FMDesign Overview

Olof Johansson

Contents

FMDesign Context and architecture

Figure 1 shows the role of FMDesign amongst some of its surrounding engineering processes, connected with major workflow arrows. Engineering tools (FMDesign, ModelicaDB, ModelicaXML, Modelica Simulation tool) support some of the processes.

Engineering models are stored in files and in databases. The communication of design information between successive tools in the workflow can be handled through task automation and tool integration using web services [8] or shared access to databases and files.

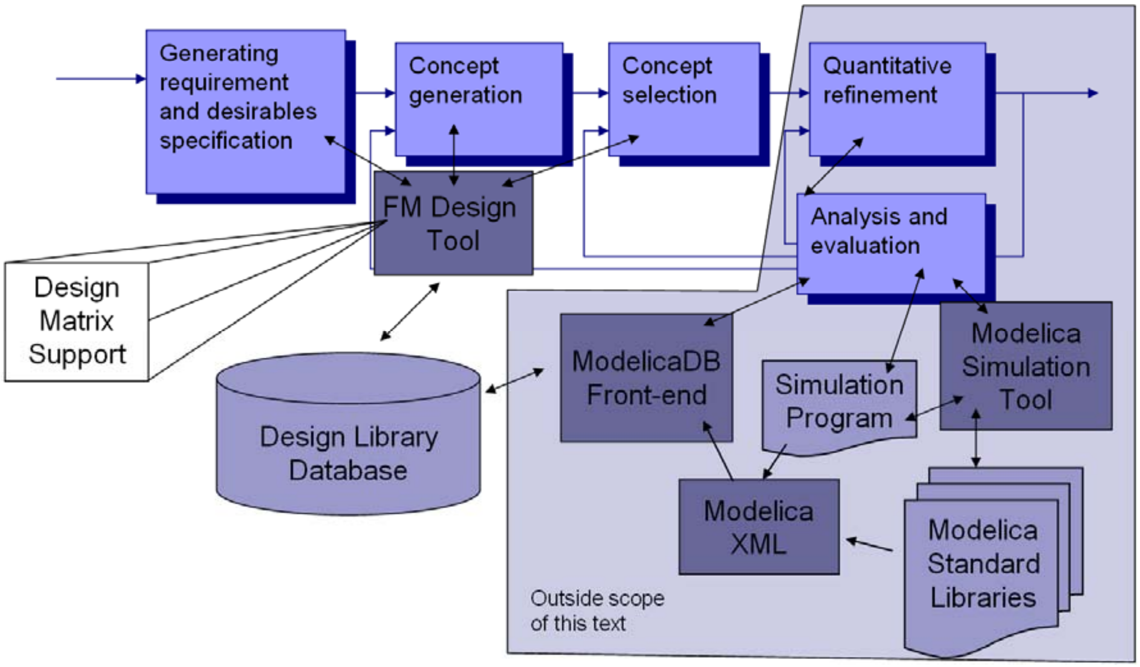


Figure 1. FMDesign in its context

FMDesign is a tool for designing product concepts with the aid of integrated requirement trees, function-means trees, product concept trees, and implementation trees. An implementation tree specifies the product structure and its interacting components of a particular product concept on an abstraction level that is detailed enough so the structure, system design, geometrical shape information etc can be used by external parameter calculation tools. There the models are refined such that calculations can be made for design variables like geometry, weight, drag, inertia, centre of gravity etc and feed back as implementation attributes on the objects in the implementation tree.

Once enough details about the static structures of a product concept have been specified, the implementation tree can be annotated with behavioural models selected from external simulation libraries, and its implementation attributes supplied as parameters to generated simulation models which are executed in external simulation tools. Current integrations with simulation tools use the Modelica language semantics as base for simulation model exchange formats [9]-[15].

Relevant processing results from external tools are feed back into the product model in FMDesign, where checks how well its different product concepts fulfil requirements can be made. Relevant calculation and simulation results stored in the product models provide convenient data sets to decision support tools for the continued product concept generation and selection process.

FMDesign stores product models in a design library database. This library includes products under development, and third party products that are used as sub-systems and components in the currently designed products. The products are organized into one or several parallel configurable product categorization hierarchies, so the designer quickly can find relevant supplier products and include them as objects in implementation trees.

Starting from requirements for a product the designer uses FMDesign for modelling alternative product concepts. The knowledge base for product design is organized into function-means trees. A function in the product can be realized by alternative means. A product concept is a set of means that document selected means for implementing the functions in a product concept. Example of a function is "Actuator Power Supply", with alternative means like "Hydraulic Power Supply" and "Electrical Power Supply". Means must be implemented by (physical) components arranged in the bill-of-material like implementation tree of implementation objects.

A means object and its corresponding implementation objects roughly represent the same thing, but at different levels of abstraction and detail. Implementation objects may represent existing component products on the market or manufactured components. Some (physical components) may implement several means, like an aircraft wing that creates lift and stores fuel. Implementation objects only carry implementation attributes that are important for the product concept design, and provide references to more detailed design information like CAD-drawings, simulation models etc as URLs, file names or object identifiers in databases.

Figure 2. RAVEN – Small Business Jet

As an example an aircraft system design is chosen. This represents a system of high complexity, a high degree of interconnectivity, it involves many domains, and have one very formal subset of requirements in the form of the Federal Aviation Regulations FAR [16]. These are available on-line and can be directly linked into the product model presented here. The fictious example here represents a business jet.

Overview of FMDesign User Interface

This section gives a walkthrough of FMDesign functionality as reflected in its user interface. Appendix A shows how the information is structured within the tool. Figure 3 shows the design library window which appears when the FMDesign front-end tool is launched at the user's workstation.

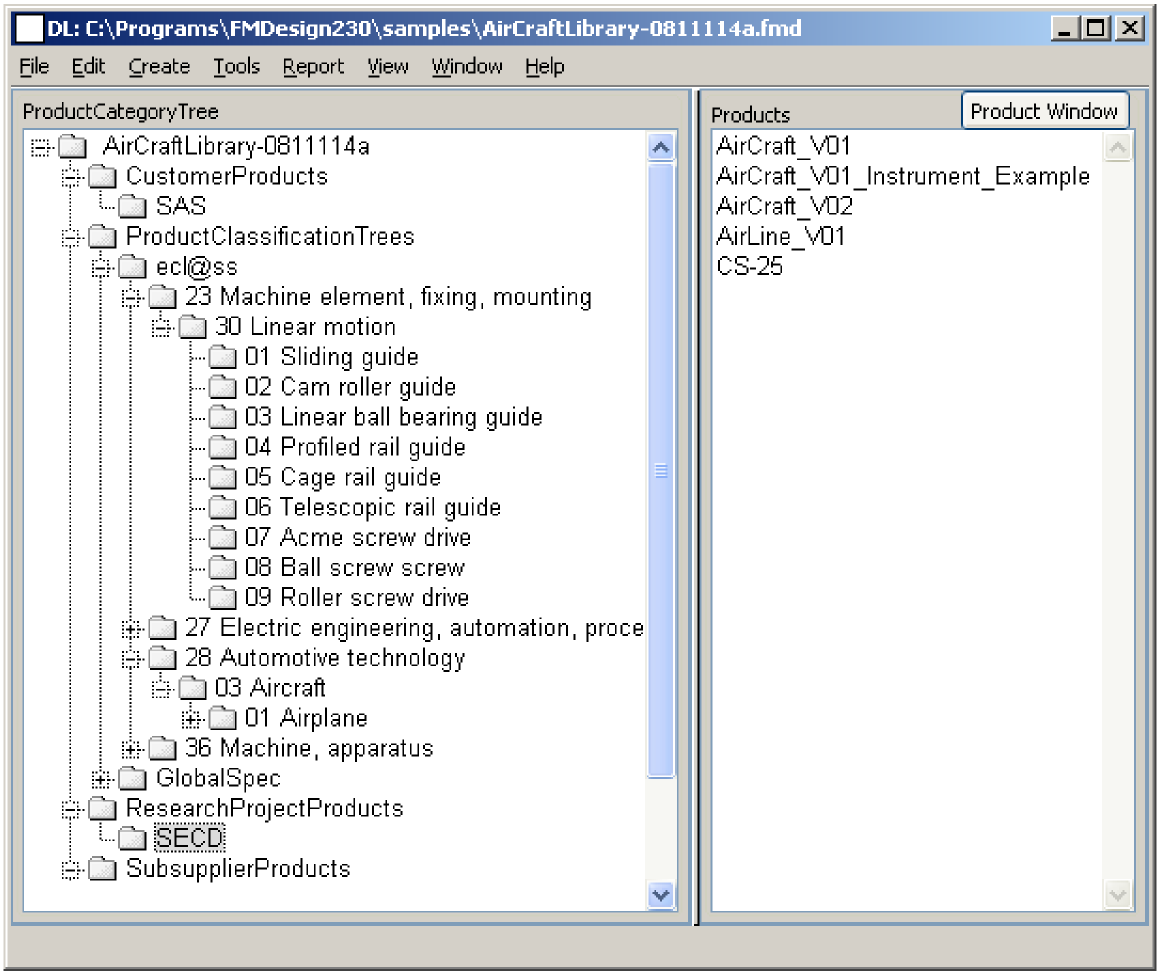
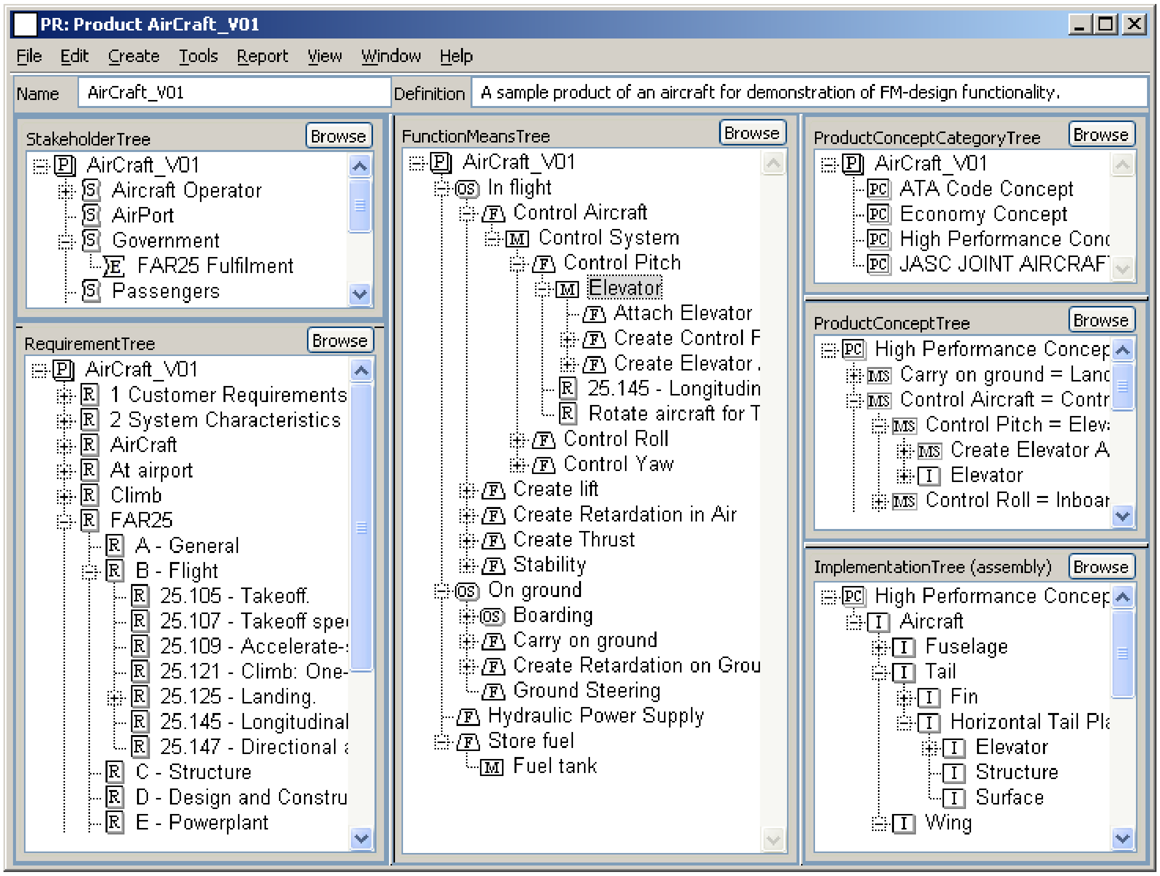


