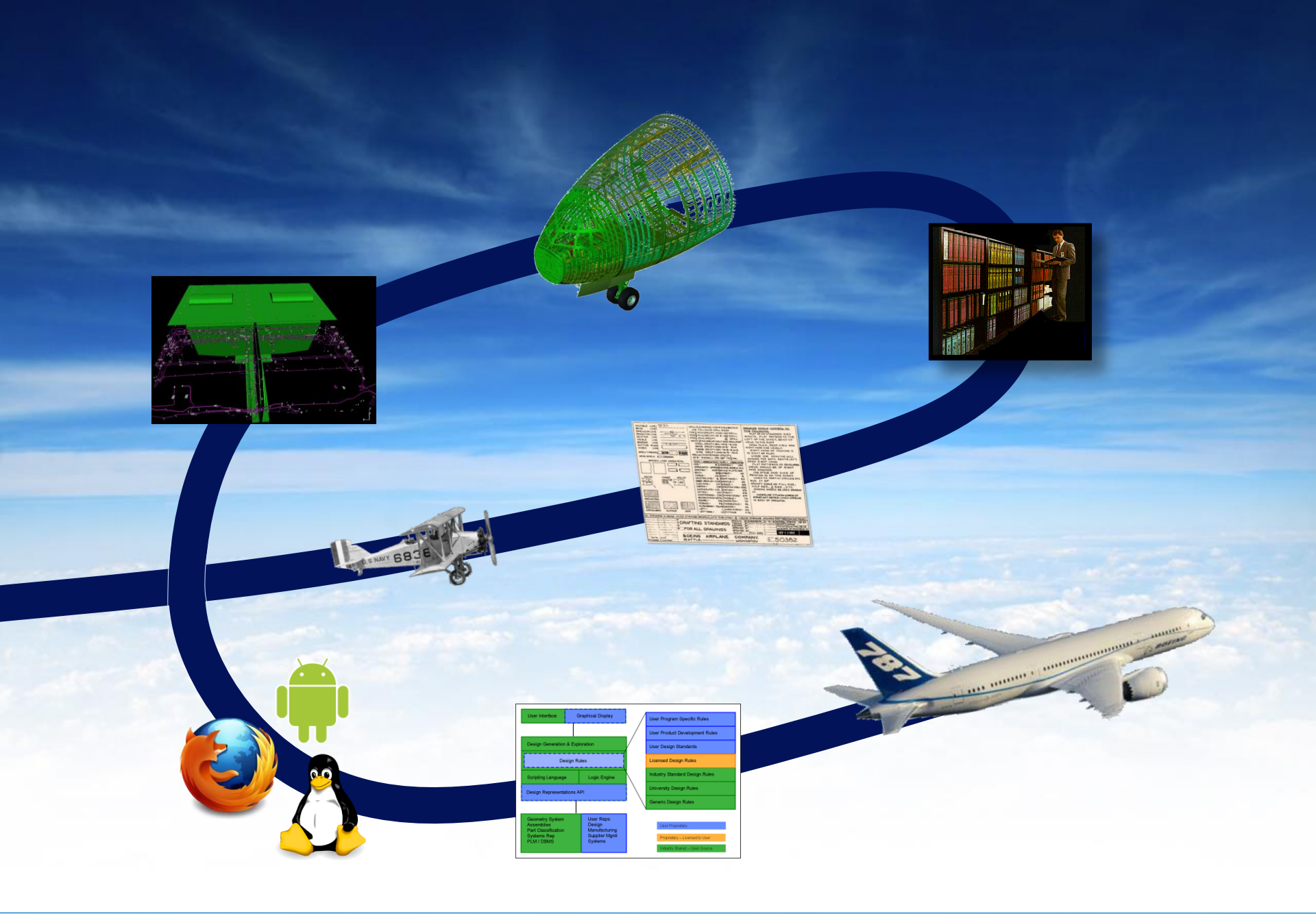
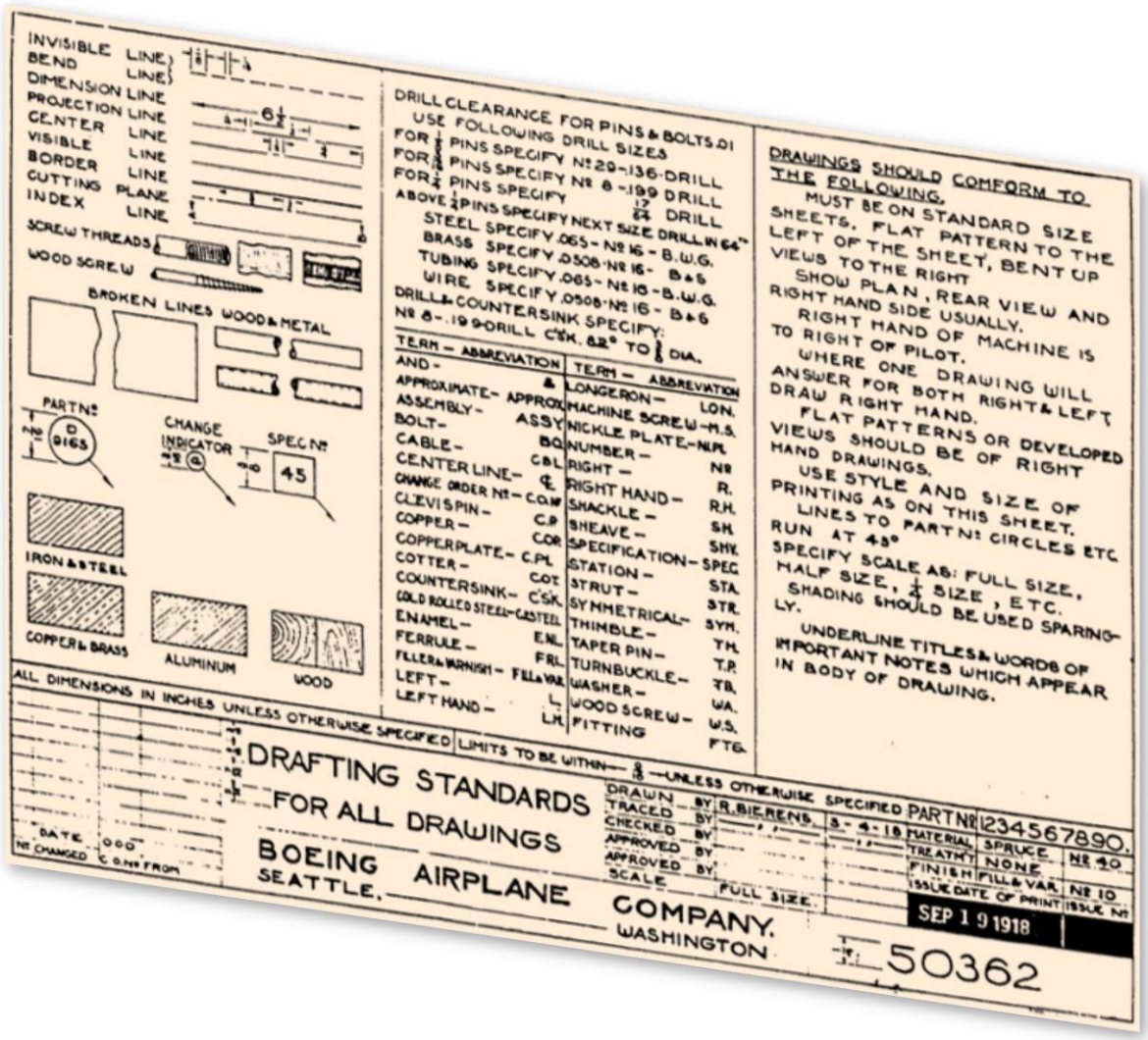
Generative Design and Automated Reasoning in the Design of Aerospace Systems

Jeff Heisserman
Boeing

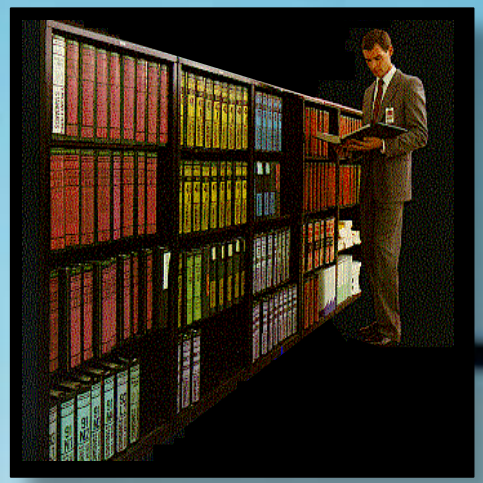


Boeing Product Development – circa 1918

Boeing Model C
(1916, 56 built)



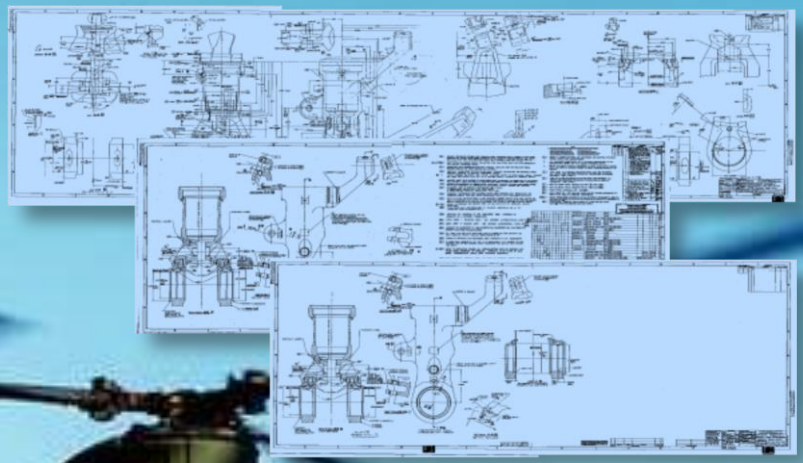
Boeing Product Development – 1970s



9000 pages of part standards

8600 part attribute rules

475 BAC process specs

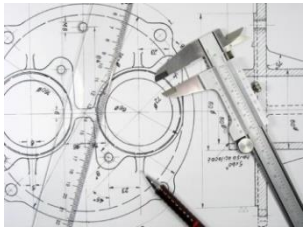


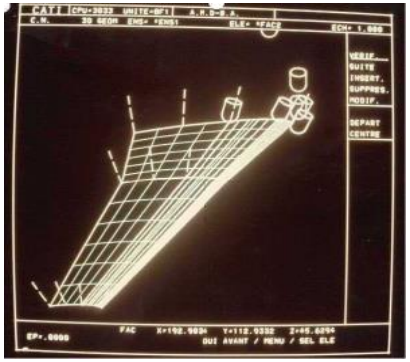
40 billion **part numbers**

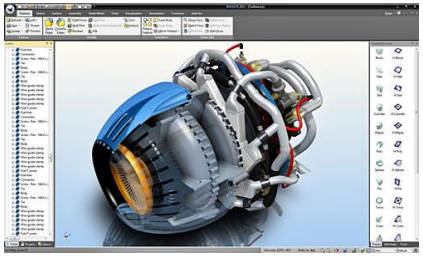
Evolution of Product Design

Global Product Data Interoperability Summit | **2015**

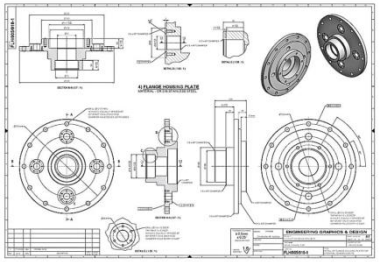
- **Drawings**
- **Lots of paper**







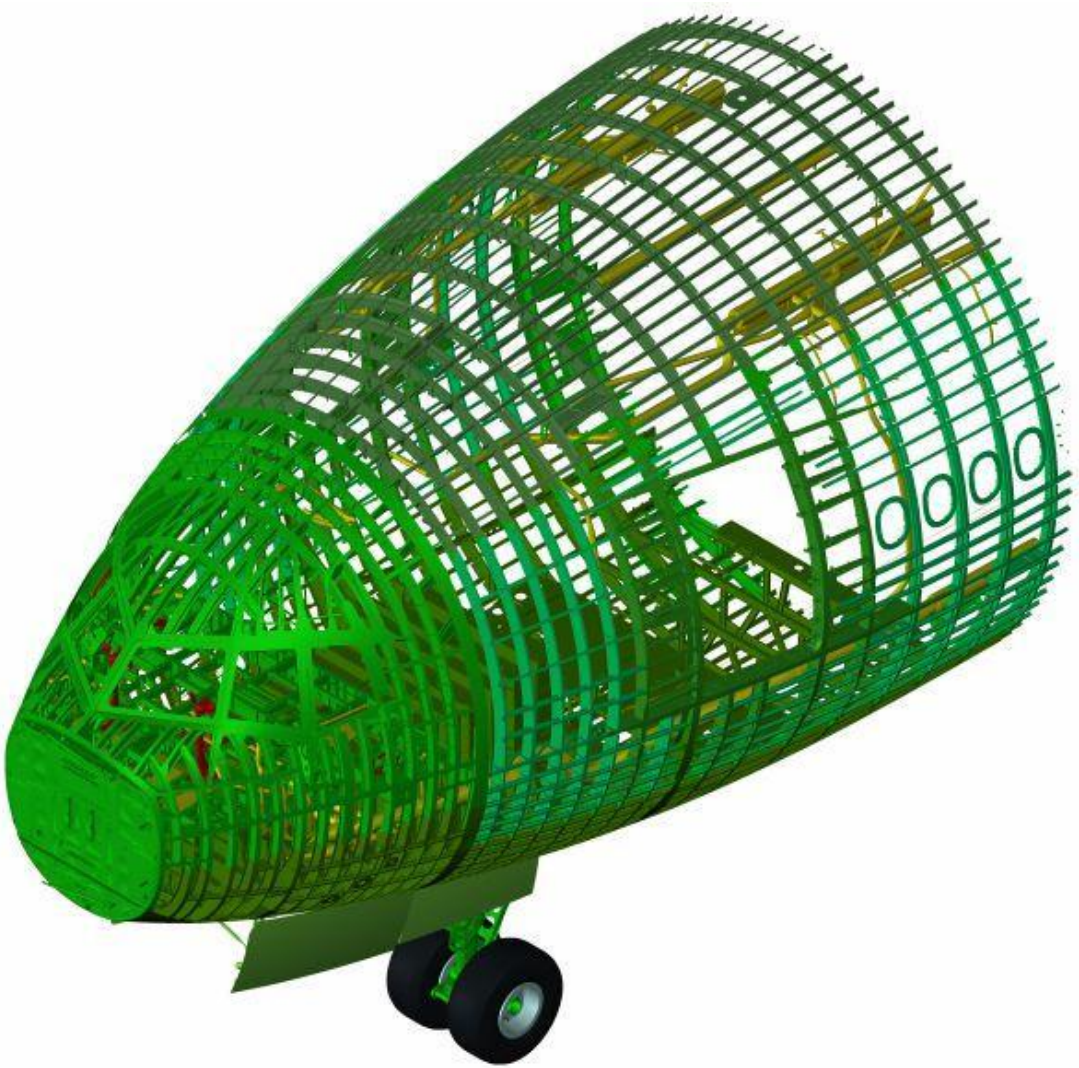
- **Digital designs**
- **Parametric modeling**
- **Model based definition**
- **3D GD&T**
- **Product Standards Data**
- **PLM, PDM, ERP, MES**
- **Virtual Manufacturing**

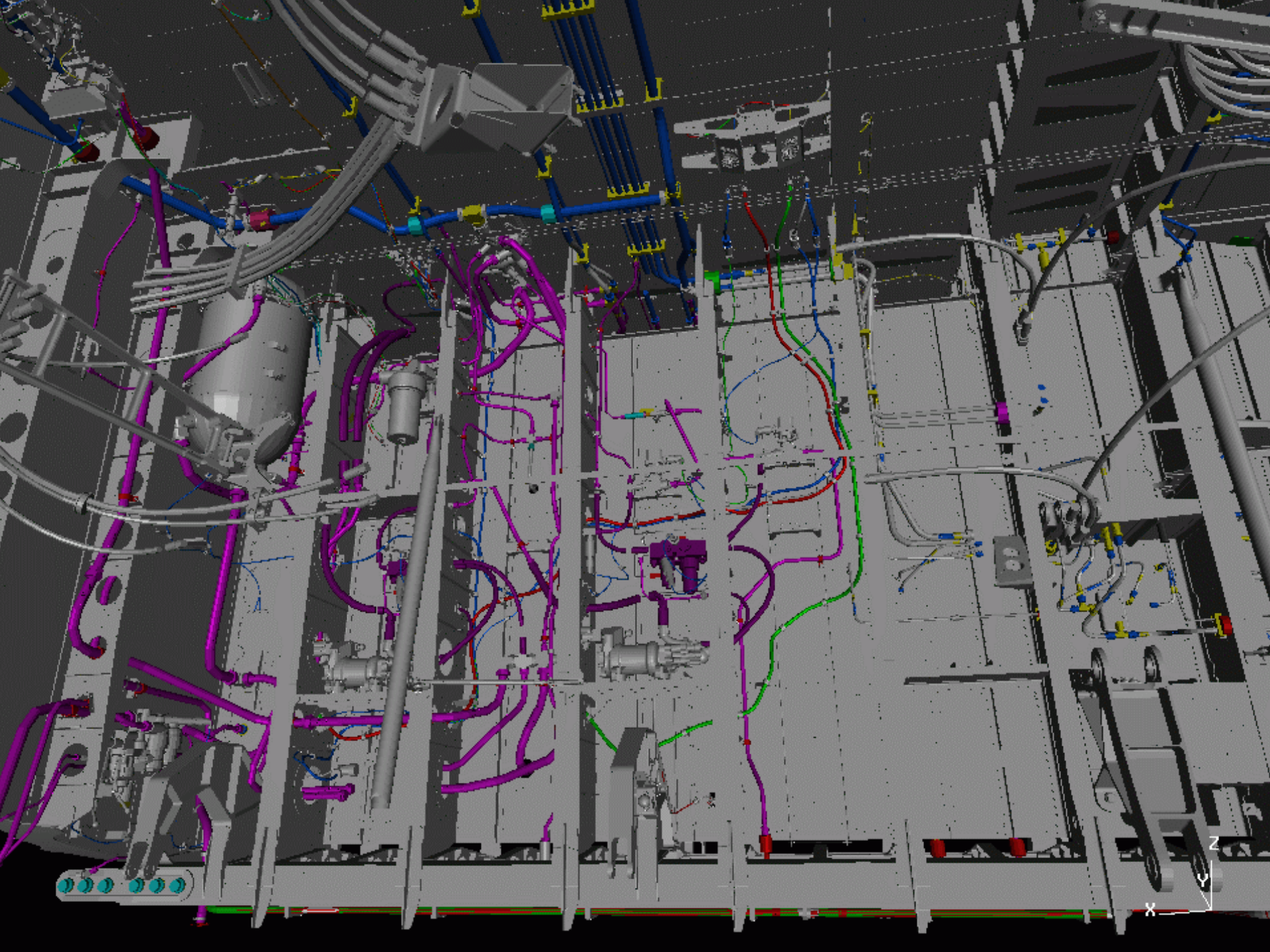


- **Digital drawings**
- **Early solid modeling**
- **Data entry systems**

1960 1970 1980 1990 2000 2010

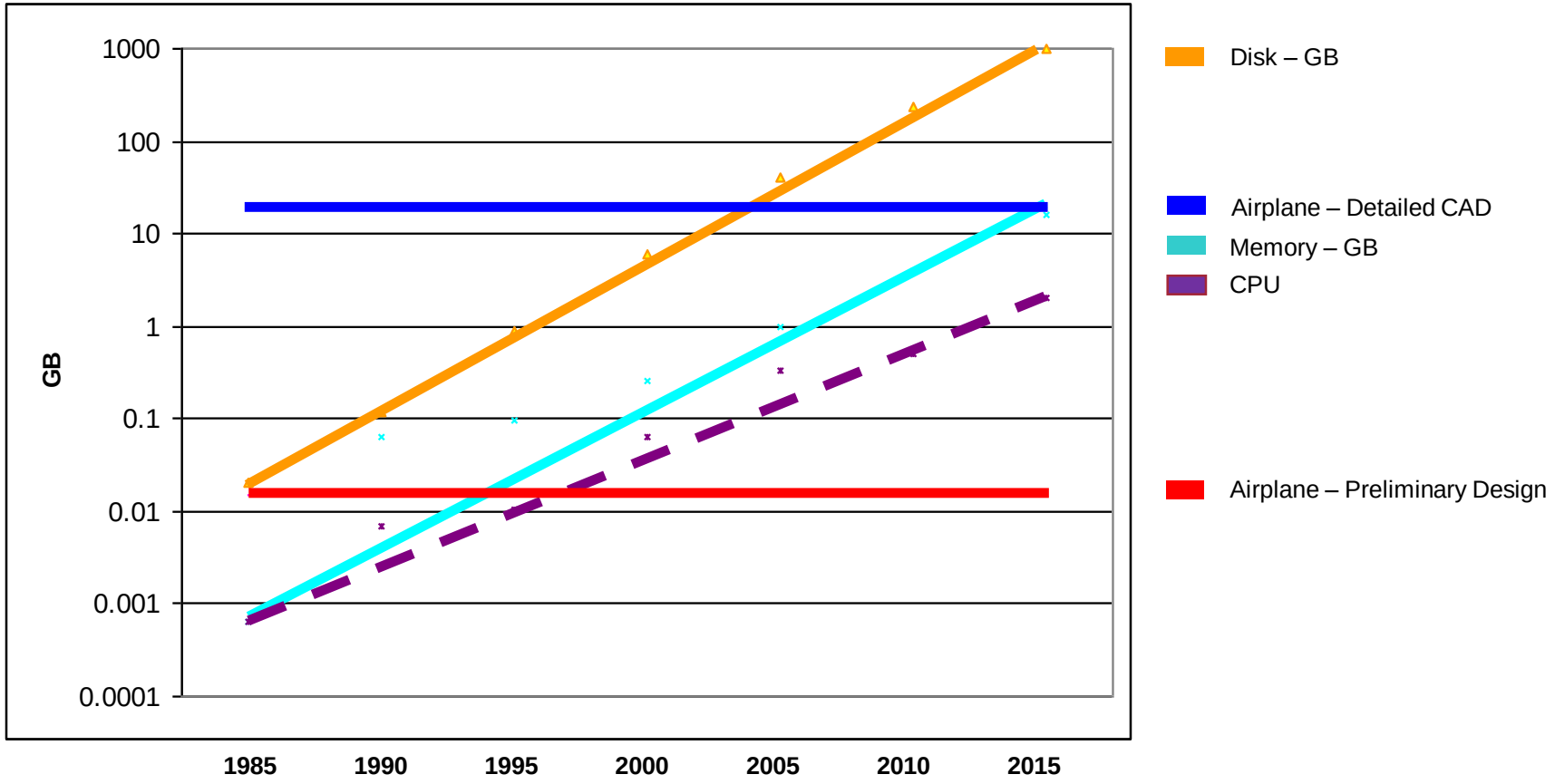
Boeing Product Development – 1990s




Z
Y
X

Desktop Computational Capacity

or Moore's Law as it applies to aircraft design



Computational Design

We want to
- Automate repetitive work
- Develop better products within shorter cycle times
- Reduce development costs

We are facing huge engineering challenges
- People are creative and flexible, but expensive
- Computers are systematic, tireless, thorough, and cheap
- … and we just don't have enough engineers

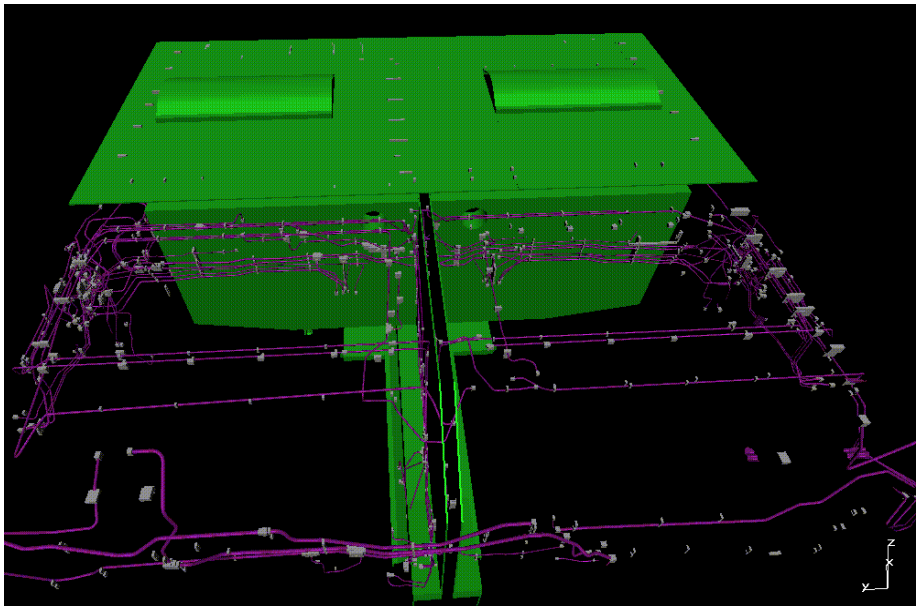
We want to create innovative designs
that we just can't create otherwise

