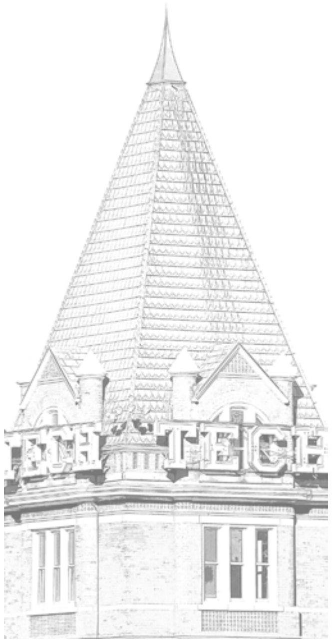
What Is a Reference Model and What Is It Good For?



Leon F. McGinnis, Professor Emeritus

Tim Sprock, Post-Doctoral Fellow

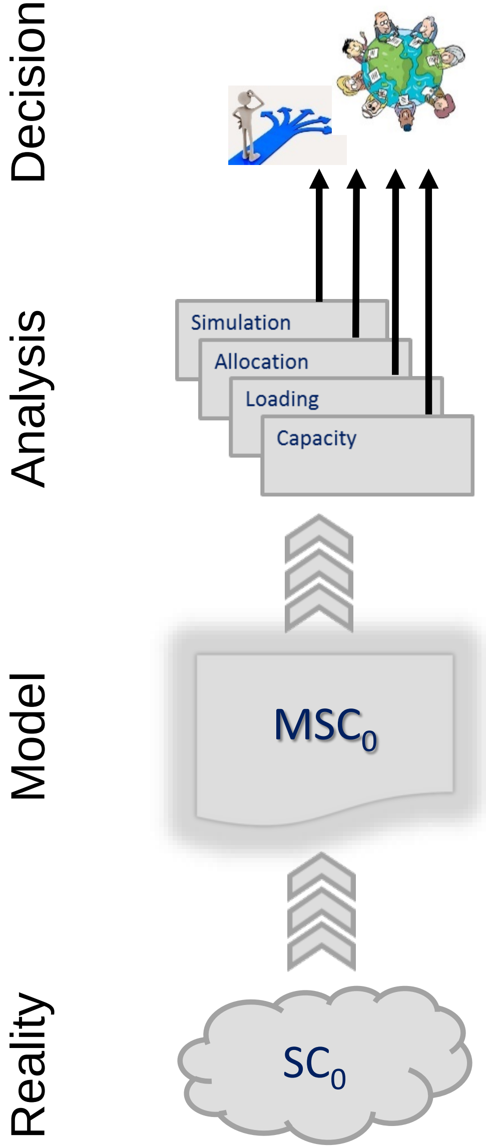
George Thiers, Post-Doctoral Fellow

Dagstuhl 16062

10Feb2016

Georgia Institute of Technology

CONTEXT: 1



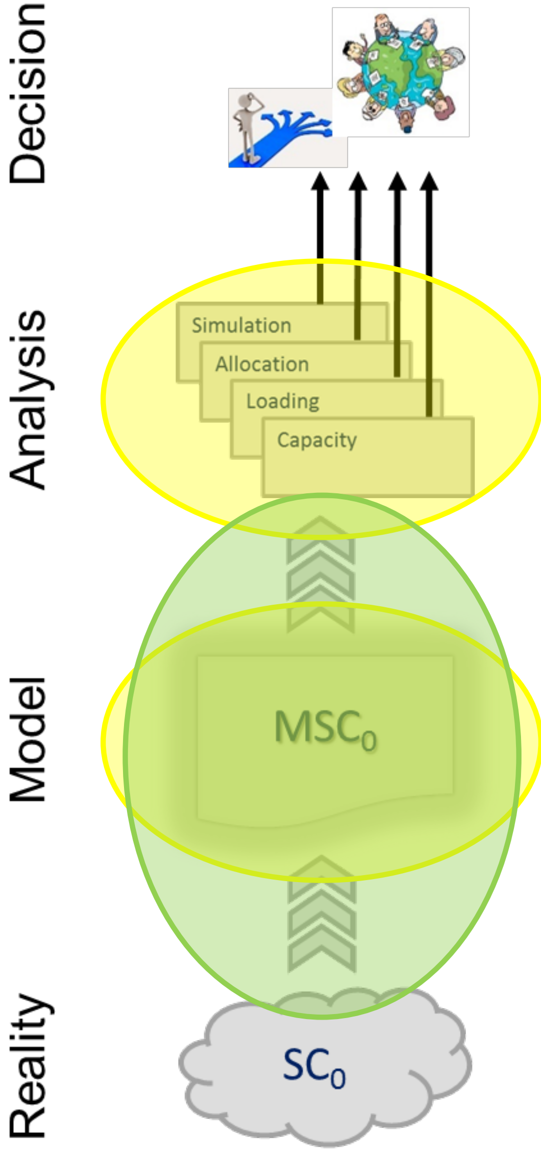
Weight
Dimension
Count

Cold
Rhomboid
Fits in hand

shutterstock · 127516526

Possible because we share a language for communicating about ice cubes and share experience of ice cubes

CONTEXT: 2

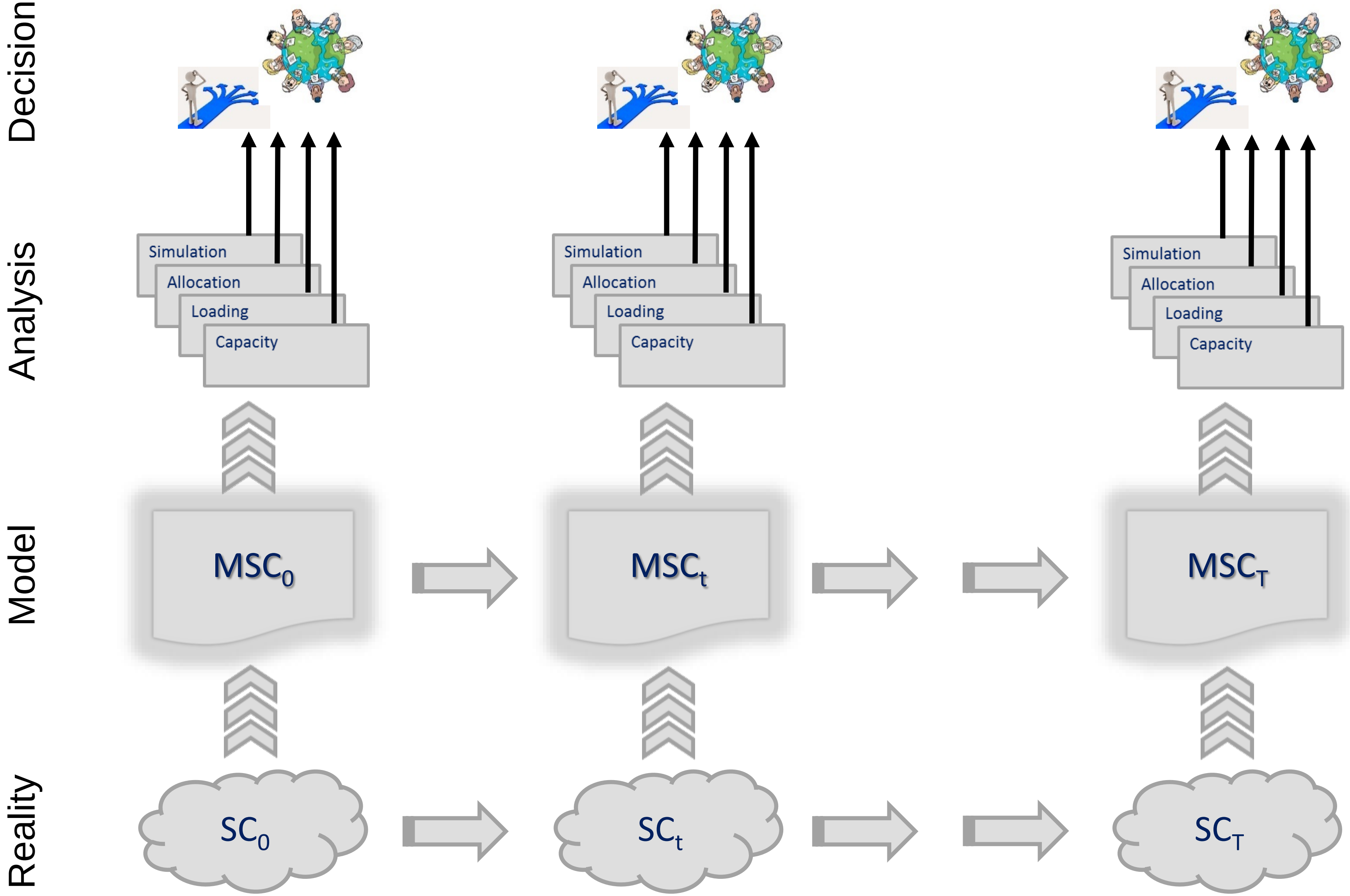


Most presentations so far—here's an analysis we can do

Hans' overview—here's how we think about our supply chain

Where I want to focus—how do we create models and how do we exploit them

CONTEXT



Decision
Analysis
Model
Reality
Simulation
Allocation
Loading
Capacity
Simulation
Allocation
Loading
Capacity
Simulation
Allocation
Loading
Capacity
MSC_0
MSC_t
MSC_T
SC_0
SC_t
SC_T

OBSERVATIONS

- When we discuss the "reality", we are using models, so we can really only discuss models
- The model is not the reality ("all models are wrong …")
- Reality changes, so the model must change