Figure 3. FMDesign Design Library Window

The product category tree allows the user to quickly narrow the list of all products displayed in the Products list-box to the ones in the category searched for. Products can quickly be moved from one product category to another with the drag-and-drop interface. Selecting a product, and pressing the "Product Window" button opens the Product Window in Figure 4.



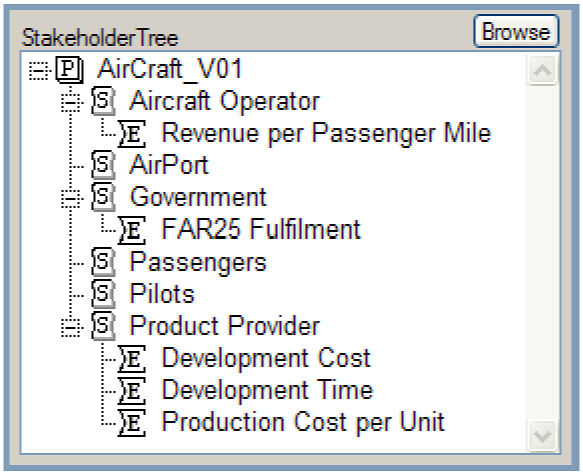
Name AirCraft_V01 Definition A sample product of an aircraft for demonstration of FM-design functionality.

Figure 4. Product Window

The Product Window provides access to the trees that structure the product model and menus with tools for checking the model with design rules, generating reports etc. Trees that are deeper than a few levels can be edited in separate browser windows. These are launched by selecting an appropriate object and opening a tree browser window that has the selected object as root with a press of a “Browse” button.
In the following the different product model trees are explained in more detail.

Stakeholder Tree

A stakeholder is an organization or category of persons with a certain interest in the product. Stakeholders can be users, operators, owners/buyers/customers, engineers, manufacturers, management etc.



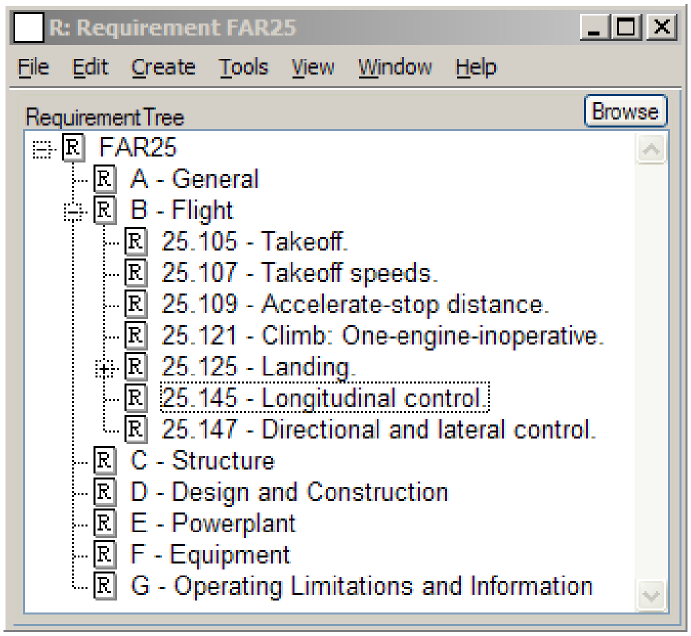
- P AirCraft_V01
  - S Aircraft Operator
    - E Revenue per Passenger Mile
  - S AirPort
  - S Government
    - E FAR25 Fulfilment
  - S Passengers
  - S Pilots
  - S Product Provider
    - E Development Cost
    - E Development Time
    - E Production Cost per Unit

Figure 5. Stakeholder Tree

The stakeholder tree organizes stakeholders into different categories. Each stakeholder may own effectiveness measures. Effectiveness measures establish the criteria by which alternative product concepts will be judged by its stakeholder. They provide guidance to the developers of structure and behavioural models on what is most important to the different stakeholders. See [5], Chapter 6.

Requirement Tree

The requirement tree structures the requirements into different topics, from overall non-functional requirements [6] down to optional requirements that are enforced when a particular means is selected for a function. Figure 6 shows a requirement browser opened on a copy of Part 25 within the Federal Aviation Regulations [16].



- R FAR25
  - R A - General
  - R B - Flight
    - R 25.105 - Takeoff.
    - R 25.107 - Takeoff speeds.
    - R 25.109 - Accelerate-stop distance.
    - R 25.121 - Climb: One-engine-inoperative.
    - R 25.125 - Landing.
    - R 25.145 - Longitudinal control.
    - R 25.147 - Directional and lateral control.
  - R C - Structure
  - R D - Design and Construction
  - R E - Powerplant
  - R F - Equipment
  - R G - Operating Limitations and Information

Figure 6. Requirement Tree

Requirements can be linked with drag-and-drop to other design objects like functions and means that are affected.