Computational Design

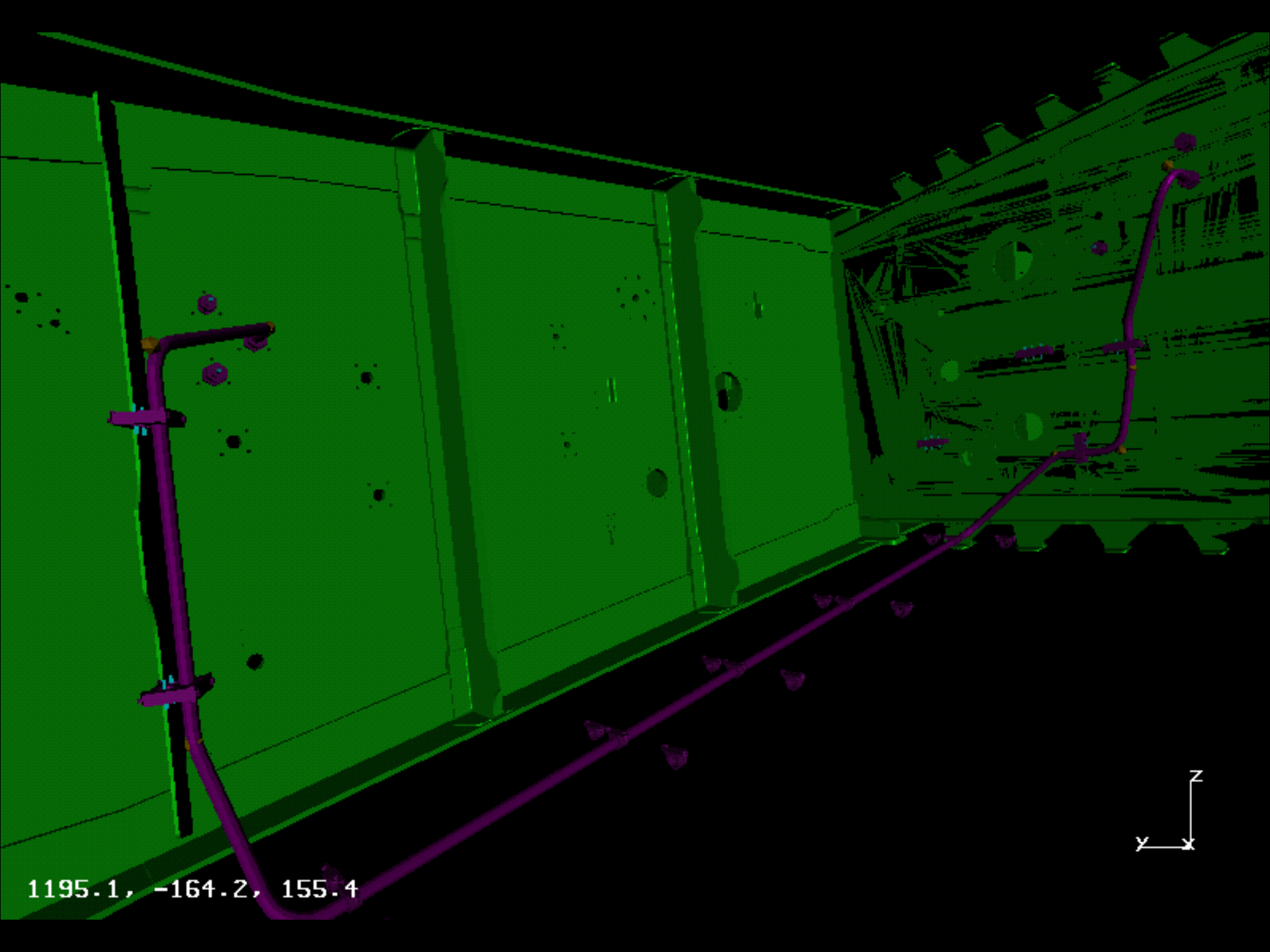
- Knowledge-Based Engineering (KBE)
  - the tools and technologies for encoding engineering knowledge and methods in software to analyze and generate aircraft, component, tooling and other designs
- Generative Design
  - automated reasoning with interactive and automated design methods - based on the concepts of shape grammars and spatial grammars - that are used for generating designs and exploring spaces (languages) of designs.
- By encoding our knowledge
  - in modular, flexible design rules
  - applying them to generate and check our designs
  - use our computers as intelligent design assistants
  - leaving the more difficult - and more interesting - problems to our engineers

Genesis / KIRTS
Knowledge-based Integrated Routing Tool for Systems

- used for the design of aircraft systems tubes, pipes, ducts, hoses, cables and bundles, including clamps, clamp blocks, fittings
- for hydraulic systems, fuels, cabin air, emergency oxygen, smoke detection, fire suppression, fresh water, gray water, sewer
- contains design rules for engineering, manufacturing, fabrication and maintenance



