# CSSE 490 Model-Based Software Engineering: Software Factories

**Shawn Bohner**

**Office: Moench Room F212**

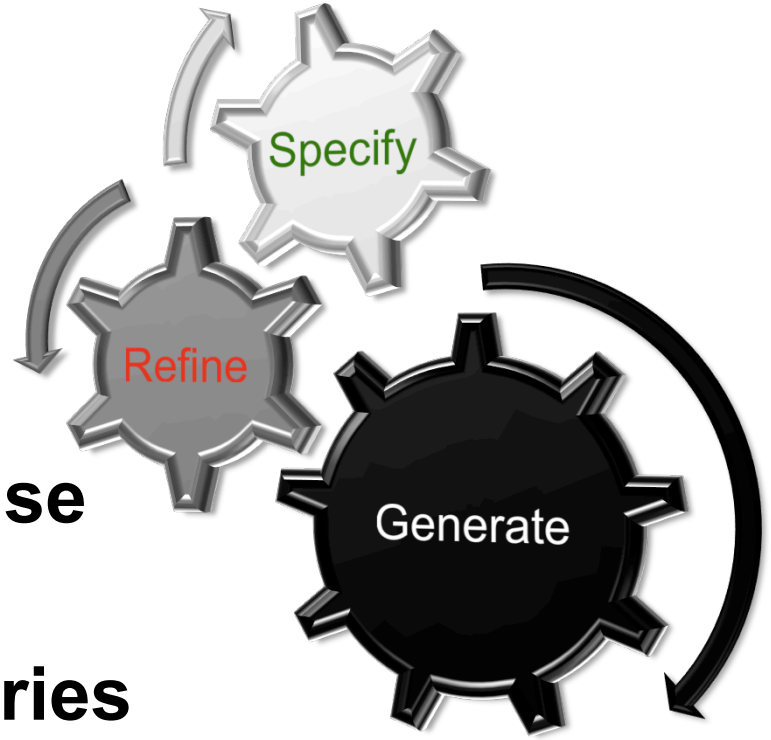
**Phone: (812) 877-8685**

**Email: bohner@rose-hulman.edu**

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Learning Outcomes: MBE Discipline

***Relate Model-Based Engineering as an engineering discipline.***




- Discussion of Milestone 3
- Finish Class Project Exercise
- Example: DRACO
- Context for Software Factories
- Key Software Factory Concepts
- Short Microsoft SF Example

# Exercise: Reflect on Class Project

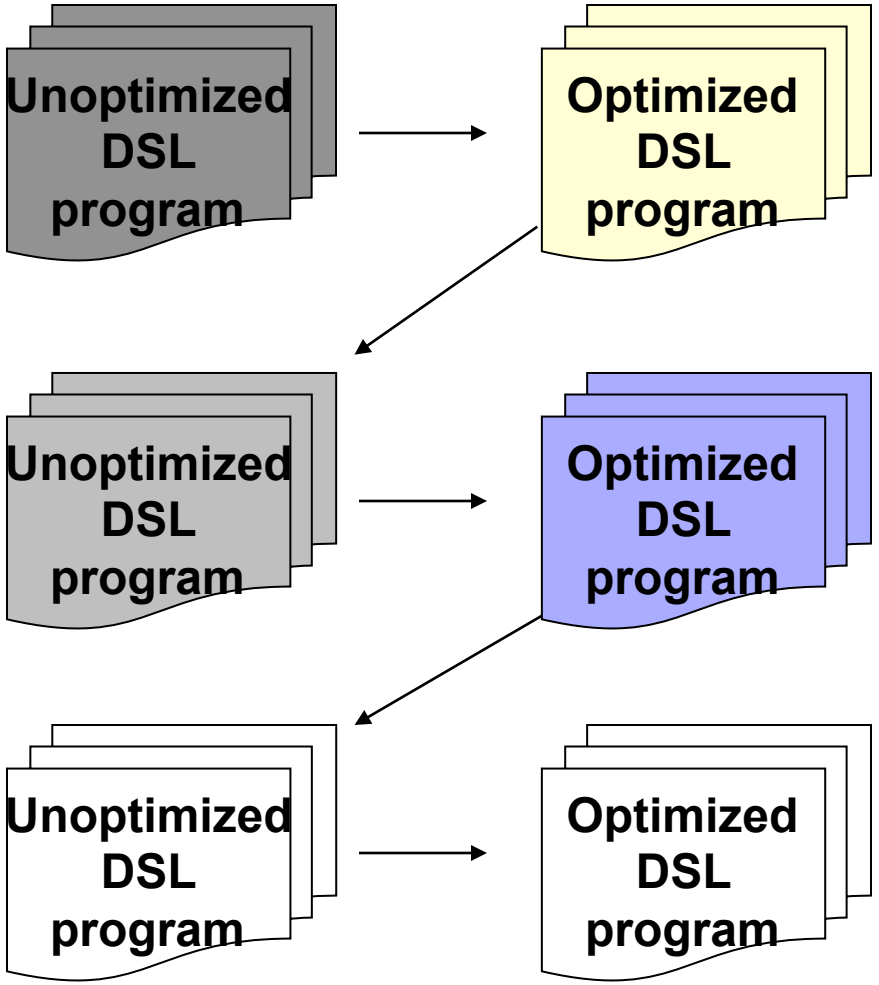
***Describe how Milestone 3 Environment Automates Software Development.***

- What are the assembling steps?
- What are the transformational steps?
- Is the domain language a separate concern?
- Draw a diagram of how it works?

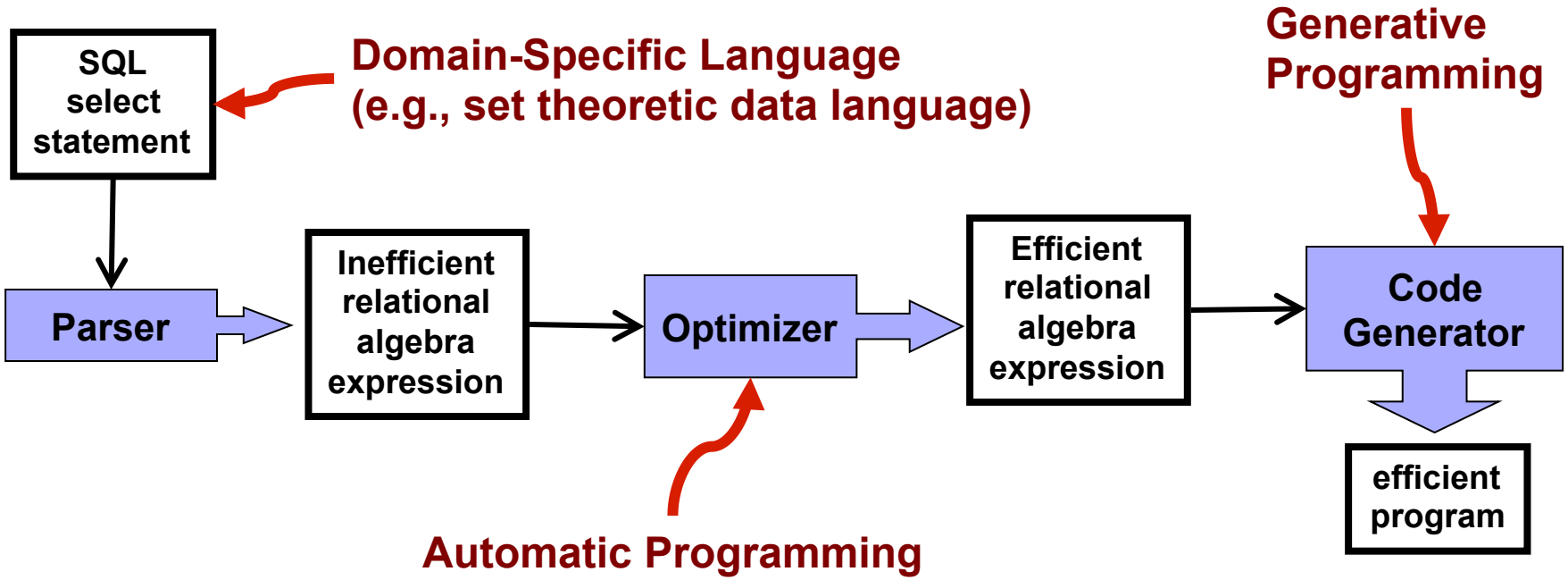
# Early Generative Reuse System

## DRACO – Jim Neighbors (1984)

- Programs are written in domain-specific languages
- Optimize DSL programs (because the domain abstractions are visible)
- Translate DSL to another, lower-level abstraction DSL and do the same
- Process repeats until you get to machine code