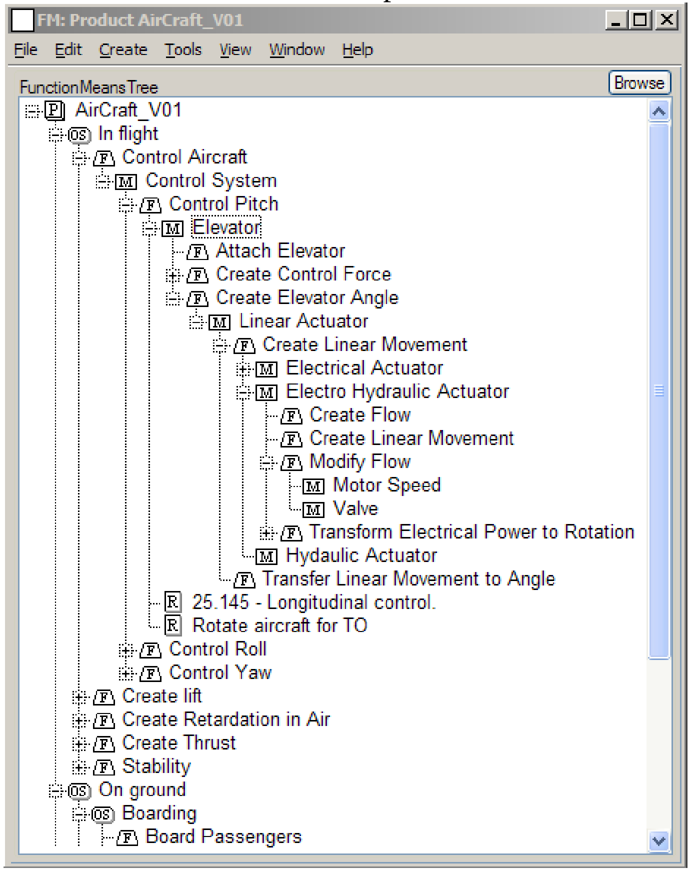
Numerical requirements like operation range, max speed etc can optionally be specified as a probabilistic distribution which describes the range in which the final design is desired to perform. This facility efficiently tells the designers what room they have for trade-offs.

Function Means Tree

The function-means tree serves as a tool for creative layout and later documentation and evaluation of the high-level design space considered for the product. See Figure 7.

It works as an ordinary function means tree, except for the inclusion of operation states. These specifies a certain state of operation for the product, e.g. for an aircraft, “In flight”, “On Ground” etc. Each operation state only groups the functions that are necessary for the product to perform in that operation state. The same function can be grouped under several different operation states. Dividing the products mission cycles into operation states gives the designer focus on what functions that are necessary in different states, and what requirements that apply to these functions in a particular state.

A function can have design variables like outputForce, outputTorque etc. These can have an assigned value, optional probabilistic distribution, and a variable type that tells if this is a variable that is controllable by the designer or not, or a value calculated by an external program that takes other variables as input.



```
P AirCraft_V01
  OS In flight
    F Control Aircraft
      M Control System
        F Control Pitch
          M Elevator
            F Attach Elevator
            F Create Control Force
            F Create Elevator Angle
              M Linear Actuator
                F Create Linear Movement
                  M Electrical Actuator
                  M Electro Hydraulic Actuator
                    F Create Flow
                    F Create Linear Movement
                    F Modify Flow
                      M Motor Speed
                      M Valve
                    F Transform Electrical Power to Rotation
                  M Hydaulic Actuator
                F Transfer Linear Movement to Angle
          R 25.145 - Longitudinal control.
          R Rotate aircraft for TO
        F Control Roll
        F Control Yaw
    F Create lift
    F Create Retardation in Air
    F Create Thrust
    F Stability
  OS On ground
    OS Boarding
      F Board Passengers
```

Figure 7. Function Means Tree

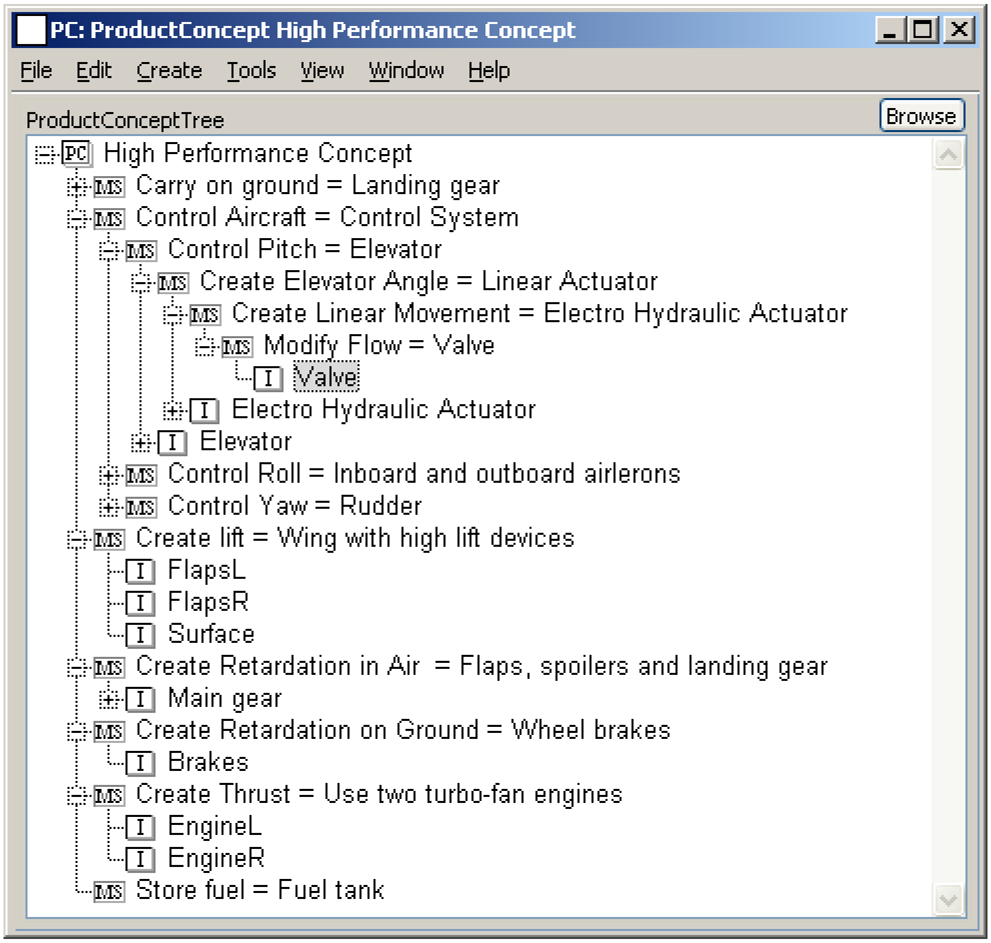
A means can also have design variables that specify the ones in its function more precisely, and perhaps additional variables that are specific to that means.