767-400 Hydraulic Systems
main landing gear bay

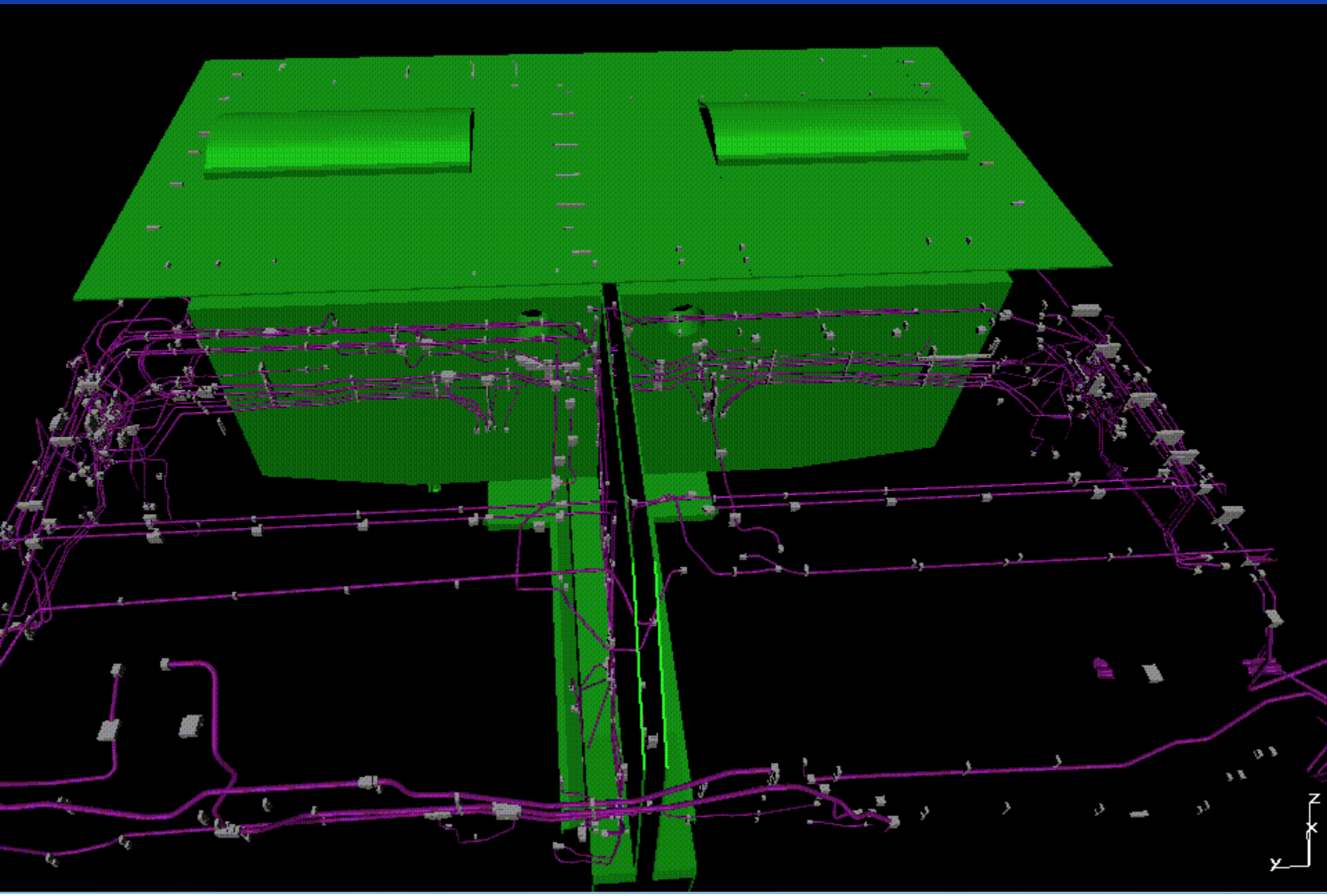

1195.1, -164.2, 155.4
Z
Y
X

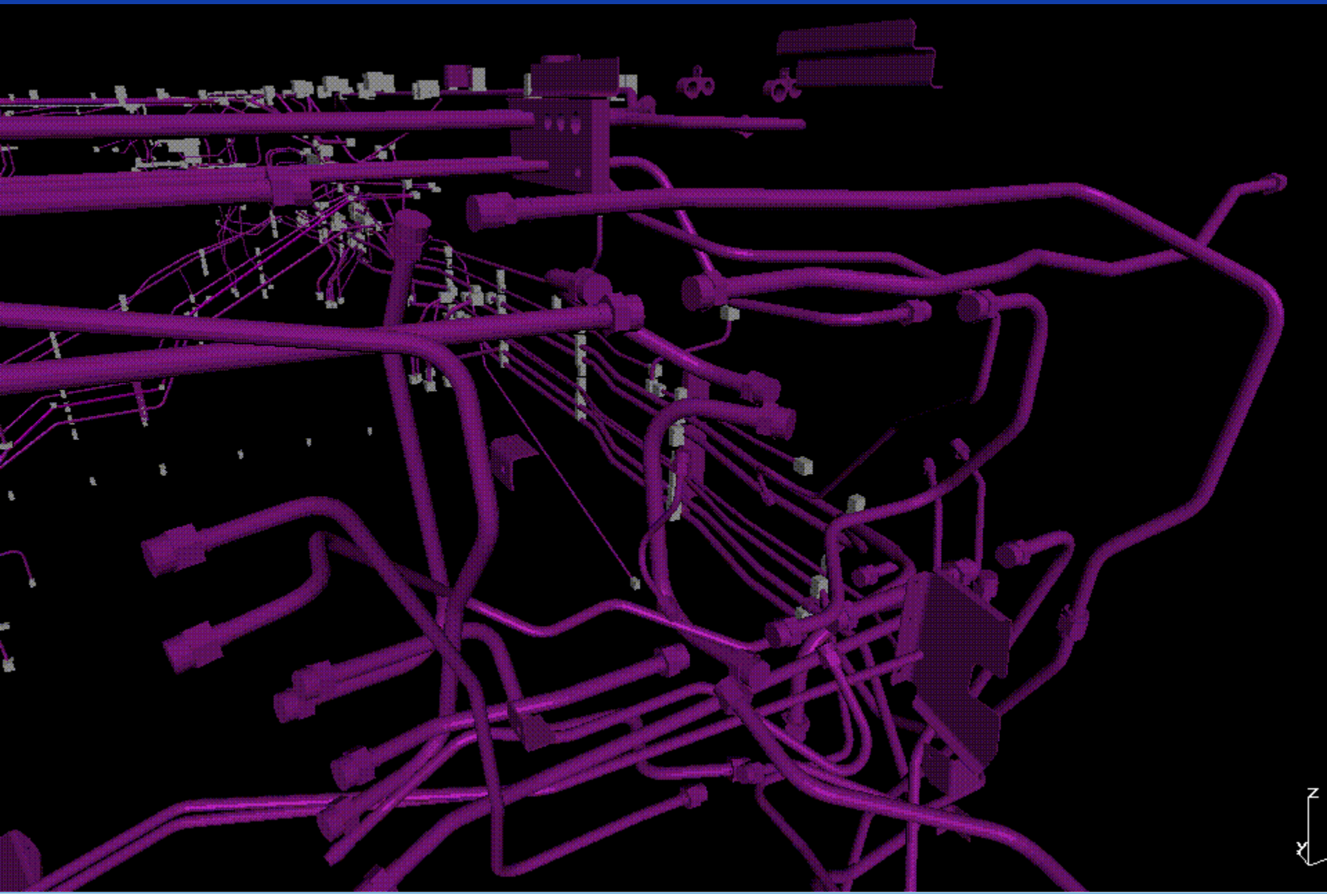
A Genesis / KIRTS Design Rule


```
condition(tube1,
    'Tube has no geometry.').

lhs(tube1, [Tube], [Part1, Part2]):-
    schematic_tube_connections(Tube, Part1, Part2),
    in_context(Tube),
    not occurrence_has_geometry(Tube),
    occurrence_has_geometry(Part1),
    occurrence_has_geometry(Part2).

description(tube1,
    'Create a tube and its fittings.').

rhs(tube1, [Tube]):-
    make_tube(Tube).
```

Logical Reasoning about Designs

- **Axioms / base representations**
  - **solids & assemblies**
  - **part type classifications**
  - **part interfaces**
  - **system schematics**
- **First-order logic**
  - **map from design rules to base representations**
- **Goal clauses**
  - **specify design conditions**