- The model does not have to be ***(should not be!)*** the same as the analysis
- Analysis is in service to decision making
- We want this to be "routine" (we know how to formulate and execute the analyses!)

PRINCIPLES

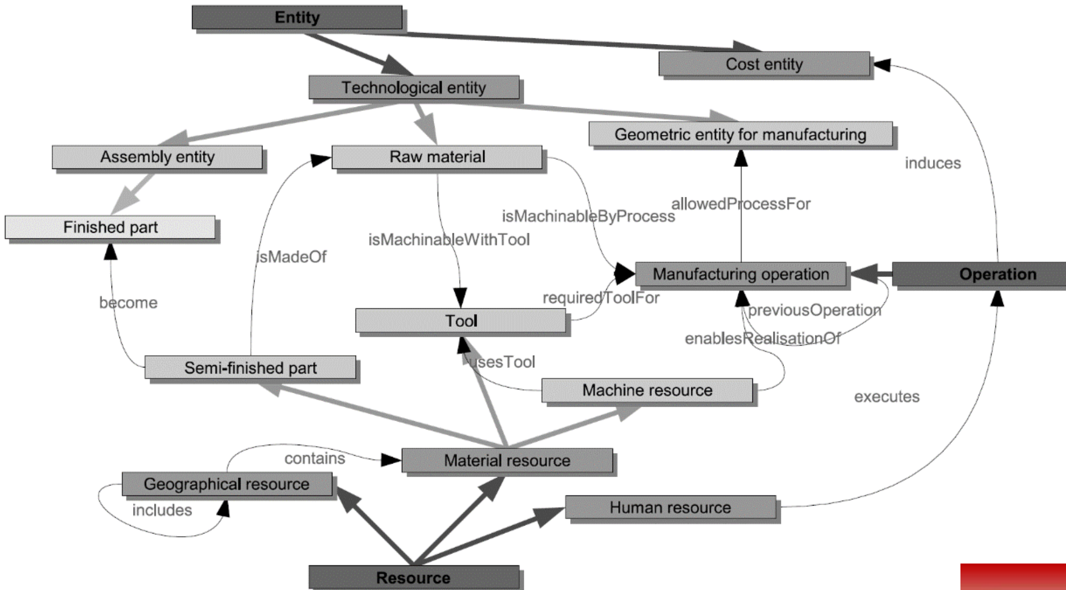
- MSC must unambiguously describe structure, behavior and control
- We must be able to detect changes in SC and reflect them in MSC (impact of accurate, r/t data …)
- MSC should be the *reference model* for *all* decision support analyses
- We should be able to generate any routine analysis *instantly and at zero (variable) cost and translate result into executable decisions*
- Analysis results must be presented in the context of executable decisions

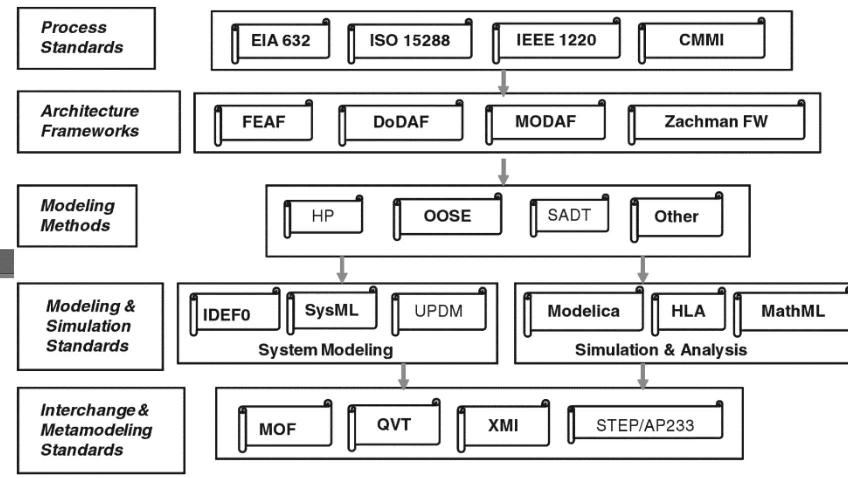
SO HOW SHOULD WE CREATE THESE "REFERENCE MODELS"?

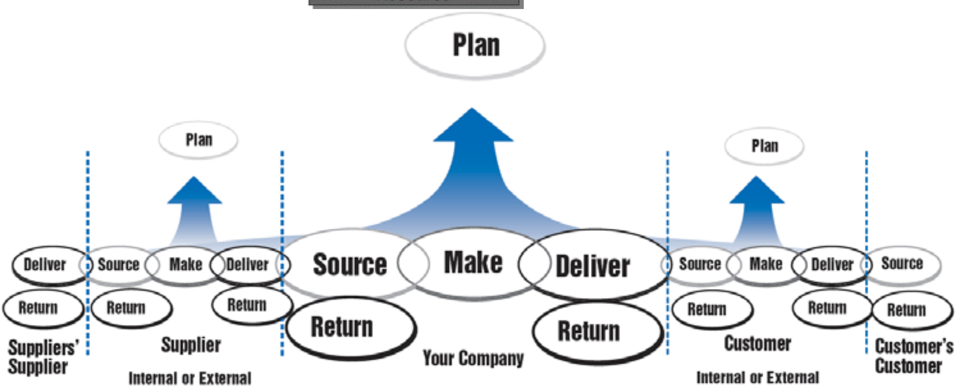
TWO FUNDAMENTAL QUESTIONS

- What tools should we (can we) use?
- How should we use these tools?

