CONCEPTUAL DESIGN USING GENERIC OBJECT INHERITANCE

http://www.iei.liu.se/machine

Dr. Olof Johansson, Henric Andersson,
Prof. Petter Krus

Division of Machine Design
Department of Management and Engineering (IEI)
Linköping University, Sweden
Conceptual Design with Function Means Trees for Layout of Product Concept Design Space

Main Function

Means (product)

Function 1

Means

Function

Means

Function

Means

Function 2

Means

Function

Means

MODPROD-2009 2
Product Model Example
(Small Business Jet Example)
Generic Object Inheritance

Example: Reuse of Function Means Trees

Reuse of Function A in several places, including any Means within their "part-of" structures.

Inheritance Link: ←→

Grey objects are inherited virtual objects, that are automatically persistently instansiated when changes are made.
Challenges of Large Integrated Object-Oriented Product Models for Product Concept Design

- Efficient Reuse of Models
- Change Management
FMDesign Context and Architecture

Tool supported systems engineering processes

- Generating requirement and desirables specification
- Concept generation
- Concept selection
- Quantitative refinement
- Analysis and evaluation

Design Matrix Support

Design Library Database

FM Design Tool

ModelicaDB Front-end

Simulation Program

Modelica XML

Modelica Standard Libraries

Outside scope of this talk

MODPROD-2009
Organization of SE objects in FMDesign (Small Business Jet Example)
Inheritance of Product Model with Local Changes in Inherited Content

Inherited objects from AirCraft_V01 displayed in grey
Change Management
with Visualization of Changes

Green = Created object, Red = Deleted object,
Blue = Updated object, Light Blue = Contains Updated parts
Grey = Inherited unchanged object
New Results

• Automatic inheritance of object’s attributes, part-of structures and referencing relationships.

• Easy to directly modify inherited content

• Use of new inheritance mechanism for Change Management
ACKNOWLEDGMENTS

This work was supported by

– Swedish Governmental Agency for Innovation Systems (VINNOVA) and SAAB in the National Aviation Engineering Research Programme (NFFP4)
  • Project Modeling Techniques for Avionics Design
Backup – Implementation Facts

- Tested with benchmark trees containing 1000 instantiated objects
  - where the deepest part-of structure contained 10 hierarchical levels
  - deepest inheritance structure 10 inheritance levels

- This benchmark model contained about 10000 virtual objects when it was completely expanded
Backup – Implementation Facts

- Implementation of framework and source code generators for GOI functionality was rather complex ~ ½ man year
  - Plenty of behaviour cases for different states (updated, deleted etc) in a large number of object manipulation operations.
  - More than 20 000 lines of code in declarative source code generator scripts.
  - Generated ~ 430 000 lines of code from the FMDesign UML model of ~120 classes, ~150 relationships and ~230 attributes.
  - Large software framework to implement core database operations and graphical user interface.
Backup

• Configurable Design Matrixes
Showing Relationships amongst SE objects with Design Matrixes
# Design Matrix

## Software Components

### X-Axis
- **Axis object A**
  - Implementation Object
  - Represented by an Axis Object Editor

### Y-Axis
- **Axis object B**
  - Requirement Object
  - Represented by an Axis Object Editor

### Crossing Object C
- Selection Implementation Link
  - Represented by a Crossing Object Editor

<table>
<thead>
<tr>
<th>Requirement Object Editor</th>
<th>Axis Object Editor</th>
<th>Crossing Object Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Axis</td>
<td>Y-Axis</td>
<td>Crossing Object C</td>
</tr>
<tr>
<td>Axis object A</td>
<td>Axis object B</td>
<td>Selection Implementation Link</td>
</tr>
</tbody>
</table>
Simplified FMDesign
Information Model (UML Class Diagram)
Design Matrix Configurator
Specifying Access Paths for Crossing Relationships

MODPROD-2009
Access Path, declaring how to build the contents of the Crossing Index B -> C for A

Class of Axis object B

Class of Crossed Object C

Class of Axis object A

Path Components
Not visible in the design matrix
Features of Configurable Design Matrixes

- **Layout and information content can be configured in any way** permitted by the information model of the engineering application.
- The design matrix is **just a view of the underlying product model**
  - No manual updating of the matrix is needed when objects and structure are changed in the product model.
- **Any related application trees can be on the X- and Y-axis.**
- Objects in the **trees** on the X- and Y-axes are **dynamically expandable to different levels of detail.**
- The **relations** between objects on the axes which are made visible in the cells of the matrix, **can be created and deleted.**
- **Attribute values on a relation object** represented in a crossing between axis objects **can be displayed and edited.**
- With efficient underlying datastructures, the **design matrixes can scale with interactive response times** on sizes of realistic system engineering product models.
Results and Experiences

• We have developed a design matrix specification language and verified that it works with an implementation in FMDesign.

• The design matrix on slide 6 in the actual business jet example has more than 50 objects on each axis, more than 4000 crossing objects but is still printable on an A2 sheet.

• A user can specify a design matrix in 10-20 minutes using the configurator

• Practical experiments with a generated requirement tree containing 250 requirements on the Y-axis, showed that we need faster underlying data structures.

• Limitation of workstation screen sizes requires support for
  – fast interactive scrolling and panning over the matrix area,
  – while still seeing the relevant sections of the trees on the axes.
Conclusions

• Configurable design matrixes seem to be a very promising tool to aid systems engineering of large and complex products.

• The formalization of the software model of the systems engineering application has made it possible to build a generic configurable design matrix framework that automates the software implementation of design matrixes.

• Implementation of a new design matrix view on a product model can now be completed in less than an hour by a user, compared to previously necessary days or weeks by a highly skilled graphics software developer.

• The presented formalization is conformant with the UML standard, and can thus be used by systems engineering tool developers that model their software designs in UML, to extend their tools with a configurable design matrix framework.
Future Work

• Improve the design of the interactive user interface for navigating in large design matrixes.
• Improve performance
  – using faster underlying data structures
  – When performance is OK, we will proceed with more design work oriented experiments with imported systems engineering designs from large industrial projects.
• One useful feature that can be added to the design matrix is sensitivity analysis.
  – In addition to indicate that there is a connection between, for example a design parameter and a system characteristics, its relative strength can be displayed.
  – This can also be applied to hierarchical structure to display the impact of a whole subsystem on a system characteristics.
  – In addition information regarding estimated uncertainties in parameters and models can be included in order to manage the confidence of the product model.
ACKNOWLEDGMENTS

This work was supported by

– Swedish Foundation for Strategic Research,
  • ProViking project Systems Engineering and Computational Design (SECD)
– Swedish Governmental Agency for Innovation Systems (VINNOVA)
  • Project Semantic Web for Products (SWEBPROD)

Reference on Systems Engineering Process
inspiring FMDesign: