Model Transformations in Model-Based Systems Engineering

Chris Paredis
Associate Director
Model-Based Systems Engineering Center
Georgia Tech
chris.paredis@me.gatech.edu
Objective

- Provide enough information about model transformations so that you can envision where in your organization the use of model transformations can add value.

- Examples are not meant to be an exhaustive review of the state of the art but only illustrations.
Overview of Presentation

- Background — Model Transformations
  - What is a model transformation? — some simple examples
  - Meta-modeling and Model transformations languages
- Model Transformations in MBSE?
  - A taxonomy of usage scenarios
  - Examples and illustrations
- How to get started?
- Summary
What is a Model Transformation?

- Model Transformation:
  
  Any operation on a model that produces another model

- Very general:
  
  Almost anything you “do” with a model

- Our primary interest:
  
  *Automated* model transformations
Example: Models Evolve Over Time

(from: A Practical Guide to SysML — Friedenthal, Moore, Steiner)
Example: Models Evolve Over Time

We forgot to drain the residue...

(from: A Practical Guide to SysML — Friedenthal, Moore, Steiner)
Example: Models Evolve Over Time

Hmm… we could make this a continuous process

(from: A Practical Guide to SysML — Friedenthal, Moore, Steiner)
Model Transformations

- A SysML model is a snapshot...
- How are the snapshots connected to each other?  
  → Model Transformations

- A model transformation represents knowledge...
- The more sophisticated the knowledge, the more difficult the transformation is to automate
Examples of Model Transformations

Transformation

System Model

Stage-Gate Documents

Transformation

Transformation

. pdf
.html
.pptx

Endogenous Transformation

Exogenous Transformation
Examples of Model Transformations

Transformation
System Model
Source Model
Transformation
Transformation
Transformation
Stage-Gate Documents
Transformation
Transformation
Transformation
Transformation
Simulation & Optimization
Project Management Metrics
Some History: Model-Driven Architecture

MDA Tool generates all or most of the implementation code for deployment technology selected by the developer.

Automated Transformations: A Closer Look…

Source Modeling Language

Target Modeling Language

Models must be Formal

Source Model

System Model

Target Model

Transformation

modeled in

modeled in

Source Modeling Language

Target Modeling Language

.pdf
.html
.pptx
Automated Transformations: A Closer Look…

**System Model**

**Source Meta-Model**
- defined by
- modeled in

**Source Modeling Language**
- modeled in

**Target Meta-Model**
- defined by
- modeled in

**Target Modeling Language**
- conforms to

Transformation

.Modeling Language
- Formal Syntax and Semantics
- wided by

*Source Model*  
*Target Model*
Modeling Languages are also Modeled Formally

- UML is defined by a meta-model
- This meta-model is defined in a meta-modeling language: The MOF language (Meta-Object Facility)

How is a meta-modeling language defined?

(from: A Practical Guide to SysML — Friedenthal, Moore, Steiner)
The OMG Meta-Model Infrastructure

- **M0**: System
- **M1**: Model
  - **M2**: Meta-model
    - **M3**: Meta-meta-model

- MOF conforms to UML
- UML conforms to User Model
- User Model represents M0 Reality
- M3 conforms to M2
- M2 conforms to M1
Model Transformation Specification

Source Meta-model refers to Transformation Specification conforms to Target Meta-model

Source Model reads Transformation Engine executes Target Model writes

conforms to executes conforms to

(Czarnecki, K., & Hellen, S., 2006)
Implementations of Model Transformation

- **Imperative**
  - Conventional programming tools
  - Access models through API of modeling tools
  - Java, Python, Ruby

- **Declarative**
  - Consist of multiple declarative model transformation rules
    - Left-hand side = applicability pattern
    - Right-hand side = model modification to apply
  - Transformation engine automatically determines which rules to apply
  - ATL, QVT, Moflon, GReAT, VIATRA2,…
Generative Model Transformation for Hydraulic System

- Model Transformation rules to generate systems

- Generate random system alternatives by applying rules in randomized order

(Example in MOFLON)
Generative Model Transformation

- Model Transformation rules to generate systems
- Generate random system alternatives by applying rules in randomized order

(Example in MOFLON)
Decision Tree of Generation Process

- **Add Cylinder**
  - Success (probability = 0.7)
  - Failure (probability = 0.3)
  - Success (probability = 0.7)
  - Success (probability = 0.3)
  - Failure

- **Add Pump**
  - Success (probability = 0.7)

- **Add Tank**
  - Success (probability = 0.7)

- **Add Directional Valve**
  - Success (probability = 0.7)

- **Failure**
Overview of Presentation

- Context — Model Transformations
  - What is a model transformation? — some simple examples
  - Meta-modeling and Model transformations languages

How can model transformations be useful in MBSE?
- A taxonomy of usage scenarios
- Examples and illustrations

- How to get started?
- Summary
Taxonomy for Model Transformations in MBSE

Organized by the nature of the knowledge encoded in the transformation

1. **Knowledge about Views**
   » Take information from a model and represent it in a different (graphical) syntax

2. **Knowledge about Analysis**
   » Take a structural description of a system and generate a corresponding analysis model

3. **Knowledge about Synthesis**
   » Generate a (more detailed) structural description

From simple to sophisticated
From specific to general
Why are Model Transformations Important?

Collaboratively Developing Complex Systems

- Testing
- Project Management
- Manufacturing
- Software
- Analysis
- CAD

- Fuselage
- Landing Gear
- Engines
- …
Why are Model Transformations Important?

Highly Interrelated Information & Knowledge

Relations / Dependencies

- Testing
- Project Management
- Manufacturing
- CAD
- Analysis
- Software

- Fuselage
- Landing Gear
- Engines
- …
Why are Model Transformations Important?

MBSE Allows for More Formal Communication

- Fuselage
- Landing Gear
- Engines
- …
Why are Model Transformations Important?

MBSE Allows for More Formal Communication

Testing

Software

Analysis

System Model

• Fuselage
• Landing Gear
• Engines
• …

Project Management

Manufacturing

CAD
Example: Representing CAD parts in Creo as SysML Blocks in MagicDraw

Step 2: A surrogate SysML block element is generated with the Creo parameters. This block represents the Creo model for the system engineer.

Step 3: The surrogate SysML block element (and corresponding instance) is connected to the Creo model. System engineer can sync parameter values.

Values read from Creo

(from: Manas Bajaj, InterCAX)
Example: Representing Assembly Structures in Windchill as SysML part properties in MagicDraw

SysML model
block structure

Part structure (BOM) in PLM systems
(e.g. Windchill)

(from: Manas Bajaj, InterCAX)
Notation: Connection of Nested Blocks

A token shows the connection of control-system elements with telescope elements: for example, the sensor cabinet ("sensCabinet") is connected with the "coolantReturn" and "coolantSupply." The solid line is a nested connector. It crosses the boundaries of the encapsulated system blocks.

Figure Diagram

{captionText = "Nested connectors",
diagram = ObservatoryContext_NestedConnector}
Cookbook for MBSE with SysML

Foreword

This book contains guidelines, recipes, and best practices for Model Based Systems Engineering (MBSE) with the Systems Modeling Language (SysML) on the basis of an interdisciplinary model of a complex real-world project, a high-tech opto-mechatronic system called the Active Phasing Experiment (APE).

The book addresses:

- Model structure and overview
- Objectives and Requirements
- Context, System Structure
- Behavior and Data
- Verification
- Model library and domain specific profiles

Part: MBSE in Telescope Modelling

Chapter 1: Introduction

In the framework of INCOSE’s strategic initiative, the Systems Engineering Vision 2020, one of the main areas of focus is model-based systems engineering.
SysML is a new language. This creates two inherent challenges. Is SysML sufficiently mature for real projects, and is it accepted by a wide range of systems engineers? Especially the fact that SysML is based on UML, which is a special tool on these challenges. Could a modeling language that was initially designed for software development be used to model systems, and will systems engineers accept a language with origins in the software discipline? An overall result of our project is that this question can be answered yes. The APE project is a proof of concept of SysML. It is complex and interdisciplinary without a special focus on software; this is a real system and not the simplified car or machine so often used as demonstration project. Although we found that SysML is practicable to model complex systems, we have made a list of the language’s shortcomings. The most significant ones are these:

- Variant modeling
- Connections of nested blocks
- Mapping of interfaces with nested ports
- Logical vs. physical decomposition
- Functional mediator abstraction

SysML + Profile

Transformation

DocBook XML file

XSLT Transformation

.html  .pdf  .docx
Using DocGen at JPL

Taxonomy for Model Transformations in MBSE

- Organized by the nature of the knowledge encoded in the transformation
  1. **Knowledge about Views**
     - Take information from a model and represent it in a different (graphical) syntax
  
  Knowledge about Analysis
  - Take a structural description of a system and generate a corresponding analysis model

  3. **Knowledge about Synthesis**
     - Generate a (more detailed) structural description

- From simple to sophisticated
  From specific to general
Why are **Model Transformations** Important?

