Model-Based Conceptual Design

Prof. Dr. Kristina Shea
Virtual Product Development Group
Institute for Product Development
Technische Universität München
http://www.pe.mw.tum.de
Outline

• Motivation
• Graph-Based Product Modeling
• Model-Based Libraries for Conceptual Design
• Conceptual Design Automation
• Summary and Future Outlook
Significance of Conceptual Design

- The least amount of quantitative information is known and the most important decisions are being made.
- The possibility of influencing costs (80%) is the highest!
- The greatest potential exists for innovation.

Systematic Product Development

- Clarify and define task
- Determine functions and their structures
- Search for working principles and their combinations

Phase I
Conceptual Design

- Divide into realizable modules
- Develop layout of key modules
- Complete overall layout

Phase II
Embodiment design

- Prepare production and operating instructions

Phase III
Detail design

Reference: [VDI 2221, 1993]
Example: New Vehicle Functions and Architectures

New functions
• Engine Start-Stop
• Improved exhaust gas cleaning (SCR)
• Active comfort systems
• Active safety systems
• Autonomous, strategic use of energy
• Grid buffer with electric cars and plug-in hybrids

New architectures
• Lightweight architectures
• Urban mobility vehicle
• Hybrid cars
• Electric cars

© GM Corp.
http://media.gm.com/ProjectPUMA/
Project P.U.M.A.
Model-Based Systems Engineering

• **Formal modeling** approach to support system requirements, design, analysis, verification, and validation activities throughout the product lifecycle

• Move from producing and controlling **documentation** to producing and controlling a coherent **model** of the system

• A **system model** specifies and integrates subsystems and components in multiple domains

• **Syntax and semantics** of the model impose rules that constrain which relationships can exist in the model

• Enhances product specification, integration of disciplines and reuse of models
Unifying MBSE and Systematic Product Development

**MBSE**
- Specification
- Functions
- Components

**Systematic PD**
- Specification
- Functions
- Effects
- Working principles
- Components

**Integration**
- Generic model-based libraries
- Integration of behavior level
- Domain Integration
- System Design Methods
- Link to Simulation
- Link to CAD
- Link to PDM

**FBS modeling**
**Systematic Product Development and Design Catalogs**

Paper-based design methods and catalogs:

## Paper-Based vs. Model-Based Conceptual Design

<table>
<thead>
<tr>
<th>Paper-Based</th>
<th>Model-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Taxonomy of basic functions and flows</strong> (NIST, Roth, Koller)</td>
<td><strong>Model-based library of functions and flows</strong></td>
</tr>
<tr>
<td>Manual search for physical effects in mapping matrix</td>
<td>Computational search for valid physical effects</td>
</tr>
<tr>
<td>Selection and adaption to create working principle</td>
<td>Selection and computational search for valid working principle</td>
</tr>
<tr>
<td>Preliminary dimensioning of parameters</td>
<td>Reasoning about design parameters and constraints</td>
</tr>
<tr>
<td>Manual selection or development of components</td>
<td>Computational search for valid components</td>
</tr>
<tr>
<td><strong>Documented in standard office software, e.g. spreadsheets</strong></td>
<td>Encoding and capture of design logic</td>
</tr>
</tbody>
</table>

**Model-based design specification**
Verb-noun pairs describing functions (e.g. Functional Basis).

Idealized physical effects (and working principles) that realize the functions (e.g. Roth or Koller).

Embodiment of required behaviors through components (artifacts).
Model-Based Design Libraries

**Goal:** For each FBS layer, define generic models and build libraries for conceptual design.

**Why?**
- Define a common practice
- Define the syntax and semantics for each FBS layer
- Avoid ambiguity of model elements and constrain use
- Represent taxonomies, e.g. functions and flows
- Speed-up modeling through re-use
- Support development of alternative solutions

| 1D and 3D mechanics | analog and digital electrical circuits, electrical machines | heat transfer, fluid systems | cont., discrete, logical blocks, state machines |

Modelica Libraries

Open Cascade Primitives
Requirements for Model-Based Design Libraries

- Not coupled to one process or methodology
- Expandable and customizable
- Use in different modes:
  - Fully manual
  - Semi-automatic
  - Fully Automatic
- Multi-domain, e.g. mechatronics
- Provide transformations to:
  - CAD
  - constraint solving
  - simulation, e.g. Modelica
Why SysML?

• Supports the specification, design, analysis, and verification of systems
• Supports **multi-domain systems**, i.e. including hardware, software, data, processes, and people
• **Graphical** modeling language with a **semantic foundation**
• **Extendable**, e.g. through profiles
• **Domain independent** with growing interest and use in industry
• Upcoming specifications for **transformation to other models**, e.g. Modelica, GAMS and Matlab
Concept Modeling with SysML

1 Use Cases

2 Requirements

3 Function Model

4 Effects and Working Principle

5 Principle Solution

"A Computational Product Model for Conceptual Design Using SysML"

Functions

- **Function**: Operator that changes a technical input flow into an output flow
- **Model**: Activity
- **Interfaces**:.Typed pins represent the point of interaction of flow between functions that are connected by object flows

![Diagram of functions and interfaces](image)

**Flow**
- Material
- Signal
- Energy
- Mechanical
  - Rotational
  - Translational
- Electrical
  - DC
  - AC
- ... (more flows)

**Operator**
- Control Magnitude
- Regulate
- Increase
- Decrease
- Convert
- Provision
- Store
- Supply
- ... (more operators)

