Capturing Domain-Specific Knowledge for Design of Hydraulic Systems

Chris Paredis

Systems Realization Laboratory
Product and Systems Lifecycle Management Center
G.W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

www.srl.gatech.edu   www.pslm.gatech.edu
Systems Engineering: A Decision-Based Perspective

Decisions

- Portfolio Planning
- Concept Development
- Design
- Production & Testing
- Sales & Distribution
- Maintenance & Support

Modeling and Simulation Provides Information in Support of Decisions

Generic Decision Process

1. Generate Alternatives
2. Evaluate Alternatives
3. Select Alternative

- Information
- Knowledge

Evaluate Alternatives

Generate Alternatives

Select Alternative
Challenges in Systems Engineering

- Multiple integrated functions
- Multiple engineering disciplines
- Multiple stakeholders
- Globally distributed, heterogeneous design teams
- Complex, emergent system behavior
- Large quantities of design knowledge and information

→ Need Formal, Model-Based Approach
Model-based Systems Engineering (MBSE)

**MBSE: Model formally all aspects of a systems engineering problem**

- **Effective and Efficient Analysis of Alternatives**
  - Model from different perspectives
  - Model at different levels of abstraction
  - Variable-fidelity modeling
  - Model reuse & modularity

- **Effective Generation of Alternatives**
  - Graph transformations for generating plausible system architectures
  - Automated generation of system models

Chris Paredis  Systems Realization Laboratory
MBSE Example Problem: Hydraulic Systems

Given:
- Primary components
- Decision objectives / preferences

Find:
- Best system topology
- Best component parameters

Very large search and optimization problem
- Many competing objectives
- Many topologies
- Many component types/sizes
- Many control strategies

How to size and connect these?

How do we best capture and use the system design knowledge?
Approach: Most Knowledge Can Be Represented as Graphs or Graph Transformations

- Requirements & Objectives: SysML
- Topology Generation using Graph Transformations
- System Alternatives: MAsCoMs
- Model Composition using Graph Transformations
- System Behavior Models: SysML
- Model Translation using Graph Transformations
- Executable Simulations: Dymola
- Simulation Configuration using Graph Transformations
- Design Optimization: ModelCenter

Diagram showing system alternative: behavior model, simulation configuration, Dig Cycle, Traj, Arm Boom Swing, Bucket Traj, behavior model simulation configuration.
Capturing Knowledge about Fluid-Power Circuits

1. A Language for describing Fluid-Power circuits
   • Language is described by a meta-model
   • Valid circuits are represented as graphs

2. A Model Library with static knowledge
   • What are the available components?
   • What are their characteristics and behaviors?

3. A set of Model Transformations
   • Knowledge on how to combine components into circuit
   • Knowledge on how to generate analysis models from circuit descriptions

4. Language Mappings to/from other domains
   • Allows results to be viewed and edited (e.g. in SysML)
   • Allows circuits to be analyzed (e.g. in Dymola/Modelica)
Language for Fluid-Power Circuits

Note: ultimately the resulting circuits need to be represented in SysML
Concrete Syntax – Extending SysML

- Extend SysML using a Profile
Specify Knowledge in Domain-Specific Model

SysML Model in SysML tool

Mapping

Domain Specific Model

Use MOFLON to define meta model

Transformation

Use MOFLON to define graph transformations to perform these actions

New SysML Model in SysML tool or Model in other tool

Mapping

Transformed Domain Specific Model
Challenges

- Language to express the Problem
  - Should cover a *set of problems* that is relevant to the user
  - The broader the set, the more complex the solution space and the more difficult the process of solving the problem could become
  - Includes objectives, requirements, etc. → very broad
  - How to anticipate all the aspects a designer may care about?

- Language to express fluid-power circuits
  - Should include each fluid-power circuit that is *optimal* for some problem instance
  - Ideally, should not include any other circuits → in practice: many more
  - Is it possible to constrain the language based on problem characteristics?
Generating System Alternatives
Generative Grammar for Design Synthesis

- Graph Transformation rules to generate systems
- Generate random system alternatives by applying rules in randomized order
- Improve system alternatives through evolutionary search algorithms
General Synthesis Approach

- Capture connectivity information in graph transformation rules
- Capture available components in model library
- Control the order in which rules are applied using decision tree
Decision Tree of Generation Process

- Add Cylinder
  - Success (Probability = .7)
  - Failure

- Add Directional Valve
  - Success (Probability = .3)

- Add Pump
  - Success (Probability = .7)
  - Failure (Probability = .3)

- Add Tank
Putting it all together
Some Challenges

- Selecting components at random:
  - Instead of simply matching one instance, need to match one instance at random

- Rule set
  - Should cover the entire space of circuits
  - Randomness should be “uniform” across space
How to impose constraints in a generative grammar?

- We have only explored graph transformations…
- Could we accomplish the same using constraint-based meta-model defined in Alloy (or similar tool)?
- Which approach is most intuitive/convenient for domain experts?
- Which approach is best suited for automated (randomized) synthesis and incremental modification (as in optimization/search)?