**MBSE Allows for More Formal Communication**

- Testing
- Software
- Analysis
- Project Management
- Manufacturing
- CAD

- Fuselage
- Landing Gear
- Engines
- …
Execute Parametric Models in ModelCenter

- Syntactic transformation from parametrics to ModelCenter
SysML-Modelica Transformation follows the principles of Model-Driven Architecture (MDA)
SysML4Modelica

Bi-directional Transformation

Modelica

model Spring "Linear 1D translational spring"
  extends Translational.Interfaces.PartialCompliant;
  parameter SI.TranslationalSpringConstant c(final min=0, start = 1) "spring constant ";
  parameter SI.Distance s_rel0=0 "unstretched spring length";
  equation
    f = c*(s_rel - s_rel0);
end Spring;
SysML4Modelica to Modelica Syntactical Mapping
Transform Structure into Analysis Model
Transform Structure into Analysis Model

- Transformation requires deep analysis knowledge…
Model Transformations for Fluid Power

Transformation Rules:

– If Cylinder in descriptive, then ModelicaCylinder in analytical
– If Pump in descriptive, then ModelicaPump and ModelicaFilter in analytical
– ...
– If connection between Pump and Valve, then ModelicaConnection between ModelicaPump and ModelicaValve
– ...

Not a good idea!
Too many rules…
Too specific, brittle
Define Model Correspondences

Correspondence Models
Structure to Analysis Mapping

SysML Descriptive Model

Transformation

Only 2 rules
One for components
One for connections

Modelica Model

SysML4Modelica Analysis Model

Transformation
Generation of Vehicle Analyses
(Work with students B. Bailey, J. Branscomb & Ford Motor Company)

- Reference model of logical vehicle architecture in SysML
- Specialize the reference model into a variant for a specific vehicle program
- Specify the types of analyses needed
Generation of Vehicle Analyses
(Work with students B. Bailey, J. Branscomb & Ford Motor Company)

- Automatically generate templates for:
  - Modelica (for physics-based modeling)
  - Simulink (for controls)
- Subsystem modeling
  - SME fill in model details
  - Validate subsystem models
- System-level model integration
  - Guaranteed compatibility
  - Modelica model integrated into Simulink
Taxonomy for Model Transformations in MBSE

- Organized by the nature of the knowledge encoded in the transformation
  
  1. **Knowledge about Views**
     - Take information from a model and represent it in a different (graphical) syntax
  
  2. **Knowledge about Analysis**
     - Take a structural description of a system and generate a corresponding analysis model

- **Knowledge about Synthesis**
  - Generate a (more detailed) structural description

- From simple to sophisticated
  - From specific to general
Synthesis Rules

- Capture heuristic synthesis knowledge: What are promising system architectures?

- Difficult to create a transparent “grammar” — which architectures are covered by the grammar?
Additional Example of Synthesis Knowledge

- Variant modeling — constraint-based representation of a set of architectures (mbse.gfse.de)

- F6 program at JPL: Generation and analysis of fractionated satellite systems
Architecture Exploration Using SysML & CPLEX
(Work with former students Alek Kerzhner)

1. Formulate Problem in SysML
2. Transform to Superstructure
3. Transform to Introduce Behavior
4. Transform to Linearize
5. Transform to AIMMS syntax
6. Solve using CPLEX
7. Transform solutions into SysML

Generation of Mixed Integer Programming model through transformations
Overview of Presentation

- Context — Model Transformations
  - What is a model transformation? — some simple examples
  - Meta-modeling and Model transformations languages

- How can model transformations be useful in MBSE?
  - A taxonomy of usage scenarios
  - Examples and illustrations

How to get started?

- Summary
How to get started..

- Imperative modeling
  - Traditional programming languages: Java, Python,…
  - Direct use of the APIs of SysML and Analysis tools

- Declarative Model Transformation Languages
  - ATL, QVT, Moflon, VIATRA2,…
  - All are Eclipse based
  - Integration of Model Transformation tools into current SysML tools is not well supported…
Model Transformation Tools

(Czarnecki, K., & Hellen, S., 2006)
So What? Should I use Model Transformations?

It depends…

◆ Costs
  – Knowledge capture, management is expensive
  – Requires “knowledge engineers” who understand how best to generalize knowledge and encode it in transformations
  – Infrastructure

◆ Benefits
  – Improved consistency
  – Once the knowledge has been captured, the cost of applying it is very small
  – Eliminate tedious, non-value-added tasks in MBSE process
Key Take-Aways

1. Model Transformations
   - Any operation on a model that produces another model
   - Automated model transformations defined at meta-model level
   - Model transformation tools in Eclipse

2. Model Transformations in MBSE
   - Views: Synchronize views and maintain consistency
   - Analysis: Generate analysis models
   - Synthesis: Generate promising architectures

3. Cost-Benefit
   - Identify simple MBSE tasks that are tedious and error-prone