We spent years searching for a perfect discrete event logistic system model:

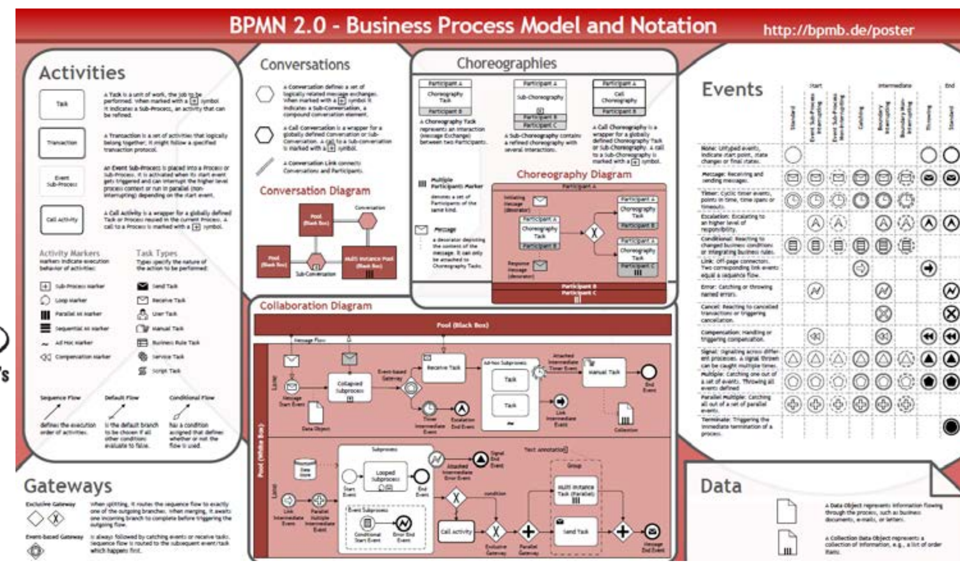






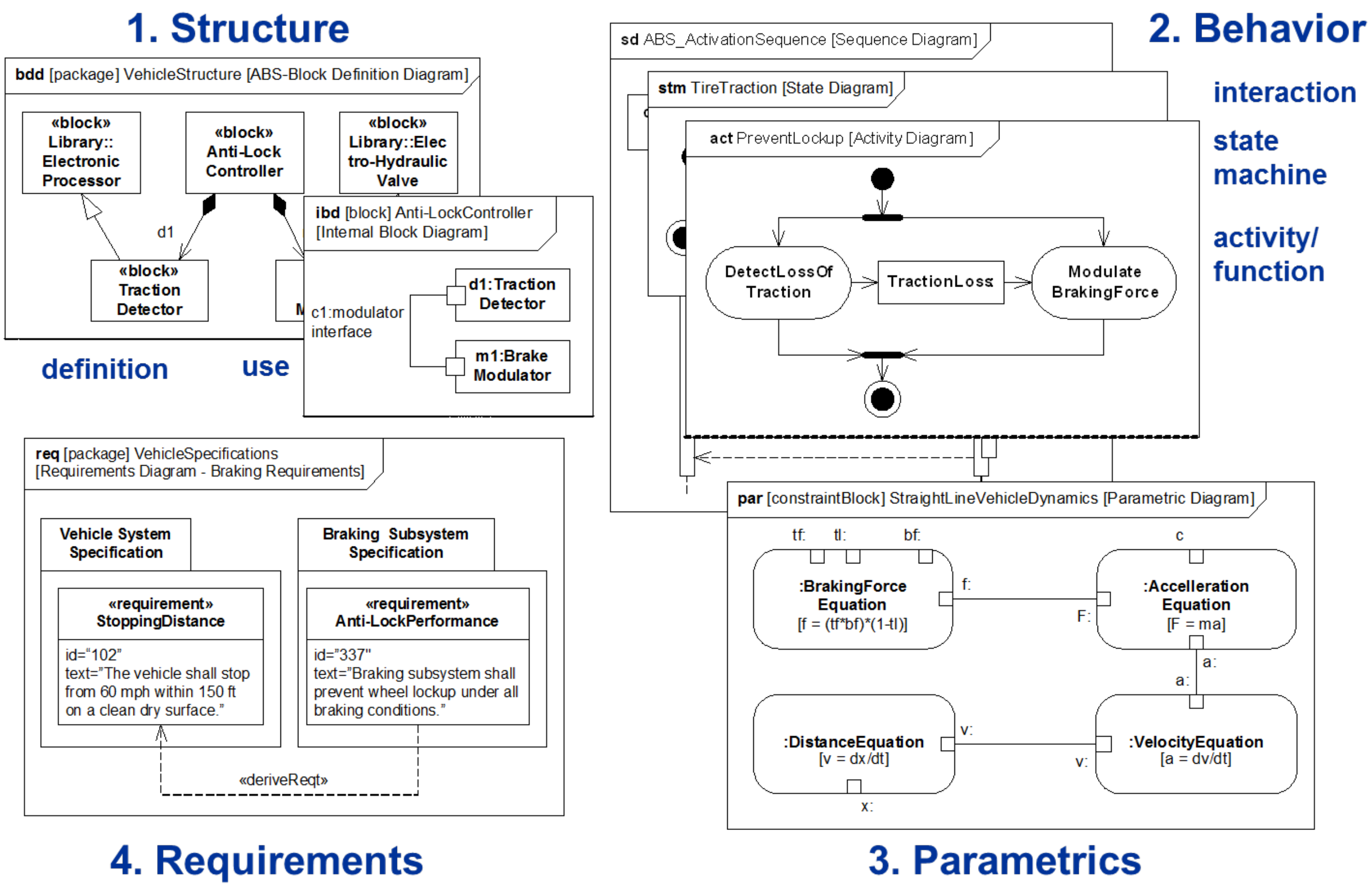
SCOR does not attempt to describe every business process or activity, including:

- Sales and marketing (demand generation)
- Research and technology development
- Product development
- Some elements of post-delivery customer support



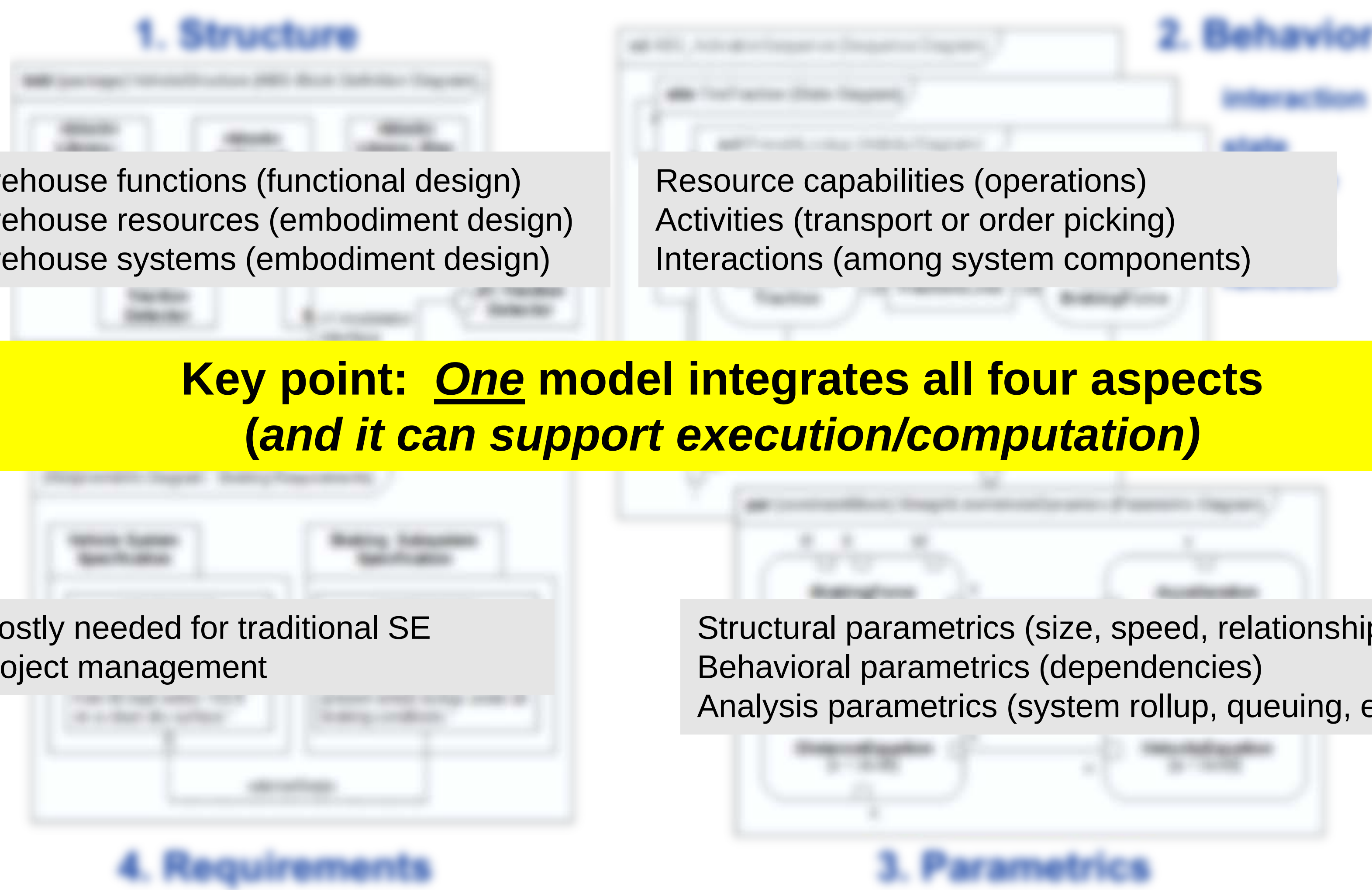
KEY ENABLER: SYSML

OMG SysML™: Systems Modeling Language



FOR EXAMPLE

Georgia Tech



Warehouse functions (functional design)
Warehouse resources (embodiment design)
Warehouse systems (embodiment design)

Resource capabilities (operations)
Activities (transport or order picking)
Interactions (among system components)

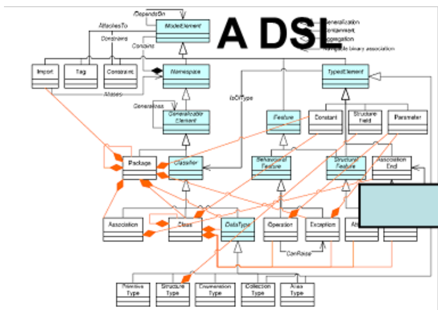
Key point: *<u>One</u>* model integrates all four aspects