Larger problems:

- Which knowledge is captured in synthesis model and which is left for analysis?
- How to work at different levels of abstraction?
  - E.g.: topology, sizing, control,…
- Is there a systematic process for capturing synthesis knowledge?
Generating System-Level Analysis Models
Systems Development: A Decision-Based Perspective

Decisions

Portfolio Planning → Concept Development → Design → Production & Testing → Sales & Distribution → Maintenance & Support

Modeling and Simulation Provides Information in Support of Decisions

Generic Decision Process

Generate Alternatives → Evaluate Alternatives → Select Alternative

Knowledge 

Information
Many different perspectives, levels of abstraction, formalisms

Hypothesis:
- One can improve the efficiency of design optimization methods by considering multiple levels of abstraction and accuracy.
# models = O(#system topologies) * 
O(#attributes) * 
O(#abstraction levels) * 
O(#fidelities)

How do we manage all these models?

→ Use model transformations to generate the models as needed

1. Create *specific* transformation rules to generate analysis models
2. Create *general* rules for composition based on model correspondence templates in library
Vocabulary of the synthesis grammar
Model Library of Fluid Power Components

- Library of Fluid Power components
  - Defined as MAsCoMs (Multi-Aspect Component Models)
- Components are the reusable elements of design
- Multi-Aspect Component Models (MAsCoMs):
  - Group all models related to single fluid power component
  - Multiple disciplines and levels of abstraction
  - Modular
  - Formal & unambiguous
How to use MAsCoMs?

Log Splitter Design Example

ISO 1219 Fluid Power Graphics

Design Concept Schema - Hydraulic System
Generating System-Level Analyses

- Principle: Separation of Viewpoints
  - Separate model for each analysis perspective
  - Don’t mix analysis and structure models

- Approach: Composition
  - Compose component models into system-level model
  - Encode the composition rules as model transformations
  - Organize the composition patterns in a model library
  - Different types of models require different composition rules
Challenges

- **How to select the “right” component-level models?**
  - Perspective, compatibility, accuracy,…
  - Cost-benefit trade-off requires meta-information about models
    - Cost, accuracy, applicability,…

- **What are the different composition transformations?**
  - Transformation depends on formalism

- **What happens if the composition transformation requires additional information?**
  - E.g., synthesize structural description $\rightarrow$ convert to behavior
    $\rightarrow$ not all physical behavior parameters are available
Model Mapping

Systems Realization Laboratory
Automatic Translation from SysML to Modelica

Formal Graph Transformations
Mapping between SysML and Other Languages  
(based on work by Andy Schürr)

1. Define meta-models  
   • May require reverse-engineering meta-model
2. Create JMI adapter for tools
3. Define a model transformation  
   • Create graphs of correspondence between meta-models  
   • Triple Graph Grammar (TGG)
4. Compile rules (MOFLON) and load as plug-in

(Czarnecki, K., & Hellen, S., 2006)
Partial Modelica Metamodel
Partial SysML Metamodel in MOFLON
Transformation Rules

- Could be automatically generated through Triple Graph Grammar mechanism
TGG Mapping Mechanism in MOFLON

(Note: My interpretation of work by Andy Schürr)
Simulation in Dymola

Modelica Modelica
Lexical Representation (auto-generated from SysML)

Simulation Results

[Johnson, 2008 - Masters Thesis]
SysML Tool-Integration: INCOSE MBSE Challenge Project

SysML Tools

RSA/E+ / SysML
- Factory Model

No Magic / SysML
- Excavator System Model

RSA/E+ / SysML
- Excavator Executable Scenario

Operational Scenario

Interface & Transformation Tools
(MOFLON, XaiTools, ...)

Traditional Descriptive Tools

NX / MCAD Tool
- Excavator Boom Model

FactoryCAD
- Factory Layout Model

Excel
- Production Ramps

Traditional Simulation & Analysis Tools

ModelCenter
- Optimization Model

Ansys
- FEA Model

Mathematica
- Reliability Model

Excel
- Cost Model

Dymola
- Dig Cycle Model

eM-Plant
- Factory Simulation

2008-02-25a
Challenges

- How general is the TGG approach?
  - Is there a point at which it breaks down?
  - Limitations of bidirectional mappings?

- Is there a universal way to interface with disciplinary tools?
  - Is a JMI adapter the best way?

- And here I ran out of time… 😊
  … time to summarize
Summary of Approach

1. A Language for describing Fluid-Power circuits
   • Language is described by a meta-model
   • Valid circuits are represented as graphs

2. A Model Library with static knowledge
   • What are the available components?
   • What are their characteristics and behaviors?

3. A set of Model Transformations
   • Knowledge on how to combine components into circuit
   • Knowledge on how to generate analysis models from circuit descriptions

4. Language Mappings to/from other domains
   • Allows results to be viewed and edited (e.g. in SysML)
   • Allows circuits to be analyzed (e.g. in Dymola/Modelica)
Acknowledgements

- **Sponsors**
  - National Science Foundation: Center for Compact and Efficient Fluid Power
  - Deere & Co
  - Lockheed Martin

- **Students**
  - Jonathan Bankston
  - Jonathan Jobe (graduated)
  - Tommy Johnson (graduated)

- **Collaborators**
  - Roger Burkhart
  - Sandy Friedenthal
  - Leon McGinnis
  - Russell Peak
  - Dirk Schaefer

Questions?