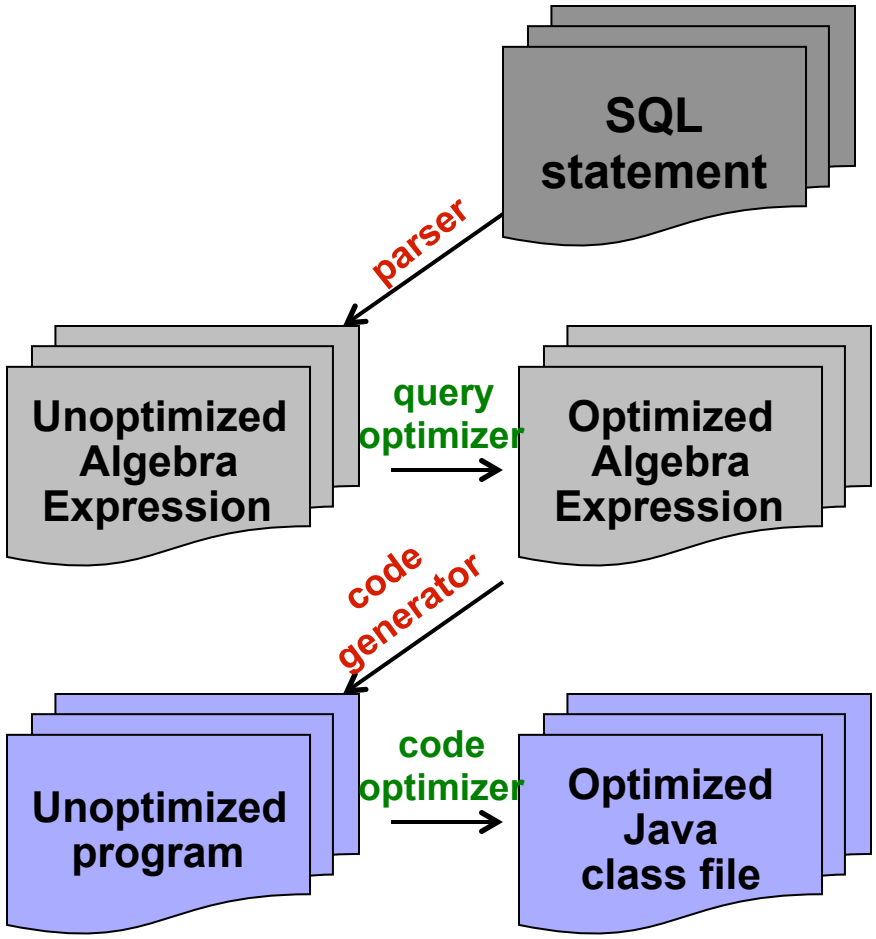


# DRACO Perspective of Programming



# DRACO View of Query Optimization

- **Generative Programming** occurs when mapping between levels of abstraction
- **Automatic Programming** occurs when optimizing (or refining/transforming) within a level of abstraction is done automatically



# MBSE Requires Advances in:

- **Generative Programming (GP)**
  - Understand domain well enough to generate software
  - Programs that synthesize other programs
    - Metaprogramming, skeleton communities
- **Domain-Specific Languages (DSLs)**
  - Elevate program specifications to compact domain-specific notations that are easier to write & maintain
- **Automatic Programming**
  - Culmination of GP and DSLs
  - Generate efficient programs from declarative specs
  - Precursor to Model-Based Software Engineering

# A Case for Automated Programming

- Structure of matter is fundamental to chemistry and physics
- Structure of material is fundamental to engineers
- Structure of software is fundamental to software engineering
  - structure = modules and their composition
- Structure of software not well-understood
  - Software design, which is a process to define the structure of an application, is an art form
  - We need a science of design for software

# So Did the Database People Get it?

CODD, E. F.

**A relational model of data for large shared data banks.** *Comm. ACM 13*, 6 (June 1970), 377–387.

In this article, the author discusses the advantages of using a relational rather than a network model for representing data in a data management system. It is his contention that such a model would provide greater independence of the data from application programs and would provide a means of making consistency checks on facts in the database.

The author describes a convention under which most data structures can be represented as sets of relations. He then describes how some of the common set operations correspond to database operations.

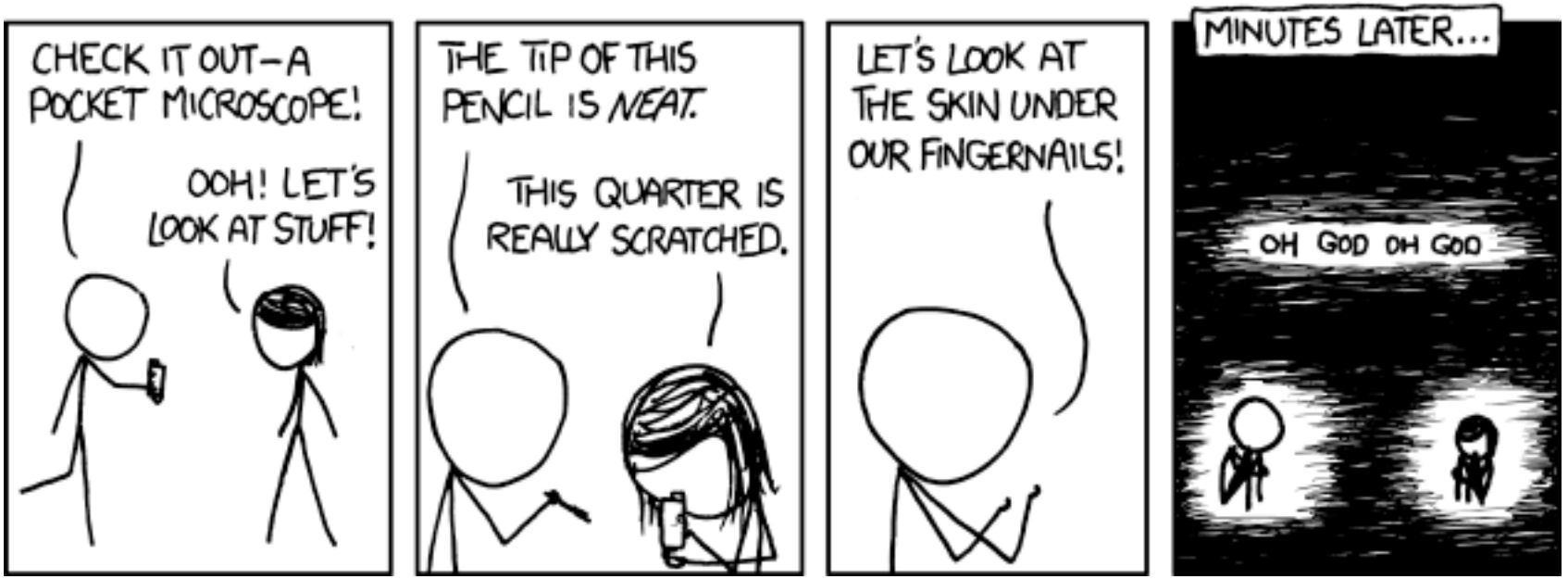
While the relational model is not a new one, this paper probably contains its most rigorous and elegant statement and, as such, represents a valuable contribution to the theory of data processing.