A benefit with function means trees compared with less formal traditional design methods, is that much of the knowledge and research that is documented in it is often lost or hard to find after the designers have decided on a particular product concept and proceeded with its detailed design. When the design of the next release of the product is started, some means that were not selected in the last product design phase may be interesting again, since technology and requirements may have changed. Much research and exploration may have to be done again, perhaps by new people if the market cycle for a product revision is long.

Product Concept Tree

A product concept tree documents which means that is selected for each used function in the function means tree. The product concept tree fixes the design choices such that a product implementation can be modelled and evaluated.

A MeansSelection is an object that documents what Means that was selected for a particular function. It may be connected to one or several implementation objects in the implementation tree. In some cases, the design variable specifications for a Function-Means pair must be further customized when it is selected in a particular product concept. The design variable value or associated probabilistic distribution may have to be changed. This can be done by ValueSelection objects that override specific design variables for a particular means selection.


PC: ProductConcept High Performance Concept
File Edit Create Tools View Window Help
ProductConceptTree
Browse

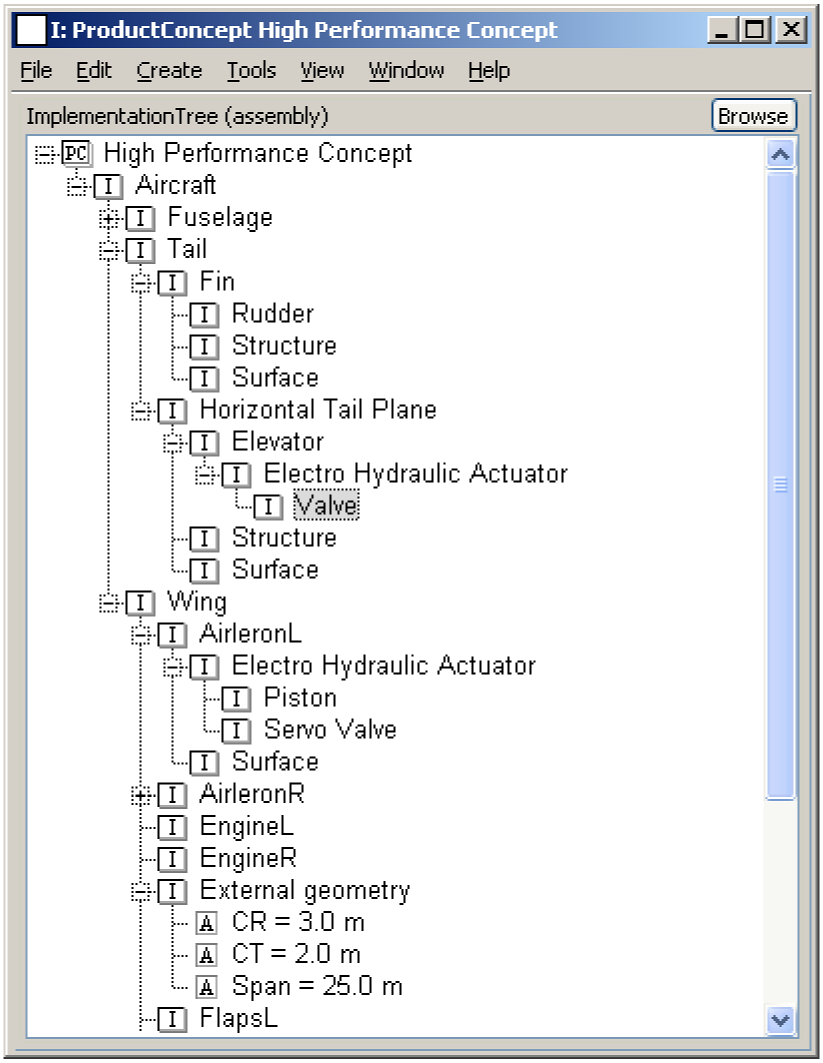
- PC High Performance Concept
  - MS Carry on ground = Landing gear
  - MS Control Aircraft = Control System
    - MS Control Pitch = Elevator
      - MS Create Elevator Angle = Linear Actuator
        - MS Create Linear Movement = Electro Hydraulic Actuator
          - MS Modify Flow = Valve
            - I Valve
        - I Electro Hydraulic Actuator
      - I Elevator
    - MS Control Roll = Inboard and outboard airlerons
    - MS Control Yaw = Rudder
  - MS Create lift = Wing with high lift devices
    - I FlapsL
    - I FlapsR
    - I Surface
  - MS Create Retardation in Air = Flaps, spoilers and landing gear
    - I Main gear
  - MS Create Retardation on Ground = Wheel brakes
    - I Brakes
  - MS Create Thrust = Use two turbo-fan engines
    - I EngineL
    - I EngineR
  - MS Store fuel = Fuel tank

Figure 8. Product Concept Tree

The same function and means can be instantiated many times in a product concept. For example the aircraft may have several engines of the same type.