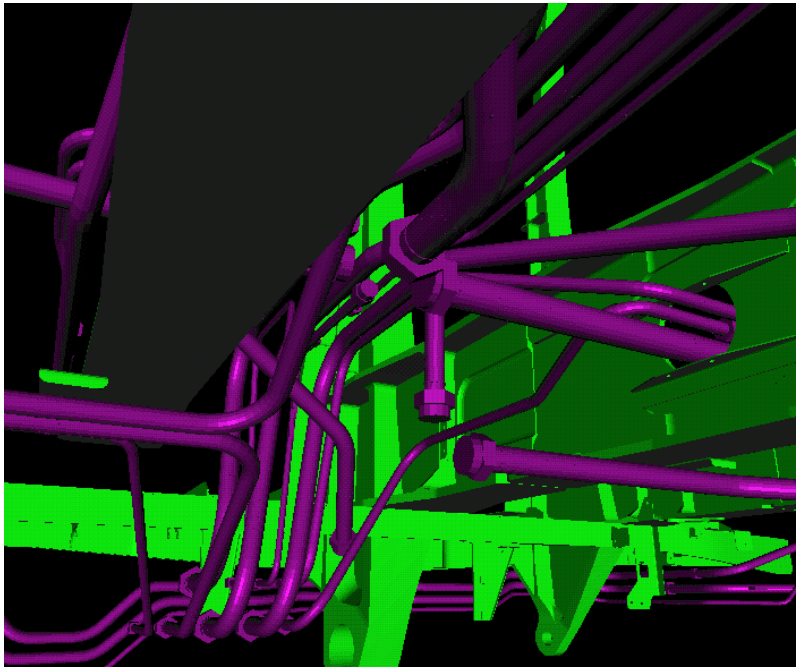


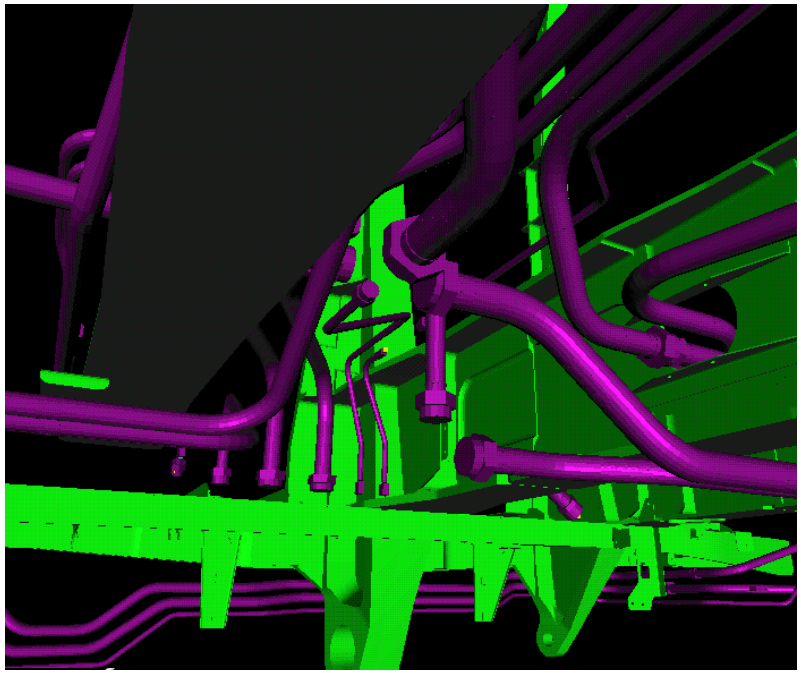


Design Generation & Exploration

- **Design generation**
  - **interactive generation and modification using design rules**
  - **automated generation of detailed designs**
- **Design checking**
  - **correctness and consistency checking**
  - **automated analysis**
  - **integrated simulation**
- **Exploring spaces of designs**
  - **automated generation of alternative designs**
  - **design iteration and quality evaluation**
  - **design comparison and merging**

Comparing Designs in Genesis/KIRTS





From Research to Production

Carnegie Mellon University / Queen Anne Houses

- Representing 3D polyhedral solids
- Logical reasoning
- Parametric shape rules
- Simple design generation

Boeing / Aircraft Systems

- Representing 3D solids, assemblies, part classes, ports, function
- Parametric shape (plus) rules
- Interactive rule application & automated generation
- Simple design evaluation & exploration
- Representing design variants (alternative) designs
- Fast interference and systems separation checking
- Large scale, in-context design
- Comparing designs

Open Source Computational Design

What we need is a *community effort* to build computational design tools that are

- high quality, robust and innovative
- interoperable with CAD/CAM/PDM systems
- embeddable in custom applications, analysis and optimization tools
- usable by everyone

Free and open source software

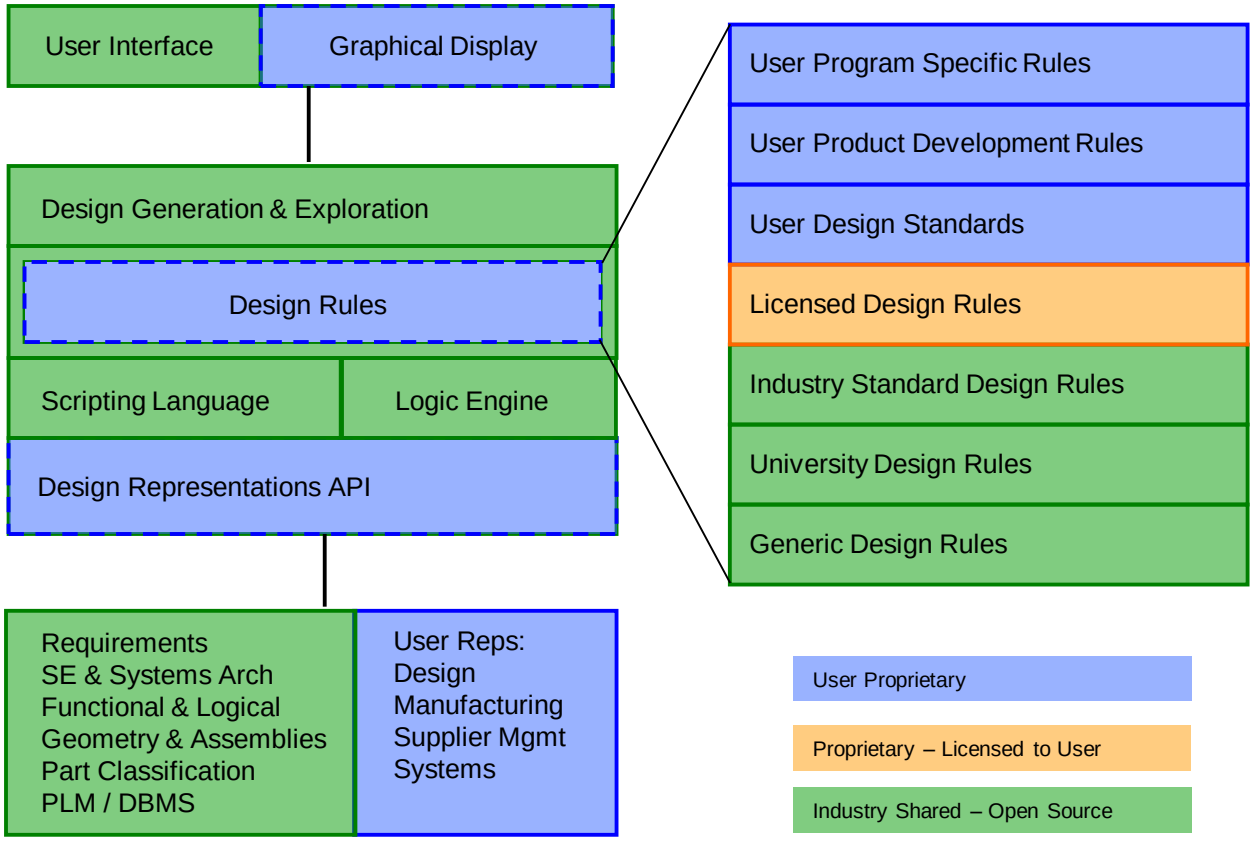
Open Source Computational Design

Why would Boeing want to develop a design system as Open Source?