However, it would appear to this reviewer that the model is of more theoretical than practical interest. While the relational model does indeed eliminate some data dependencies (e.g., ordering dependence) it does so by eliminating the practical advantages of ordering. While it does provide a mechanism for automatically making some data checks, these checks would often be so computationally painful that they would be of little practical value.

*R. W. Elliott*, College Station, Texas
#20,780 March 1971

- **Nope…Look at the Computing Surveys Review of Codd's 1970 seminal paper on the Relational Model...**
- **Do you get it?**
- **This is a race and my leg is running now…**
  - **Your leg is coming up!**

# Looking at the details ....



# Key Original Papers

## Automatic Programming

Robert Balzer. A 15-year Perspective on Automatic Programming. *IEEE Transactions on Software Engineering*, 11(11):1257–1268, November 1985.

David R. Barstow. Domain-Specific Automatic Programming. *IEEE Transactions on Software Engineering*, 11(11):1321–1336, November 1985.

Charles Rich and Richard C. Waters. Automatic Programming: Myths and Prospects. *IEEE Computer*, 21(8):40–51, August 1988.

## Automated Planning

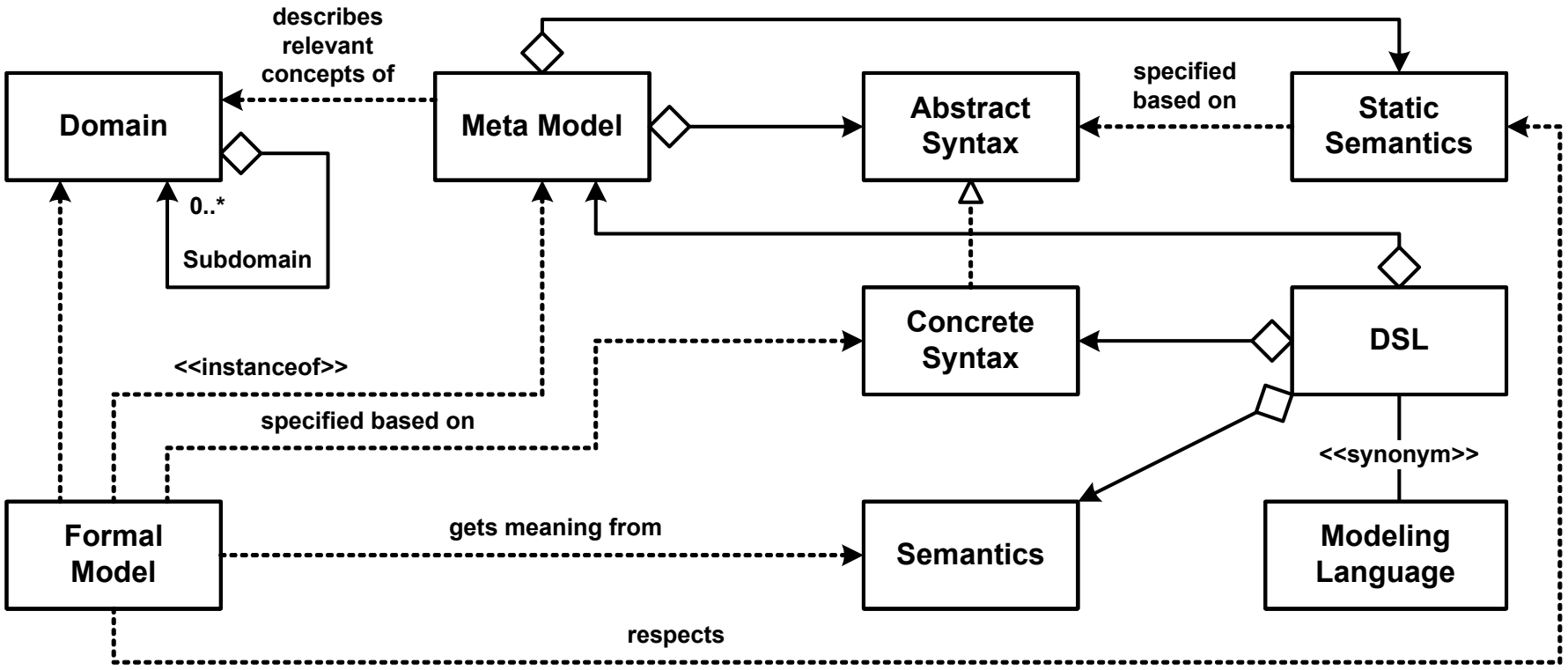
Keith Golden. A Domain Description Language for Data Processing. Proc. of the International Conf. on Automated Planning and Scheduling, 2003.

M. Stickel et al. Deductive Composition of Astronomical Software from Subroutine Libraries. Proc. of the International Conf. on Automated Deduction, 1994.

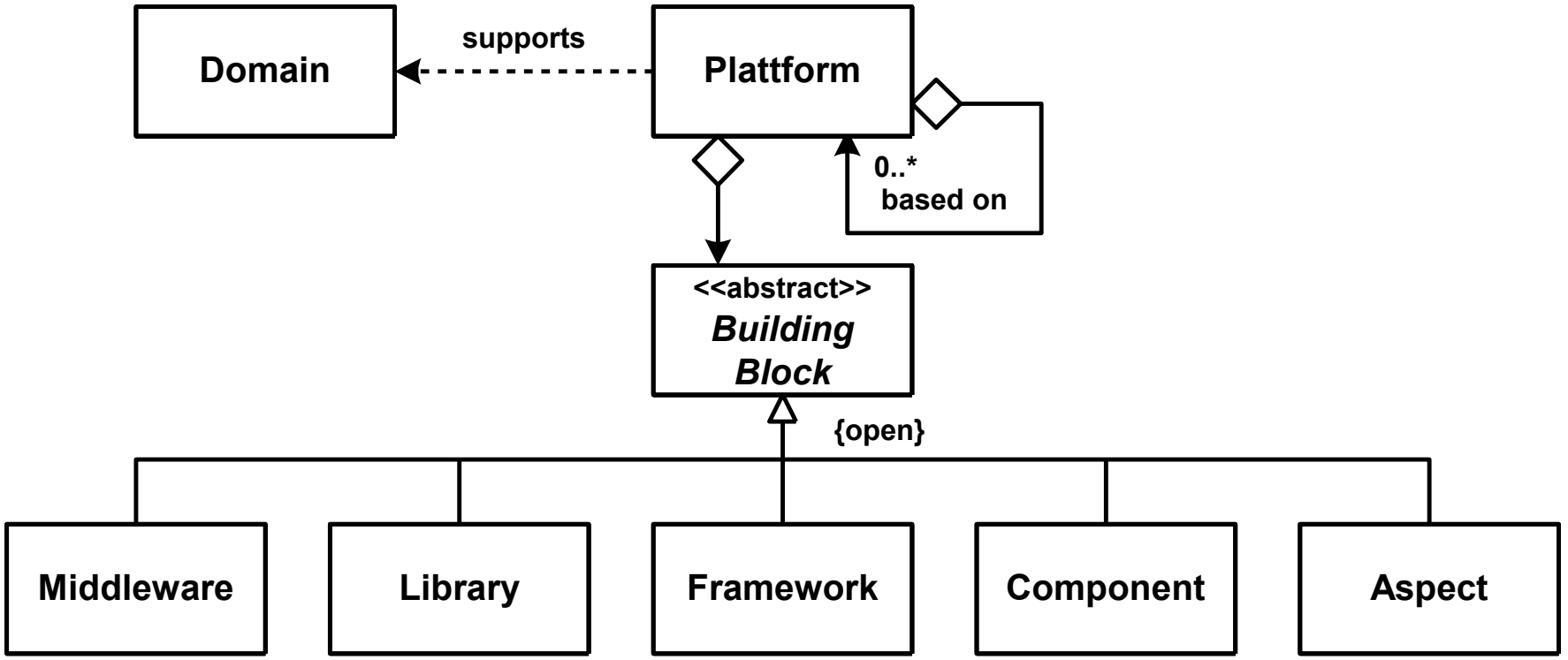
# If you automated software production, what would the factory contain?