Implementation Tree

The implementation tree displays and provides functionality for editing one of many configurable implementation trees for a particular product concept.


I: ProductConcept High Performance Concept
File
Edit
Create
Tools
View
Window
Help
ImplementationTree (assembly)
Browse

```
PC High Performance Concept
  I Aircraft
    I Fuselage
    I Tail
      I Fin
        I Rudder
        I Structure
        I Surface
      I Horizontal Tail Plane
        I Elevator
          I Electro Hydraulic Actuator
            I Valve
        I Structure
        I Surface
    I Wing
      I AirleronL
        I Electro Hydraulic Actuator
          I Piston
          I Servo Valve
        I Surface
      I AirleronR
      I EngineL
      I EngineR
      I External geometry
        A CR = 3.0 m
        A CT = 2.0 m
        A Span = 25.0 m
      I FlapsL
```

Figure 9. Implementation Tree

These implementation trees organize the implementation objects that represent and refer to more detailed models of physical objects, functional models, simulation models, geometrical layout models etc, and organize them into trees that are useful for interfacing with tools later in the product development process.

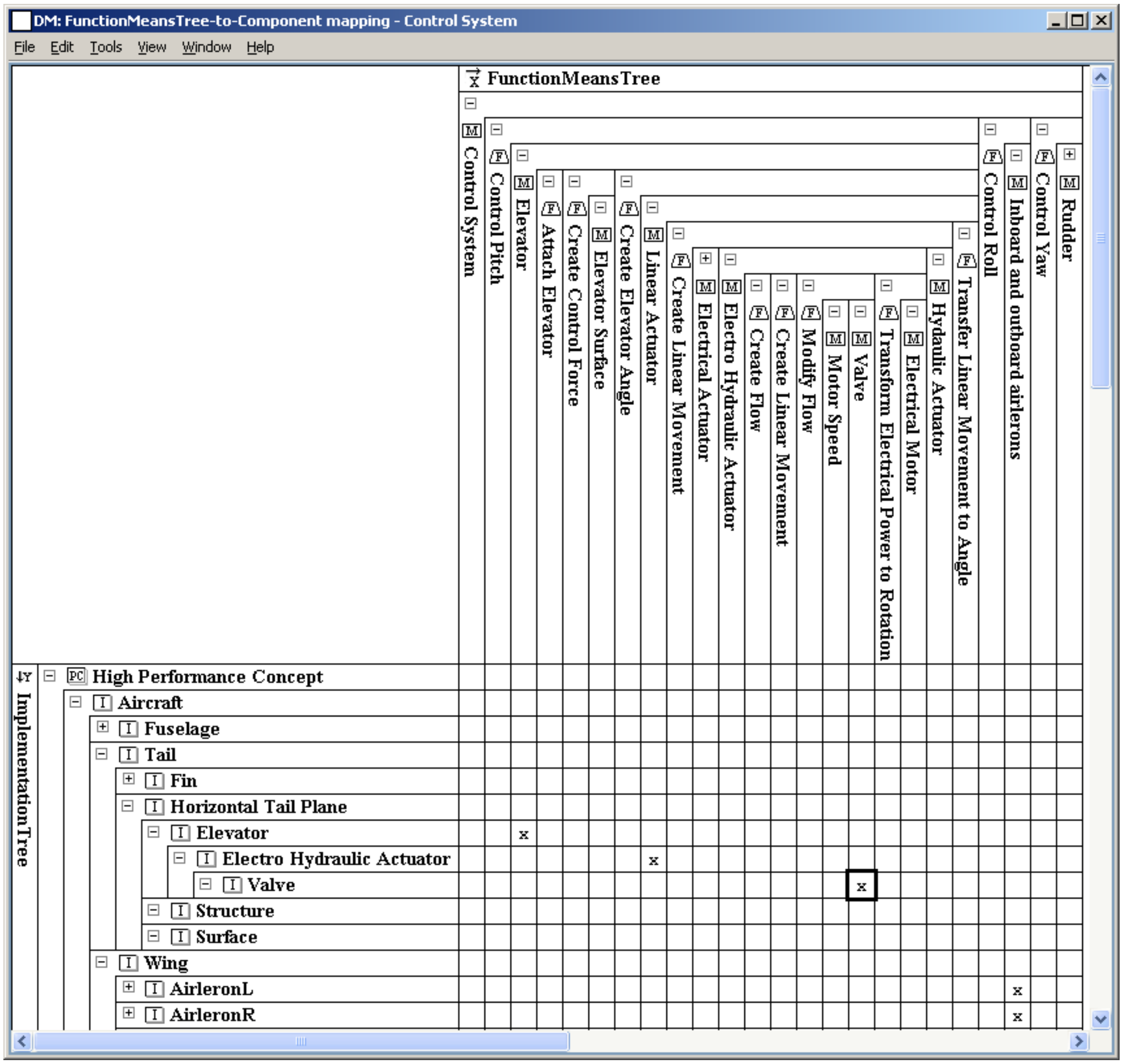
Probabilistic Variables

All variables that may have a numeric value like Requirements, DesignVariables, and ImplementationAttributes are probabilistic variables that can have an optional probabilistic distribution. Currently normal-, uniform-, interval- value-, and a special custom distribution with a configurable probability density function are supported.