We want to

- develop a robust, long lived system that can be used to develop multiple generations of airplanes and aerospace products
- share the costs of developing and maintaining the system
- move to a standard, neutral knowledge representation
- enable sharing of design rules, e.g. from NASA, FAA, universities
- encourage collaboration with universities, government labs and industry to develop state-of-the-art capabilities
  - speed technology transfer into Boeing
  - speed the application of new technologies to Boeing products
  - supports research collaboration with Boeing
  - facilitate hiring of outstanding engineers into Boeing

Architecture and User Scenario



- We use a modular architecture, with
  - modules that are shared open source
  - proprietary modules that are not shared
- We choose an open source license that allows users to use open source modules with proprietary modules – without having to share the proprietary modules
- Users can decide what design rules to share and what to keep to themselves
- Users can develop proprietary capabilities to augment the open source capabilities

Computational Design

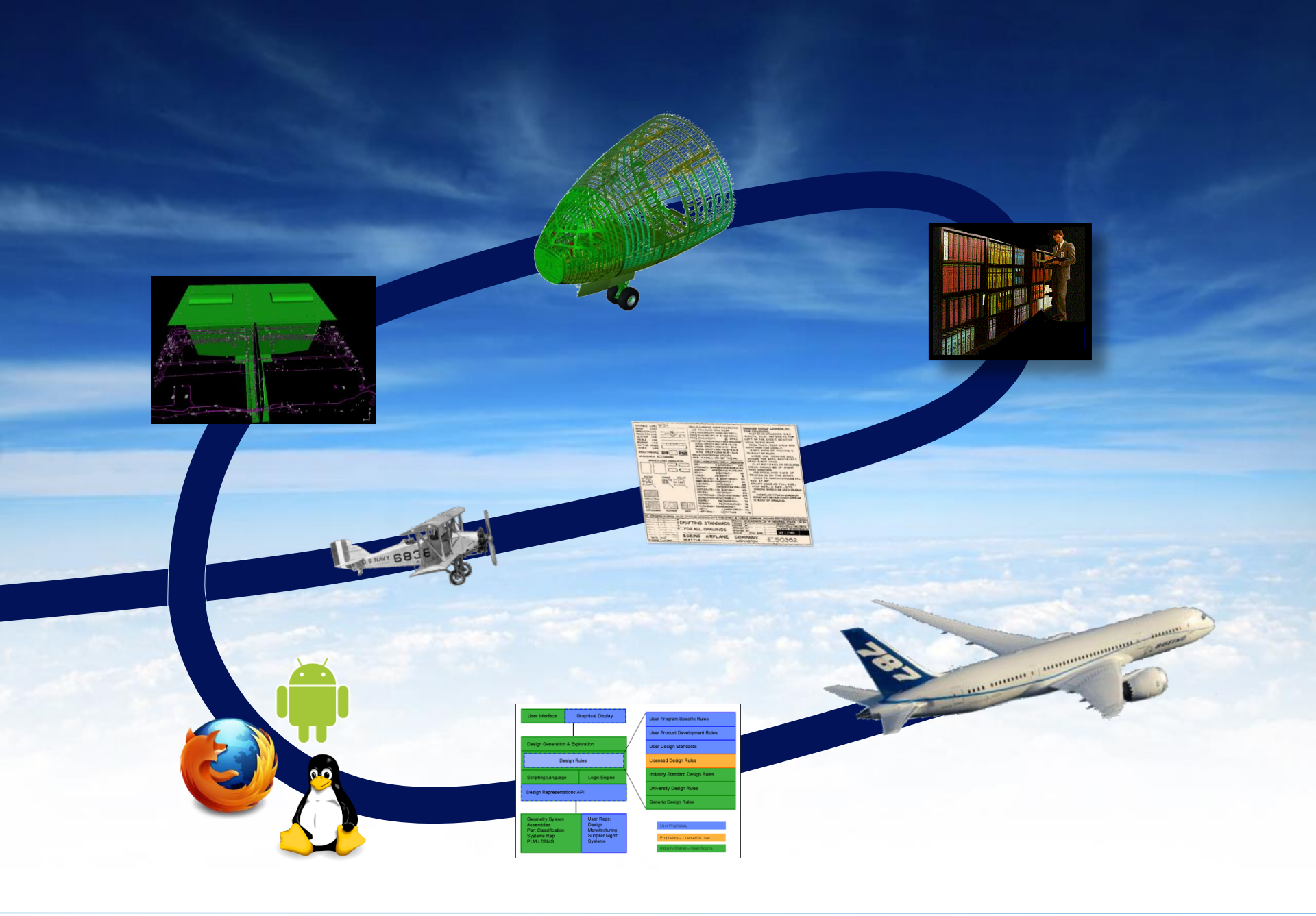
Our Engineering Grand Challenge:

Construct, apply and share engineering knowledge

- in an active, computational form
- able to analyze designs
- able to synthesize and
- able to optimize designs
- with the knowledge independent of the design system

… in other words

Global Product **Knowledge** Interoperability



Credits

OSSKBE

- Jeff Heisserman
- Chris Esposito
- Bill Brown
- Tom Velte

Genesis / KIRTS

- Jeff Heisserman
- Sean Callahan
- Raju Mattikalli
- Mike Drumheller
- Jan Vandenbrande
- Virgil Bourassa
- Ian Angus
- Bob Abarbanel
- Fred Holt
- Carl Pearson
- Greg Green
- Bob Perry
- Steve Cheng
- Harry George
- Dat Tran
- Bill McClay
- Mark Williams
- Eric Haberman
- John Kershinar
- Matt McMullen
- Mary Hopwood
- Frank Gosson
- Mike Galuska
- Britt Thompson
- Kelly Rogerson
- Rajendra Deonarine
- David Kamihara
- Mary McCartor
- Fred Aboosaidi
- Bernard Thompson
- Steven Wright
- … and more

BOEING