- Think for 15 seconds…
- Let's talk…

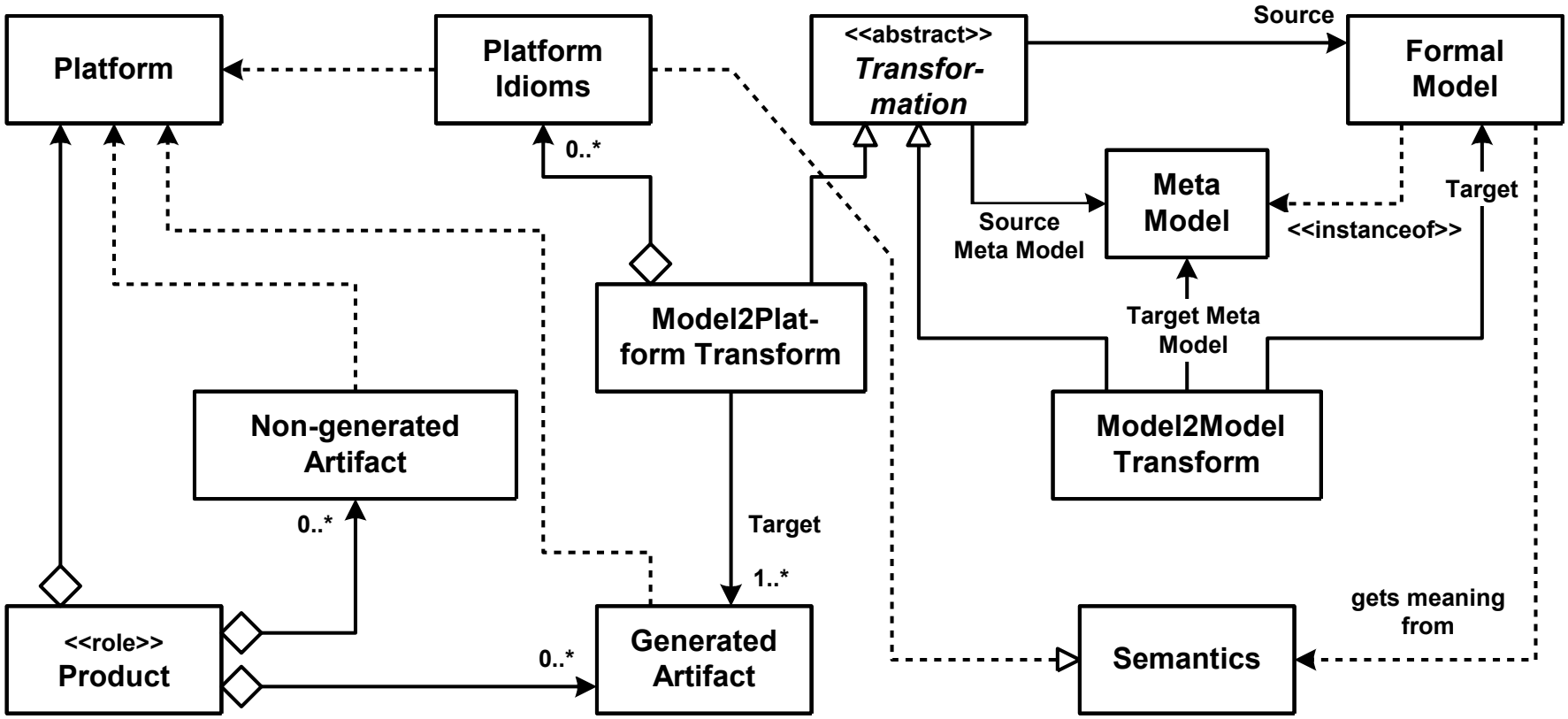
# Recall: Modeling Concepts



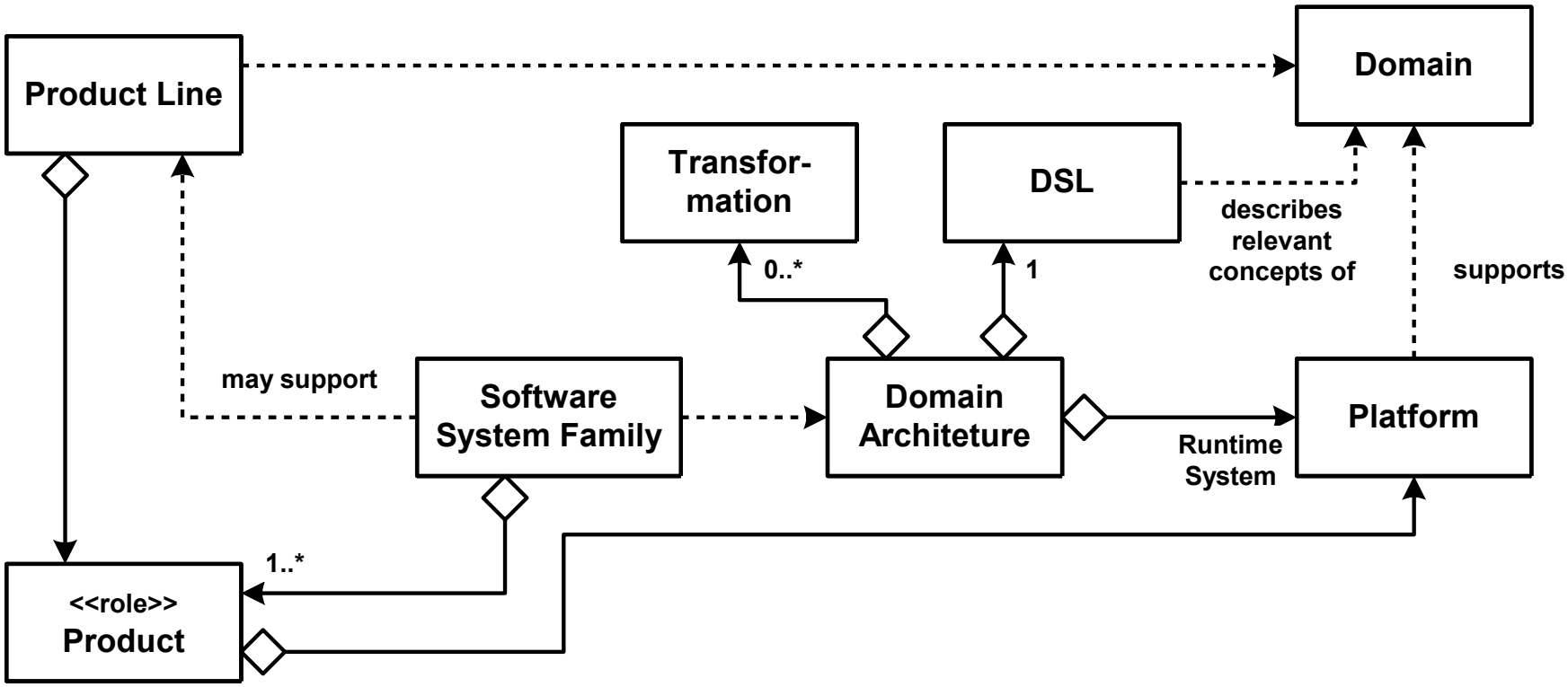
# Recall: Platform Concepts



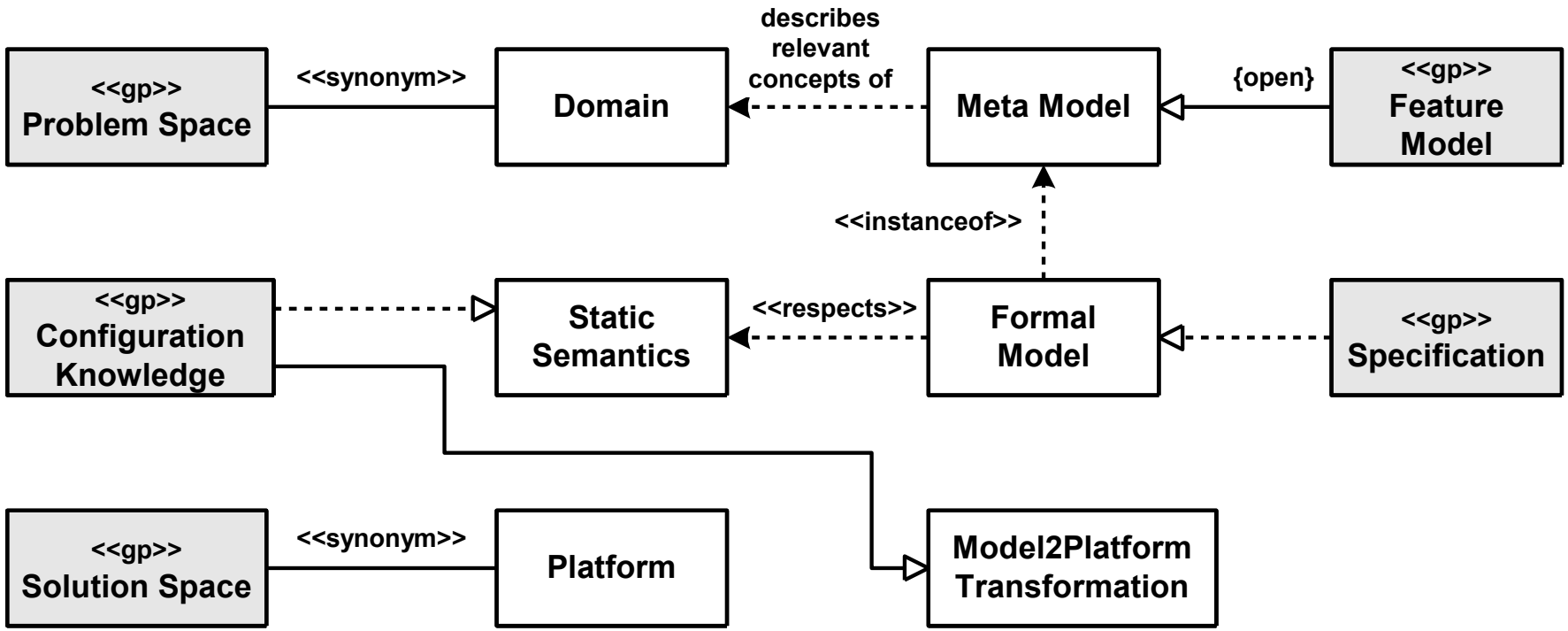
# Recall Transformations



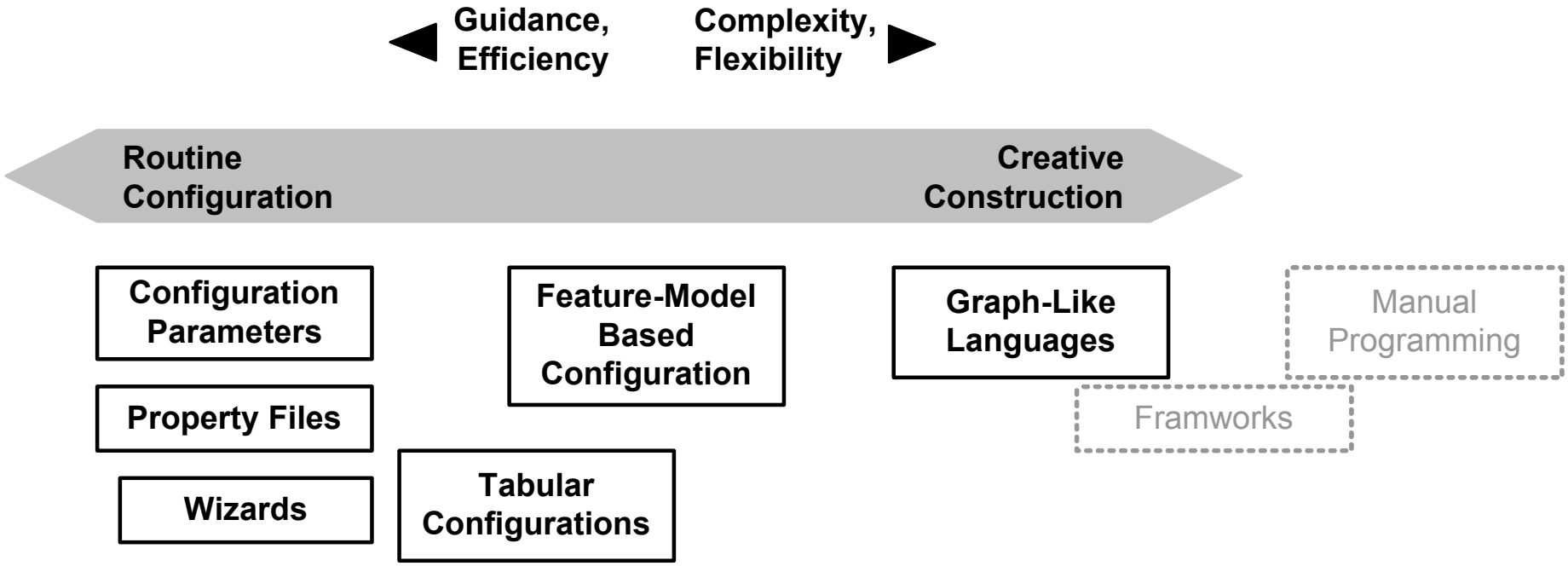
# Recall: Software System Product Lines



# Generative Programming Concepts



# More GP Concepts




- Platform consists of maximally combinable and minimally redundant components
- GP is often routine configuration as opposed to creative construction

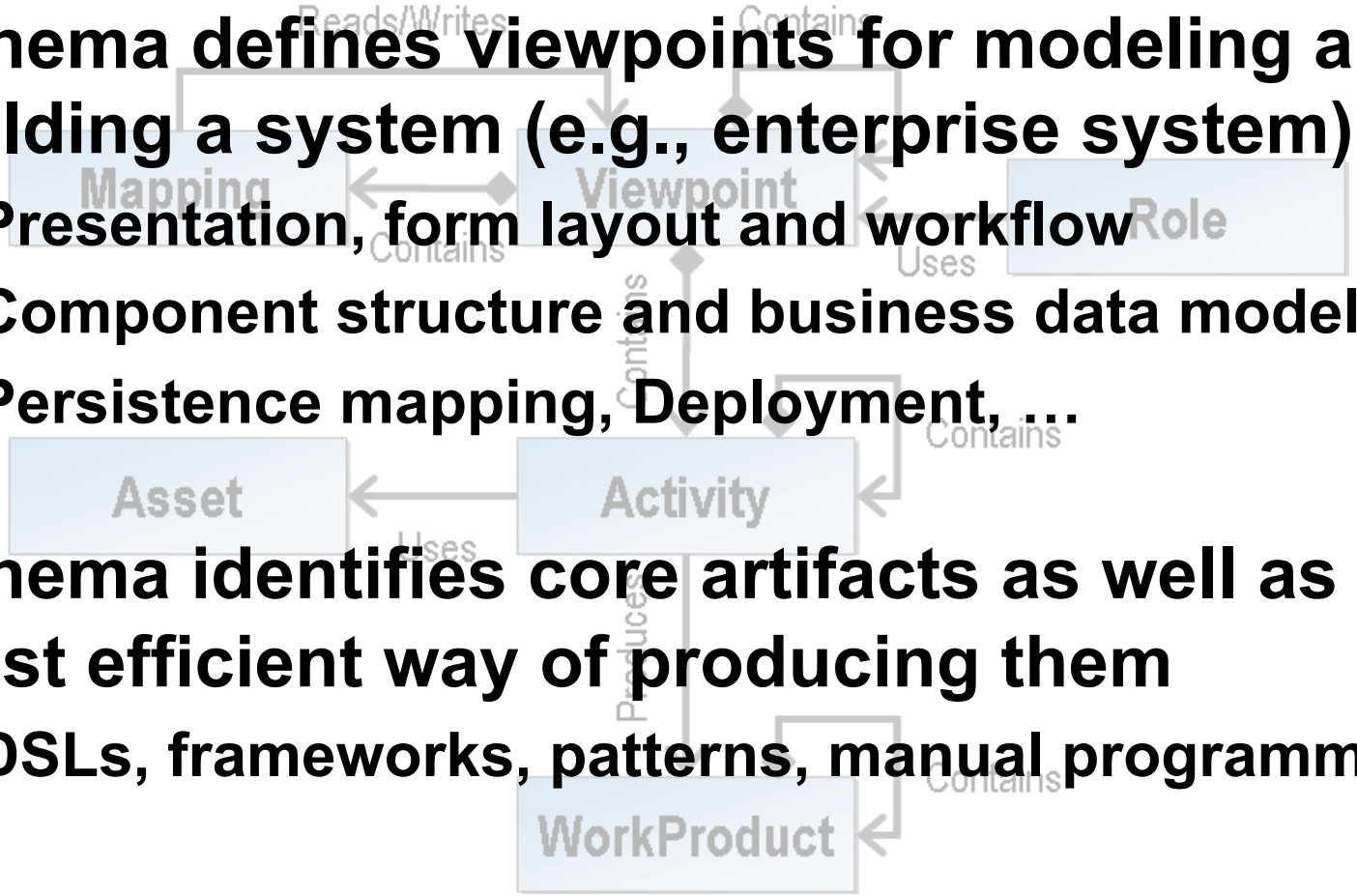
# Software Factories

- 1990's Software Factories emerged as the new automated programming
- Faced an untrained community coupled with limitations in computing capabilities
  - The Virtual Software Factory
  - Software Templates
  - Software Refinery

- Devolved into IDEs configured for efficient development of Domain applications (led by Microsoft these days)

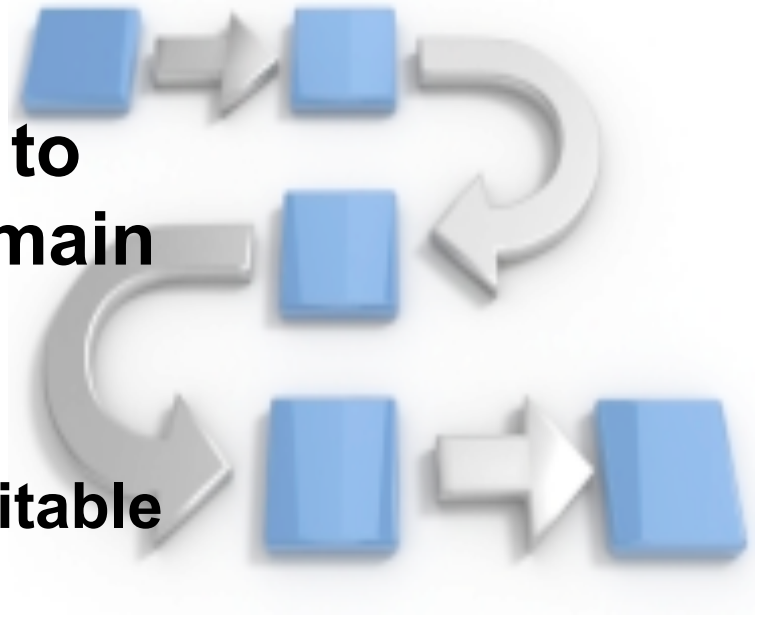
# Software Factories Schema

- Schema defines viewpoints for modeling and building a system (e.g., enterprise system):
  - Presentation, form layout and workflow
  - Component structure and business data model
  - Persistence mapping, Deployment, …
- Schema identifies core artifacts as well as the most efficient way of producing them
  - DSLs, frameworks, patterns, manual programming
- Schema identifies commonalities and differences among applications in the domain



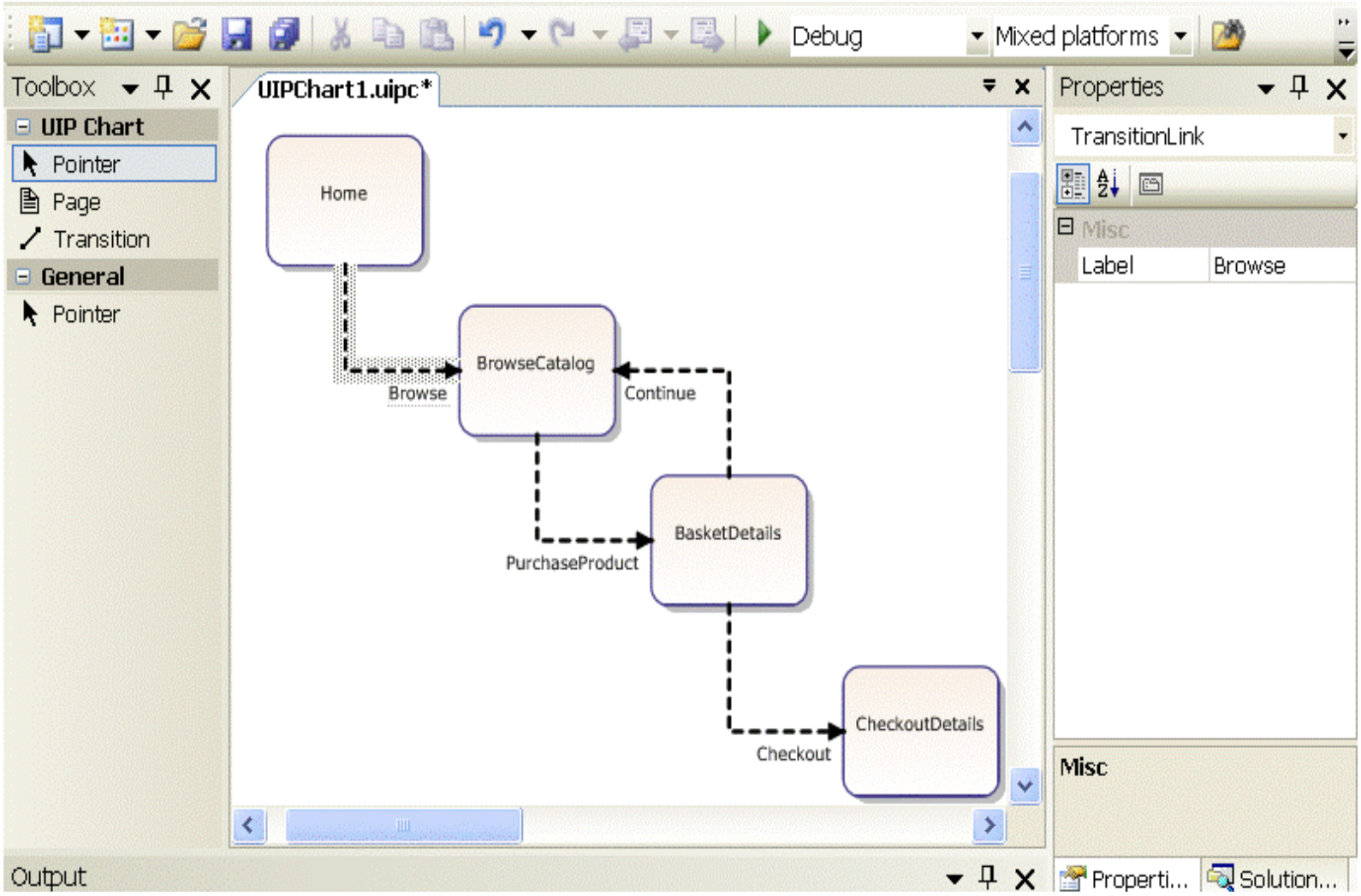
# Software Factories Templates

- Makes the Schema usable



- Load SF Template into IDE to configure it for specific domain
  - Provides the necessary frameworks or libraries
  - Contributes project types suitable for the factory
  - Delivers build scripts
  - Extends IDE with DSL editors and transformations