Probabilistic variables may optionally have an assigned designSpaceMax, and designSpaceMin value, which let the designer set fixed limits for design space exploration by external optimization programs.

Design Matrixes

Design matrixes provide views on the trees and how individual objects in different trees are related to each other.



	Control System	Control Pitch	Elevator	Attach Elevator	Create Control Force	Elevator Surface	Create Elevator Angle	Linear Actuator	Create Linear Movement	Electrical Actuator	Electro Hydraulic Actuator	Create Flow	Create Linear Movement	Modify Flow	Motor Speed	Valve	Transform Electrical Power to Rotation	Electrical Motor	Hydaulic Actuator	Transfer Linear Movement to Angle	Control Roll	Inboard and outboard airlerons	Control Yaw	Rudder
High Performance Concept																								
Aircraft																								
Fuselage																								
Tail																								
Fin																								
Horizontal Tail Plane																								
Elevator			x																					
Electro Hydraulic Actuator								x																
Valve																x								
Structure																								
Surface																								
Wing																								
AirleronL																						x		
AirleronR																						x		

Figure 10. Design Matrix

Figure 10 shows a function means tree to implementation tree mapping. Other design matrixes show:

Stakeholder requirements to effectiveness measure mapping, where the requirements specify target values for the effectiveness measures relevant for different stakeholders.

Stakeholder tree to main function tree mapping. A main function tree only show the root-most functions of the product and not any sub functions of any particular means. The values in this matrix show the requirement priority a certain stake holder gives to a particular function, and can guide the design project resource allocation and scheduling of implementation and evaluation efforts for different functions.
Connection matrix, which is an implementation tree to implementation tree mapping, where the values in the matrix specify if two implementation objects are connected. This information may guide the initial geometrical layout of the product.

Summary

FMDesign is a tool for designing product concepts with the aid of integrated requirement trees, function-means trees, product concept trees, and implementation trees.
Design matrixes provide views on the trees and how individual objects in different trees are related to each other.

References

[1] Mogens Myrup Andreasen. "Machine Design Methods Based on a Systematic Approach (Syntesemetoder pa systemgrundlag)", Lund Technical University, Lund, Sweden, 1980

[2] Bracewell, R.H. and Sharpe, J.E.E., "Functional descriptions used in computer support for qualitative scheme generation - "Schemebuilder"", AI EDAM Journal - Special Issue: Representing Functionality in Design, v10, n4, pp333-346, 1996

[3] Bracewell, R.H. and Sharpe, J.E.E., "The use of Bond Graph Methodology in an Integrated Interdisciplinary Design System", in Joint Hungarian-British Mechatronics Conference, Budapest, Computational Mechanics Publications, pp595-600, 1994

[4] Sören Wilhelms, "Reuse of principle solution elements in conceptual design : an information model for concepts in systematic design", 2003, ISBN : 91-7373-746-1

- Krister Sutinen, "Supporting requirements management by requirements driven product modelling", Doktorsavhandlingar vid Chalmers tekniska högskola. New series nr 2065, Sweden, 2004

[5] David W Oliver et. al "Engineering Complex Systems with models and objects", McGraw-Hill, 1997, ISBN 0-07-048188-1

[6] Andreas Borg, “Contribution to Management and Validation of Non-Functional Requirements”, Linköping University, 2004

[7] INCOSE, *International Council on System Engineering*, http://www.incose.org

[8] Björn Johansson, Petter Krus, "A WEB SERVICE APPROACH FOR MODEL INTEGRATION IN COMPUTATIONAL DESIGN", Proceedings of DETC’03, ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference Chicago, Illinois USA, September 2-6, 2003, DETC2003/CIE-48196

[9] Modelica Association, "A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification version 2.1", Modelica Association, 2004

[10] Modelica Association, http://www.modelica.org

[11] Modelica Association, "Modelica Libraries", http://www.modelica.org/library

[12] Peter Fritzson, "Principles of Object-Oriented Modeling and Simulation with Modelica 2.1", IEEE Press, 2004, ISBN 0-471-47163-1

[13] Olof Johansson, Adrian Pop, Peter Fritzson, "ModelicaDB - A Tool for Searching, Analysing, Cross-referencing and Checking of Modelica Libraries", in Proc of 4th International Modelica Conference, 8-10 March, 2005, Hamburg, Germany, http://www.modelica.org/events/Conference2005

[14] Olof Johansson, Adrian Pop, Peter Fritzson, "A Functionality Coverage Analysis of Industrially used Ontology Languages", in Model Driven Architecture: Foundations and Applications (MDAFA), 2004, 10-11 June, 2004, Linköping, Sweden.

[15] Adrian Pop, Olof Johansson, Peter Fritzson, "An integrated framework for model-driven product design and development using Modelica", in Proceedings of the 45th Conference on Simulation and Modeling (SIMS), 23-24 September 2004, Copenhagen.

[16] Risingup Aviation, "Federal Aviation Regulations Part 25: Airworthiness standards: Transport category airplanes", Web resource for research: http://www.risingup.com/fars/info/25-index.shtml

[17] RosettaNet, http://www.rosettanet.org

[18] RosettaNet, "RosettaNet Technical Dictionary", http://www.rosettanet.org/technicaldictionary

[19] Word Wide Web Consortium (W3C), "HyperText Markup Language (HTML) Home Page", http://www.w3.org/MarkUp/

[20] Word Wide Web Consortium (W3C), "XML Schema", http://www.w3.org/XML/Schema

[21] Word Wide Web Consortium (W3C), "Extensible Markup Language (XML)", http://www.w3.org/XML/

[22] OMG, "Unified Modeling Language (UML) Resource Page", http://www.omg.org/uml

[23] OMG, "XML Metadata Interchange (XMI)", http://www.omg.org/technology/documents/formal/xmi.htm

[24] Martin Fowler, "UML Distilled, 3rd Edition: A Brief Guide to the Standard Object Modeling Language", http://www.martinfowler.com/books.html#uml

[25] IEC, "IEC 61360 Standard data element types with associated classification scheme for electric components", http://webstore.iec.ch for a fee

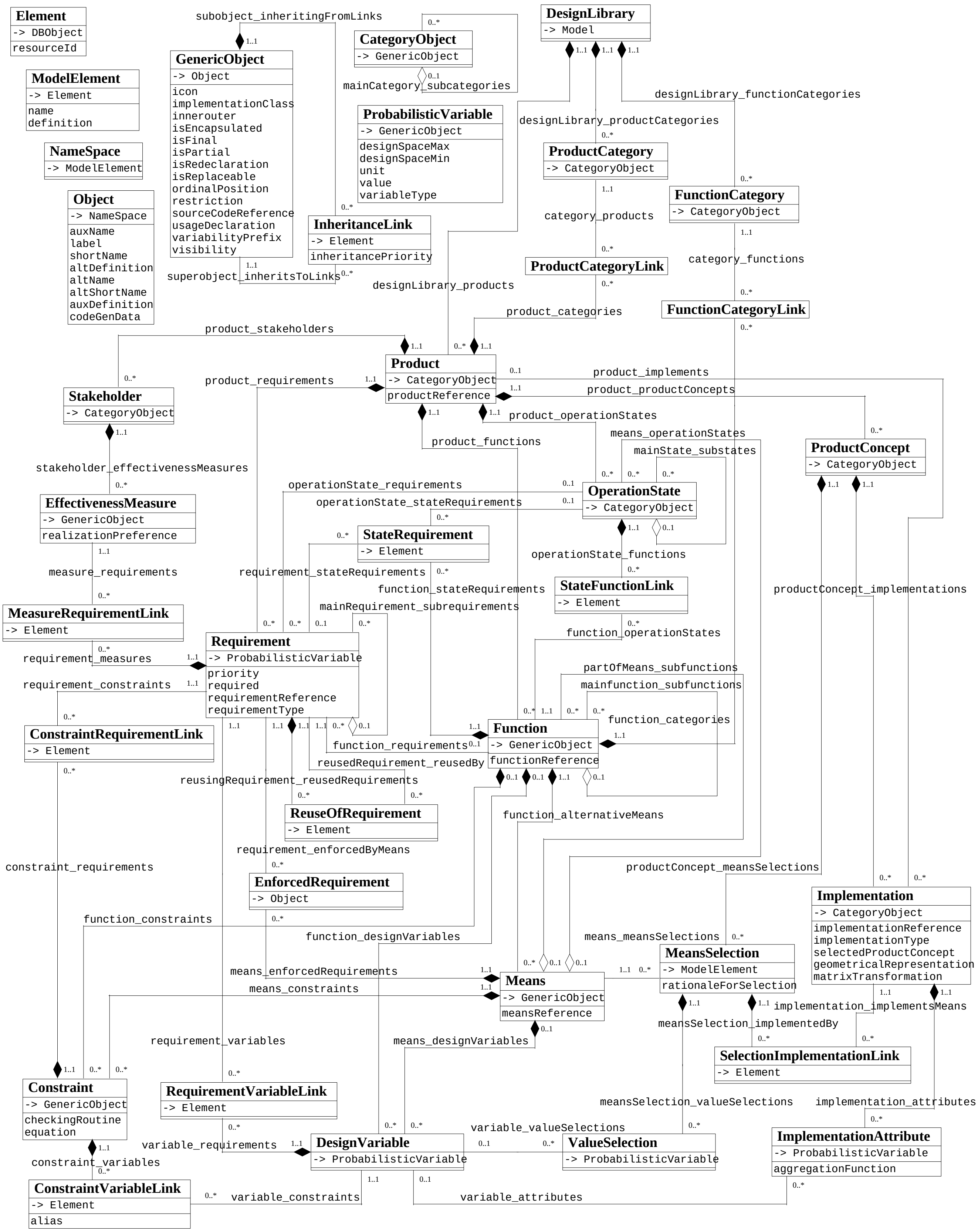
[26] ISO/IEC, "ISO/IEC 9075 - Information technology -- Database languages -- SQL", http://www.iso.org for a fee

[27] ISO, "ISO 31 - Quantities and Units", http://www.iso.org for a fee

Appendix A. FMDesign 2.0 UML class diagram

Descriptions of the notation of UML class diagrams can be found through [22]. See [24] for an introduction to UML. In Appendix A, classes have the class name in their first box. Their second box contains the superclass’s name preceded by a “->”. All attributes and relationship specified in the superclass are inherited by the class. The third box within a class contains a list of attribute names.

Relationships between classes have cardinality constraints telling how many instances that must participate in the relationship on each side specified as intervals [<min>..<max>]. A ‘*’ as <max> denotes infinity. A black diamond on the owner class side of the relationship denotes that instances on the other side belong to that class. An unfilled diamond denotes that the instances on the other side are aggregated by a class, but are “physically” owned by another relationship.



Date Printed
2008-01-09 16:57:17

	Domain Model	Filename	Date Added	Date Modified	#Cls	#Rel	#Att
DM.	FMDesign23l	FMDesign23l.odm	2003-04-24 20:01:23	2008-01-09 16:55:50	123	153	233
	Information model of the function-means tree centered product concept design system, extended with Generic Object Inheritance.						
Dgm	A) Function Means Application		2006-02-13 05:19:27	2008-01-09 16:55:50	36	55	-
	Appendix A: UML class diagram showing the architecture of the FMDesign systems engineering application.						