**Flow type**
- input: Energy
- output: Energy

**Pin name**
- input
- output

**Flow direction**
- from input to output
Working Principles and Effects

- **Effect**: describes a physical/chemical/biological phenomena by means of a formula and specifies the ratio of how the flow is changed
- **Model**: Block Interfaces: Ports / Connectors representing flows
- **Working principle**: abstract model of a component with a behavioral model behind it that relates effects to a first dimensioning of the device
- **Model**: Block Interfaces: Ports / Connectors representing flows
Components

- **Components**: physical entity or module in a product
- **Model**: block with attributes that allow evaluation, e.g. performance, cost and weight
- **Interfaces**: ports / connectors require additional types to flow types and additional information, e.g. orientations, positions, degree-of-freedom, etc.
Example: Luggage Compartment Cover

- Open
- Closed
- Disengaged
1. Define super-function with boundary crossing flows (black box)
Example: Luggage Compartment Cover

2. Create a new Diagram from superfunction to create decomposition
Example: Luggage Compartment Cover
Example: Luggage Compartment Cover

Modify library element for use:
- unique name
- documentation
Example: Luggage Compartment Cover

Modify Interfaces
- specify type form flow library
- display type
Example: Luggage Compartment Cover

Connect Interfaces
- context sensitive highlighting
Example: Luggage Compartment Cover

Completed function structure

- **Cover**: Supply
  - **Energy from Material**: Branch
    - **Mechanical Energy**: Store
    - **Retaining Element**: Support
      - **Translate Cover**: Channel
        - **Handle**: Connect
          - **Basic Flow**: Hand
Searching for Elements in Libraries

How to find a corresponding physical effect for a function?

1. Hard coded, 1:1 mapping of functions to physical effects

- Mapping is done within a library (decomposition) or among different libraries
- Relationships are defined for each possibility
- Gives a fast overview of possible mappings for selection
- Encapsulated mapping of corresponding pins/ports and parameters
Searching for Elements in Libraries

2. Port type matching:

- Works also with higher level functions $\rightarrow$ decomposition
- Requires a clear definition of pins, ports and matching constraints
- Discovers multiple possible mappings
- Finding solutions by evaluating two or more relations
- Requires user support for search process
Conceptual Design Automation

Process:

1. **Knowledge**
2. **Formalization**
3. **Formal language**
4. **Synthesis**
5. **Product concepts**
6. **Simulation & optimization**

- **Computationally describe** design spaces and languages
- **Reason** from fundamental knowledge and physics-based simulation
- **Generate** both known and new designs
- **Automate** processes (design or fabrication)
- **Search** for feasible, “good” solutions and **optimal** designs
- **Understand** complex solution spaces and complex constraints
- **Spark** creativity and innovation
Conceptual Design Automation

Challenges:

- Using existing knowledge
- Knowledge evolves
- Combinatorial approach is not feasible

Simulation & optimization
Graph Grammars and FBS

Vocabulary

Provide linear displacement

Rules

Provide linear displacement

Import energy

Convert energy to mec. transl. energy

Function

Behavior

Structure

Graph Grammar: Rule Set

Problem-specific rules

- Input
- Recuperate energy
- Convert
- Regulate
- Convert
- Store
- Input
- Recuperate energy
- Input
- Regulate
Full Hybrid Powertrain

Initial functional model:
Full Hybrid Powertrain

Resulting structure model:
Powertrain Architecture Synthesis

Design Language
- ICE vehicle
- Full hybrid
- Electric vehicle
- Micro hybrid
- Mild hybrid
- Plug-in hybrid

Over 200 hundred solutions generated!
Full hybrid powertrain (serial, through the road)

Resulting structure model:
Software Platform

GrGen.NET
Wrapping
Graph-transformation (via libGr)

Own software

Evaluation

Meta-model
Rule set
Rule sequences
## General Languages vs. Domain-Specific Languages

<table>
<thead>
<tr>
<th>e.g. SysML</th>
<th>e.g. graph re-writing (GrGen)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Defined syntax</td>
<td>+ User-defined syntax definition according to language requirements</td>
</tr>
<tr>
<td>- Learning the syntax</td>
<td>+ Deeper understanding of domain through language definition</td>
</tr>
<tr>
<td>- Complex constructs</td>
<td>+ Reduced language</td>
</tr>
<tr>
<td>- Hard to map domain concepts to language concepts</td>
<td>- Knowledge about language specification required (learning the meta-syntax)</td>
</tr>
<tr>
<td>+ Standardized tools and growing community</td>
<td>- Can be difficult to integrate and grow in industry</td>
</tr>
<tr>
<td>+ Customization and extension is possible</td>
<td></td>
</tr>
</tbody>
</table>
Summary and Future Outlook

Model-based library approach to support conceptual design:
• provides a general and standardized way for structuring knowledge
• standardize the way of using SysML for product specification
• support model re-use and faster modeling → more alternatives
• support evolving knowledge

Main issues:
• specification of connection types and SysML modeling approach
• methods for decomposition and mappings
• multi-domain models, e.g. mechatronic systems
• robust translation to other models, e.g. automated synthesis methods
• human factors: how can engineers best work with the approach?
Acknowledgements

Thanks to:

– Dipl.-Ing. Stefan Wölkl and Dipl.-Ing. Bergen Helms, Virtual Product Development Group, TUM, for their work on the preparation of this presentation.

– Prof. Chris Paredis, Georgia Institute of Technology, USA, for his many discussions on modeling in SysML and model-based libraries for concept design.
Contact Details

Prof. Dr. Kristina Shea  
Virtual Product Development Group  
Institute of Product Development  
Technical University of Munich  
Boltzmannstrasse 15  
85748 Garching

Phone: +49 (0)89 289 151 43  
Fax: +49 (0)89 289 151 44  
Email: kristina.shea@pe.mw.tum.de