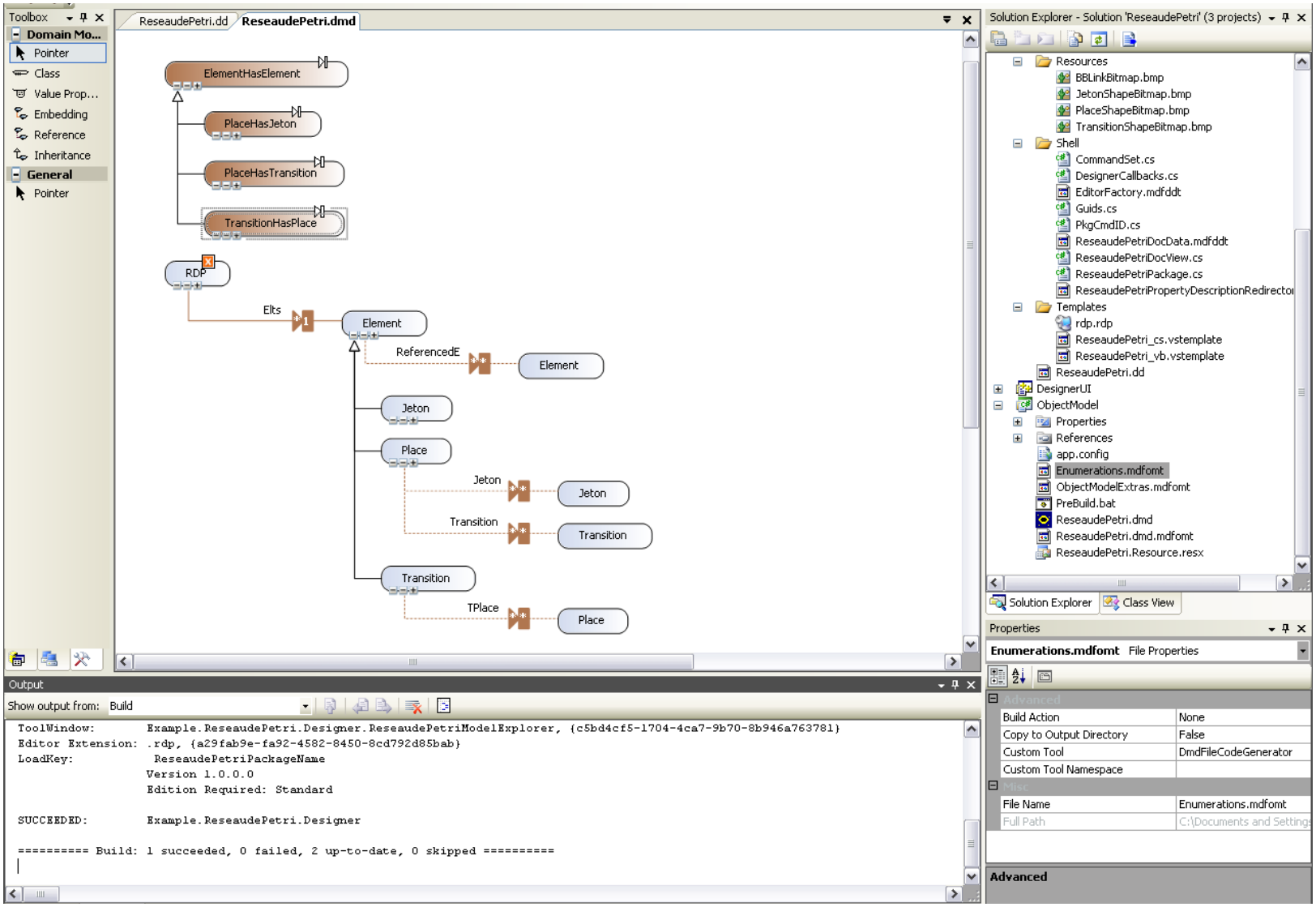
# MS DSL Tools Example





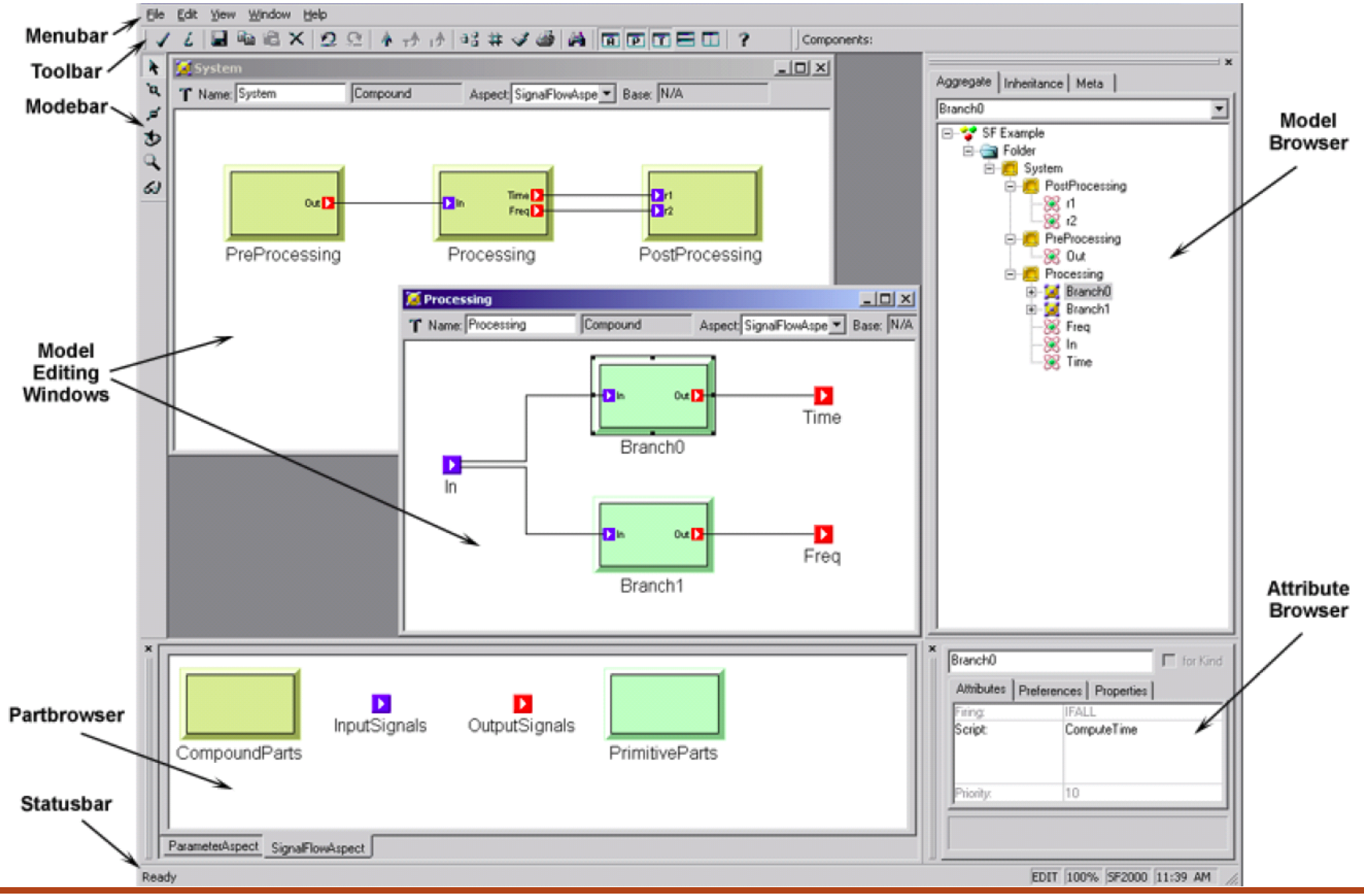

Source: http://galilee.microsoft.fr/(ncqibzbvkp2ezr45aevbcqjk)/a17fdcfb90f14a7592045f1c0fc5e97f/

# Defining a Metamodel



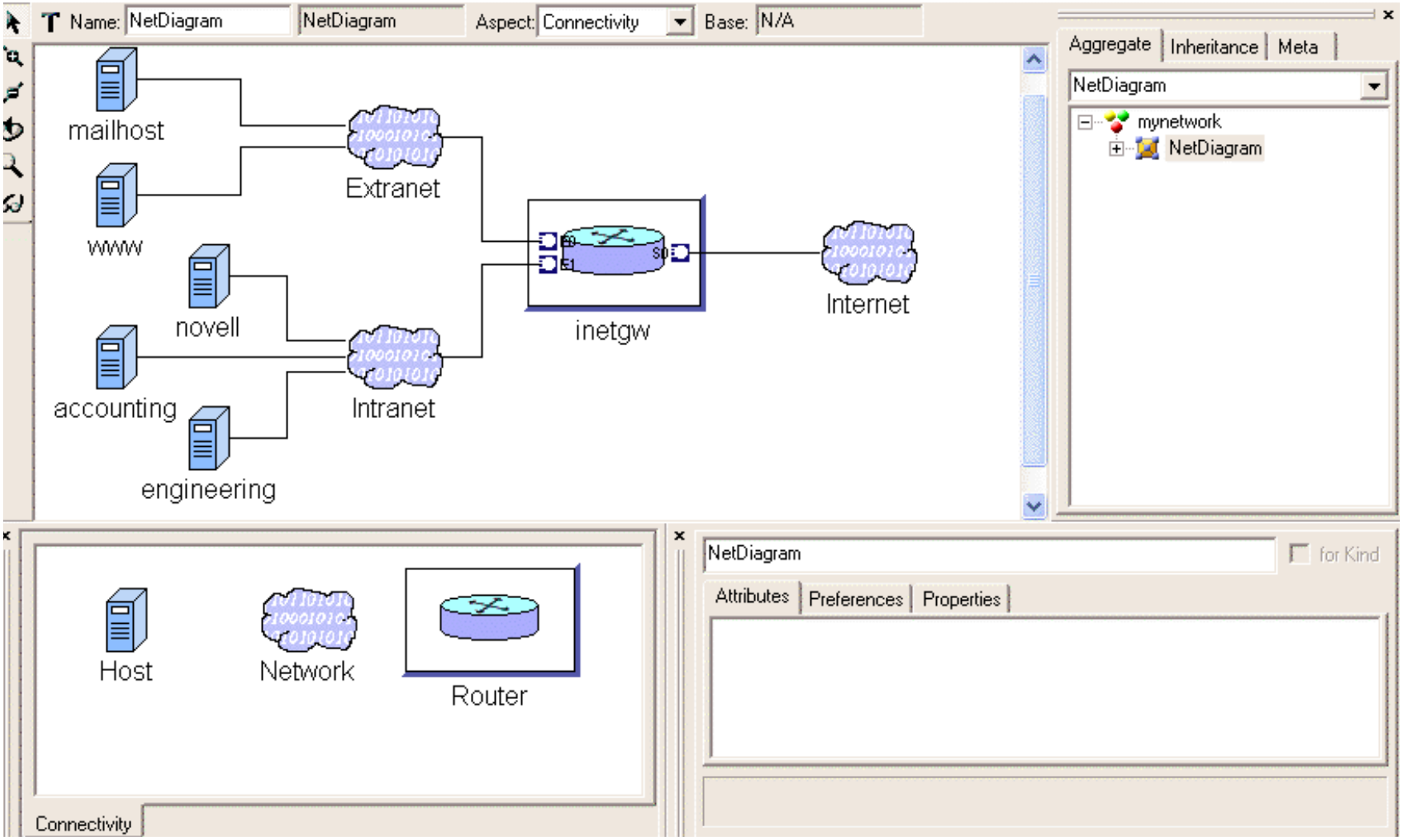

Source: http://galilee.microsoft.fr/(ncqibzbvkp2ezr45aevbcqjk)/a17fdcfb90f14a7592045f1c0fc5e97f/

# Graphical Model Editor

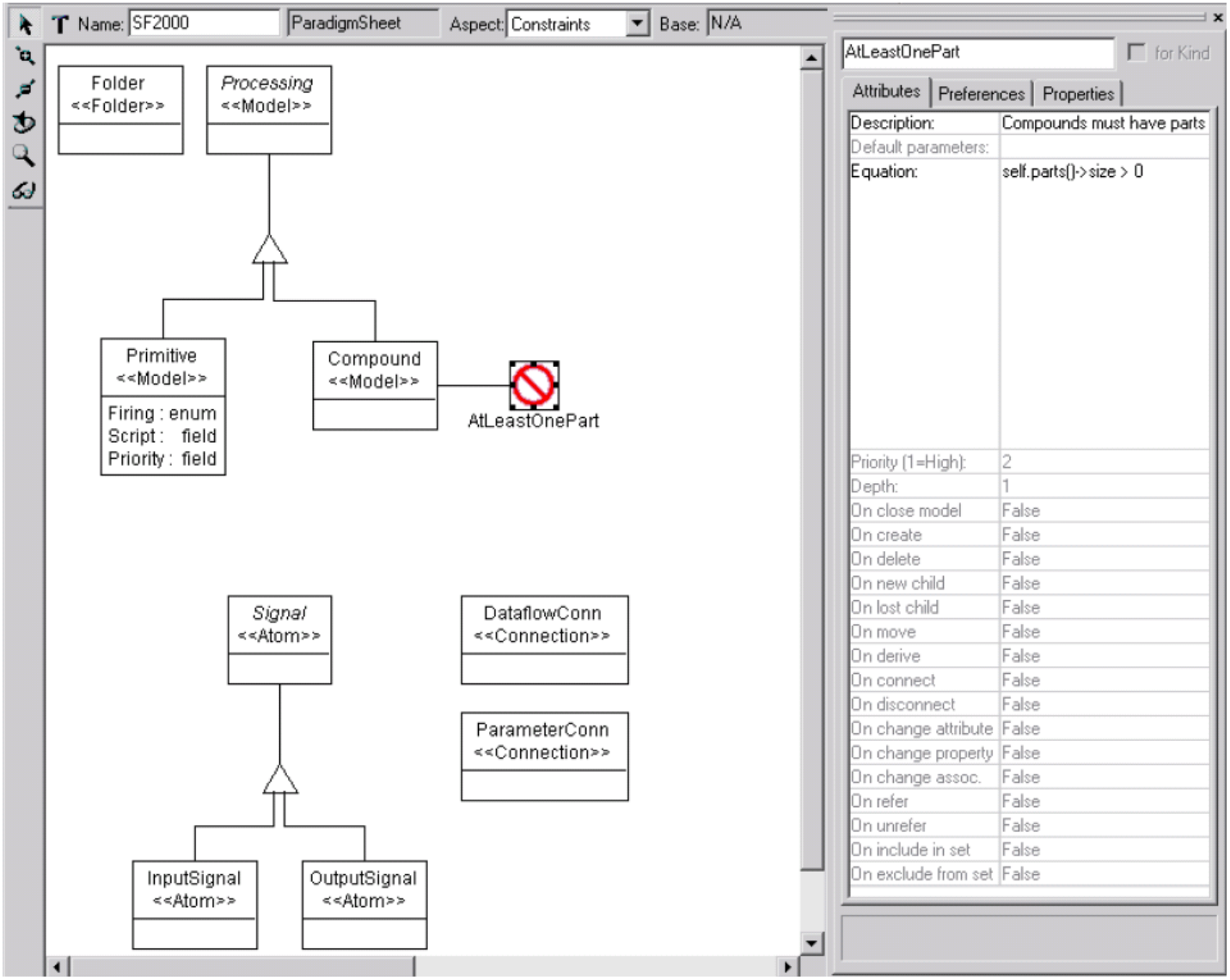



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# GME: Modeling based on previously defined MM



# GME: OCL Constraint Validation



# Homework and Milestone Reminders

- Milestone 3: Light-Weight Transformation Environment (see Milestone 3 assignment)
  - Due by 11:55pm, Thursday, May 12th, 2011.
- Milestone 4: Final MBSE Environment (see Milestone 4 assignment)
  - Due by 11:55pm, Friday, May 13th, 2011.