(and it can support execution/computation)

Mostly needed for traditional SE project management

Structural parametrics (size, speed, relationships)
Behavioral parametrics (dependencies)
Analysis parametrics (system rollup, queuing, etc)

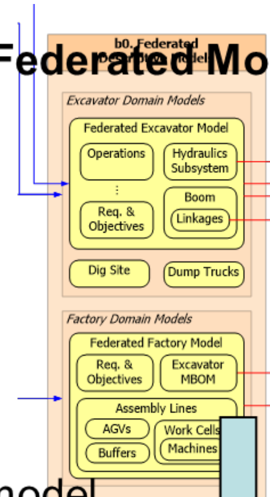
THE BASIC IDEA

A DSL



Use DSL to create “federated” model of a problem of interest in the domain (user model)

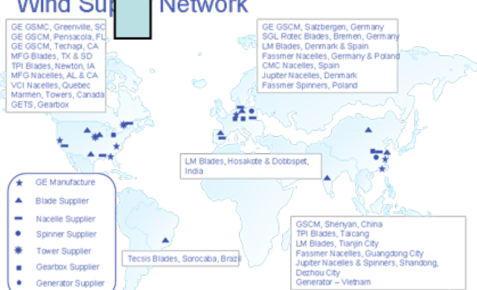
Federated Model



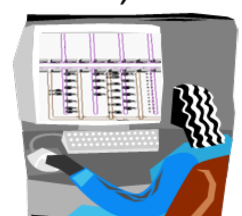
Use SysML to create a domain specific language (meta model)

Use model transformation to generate decision support models (instance model)

Wind Supply Network

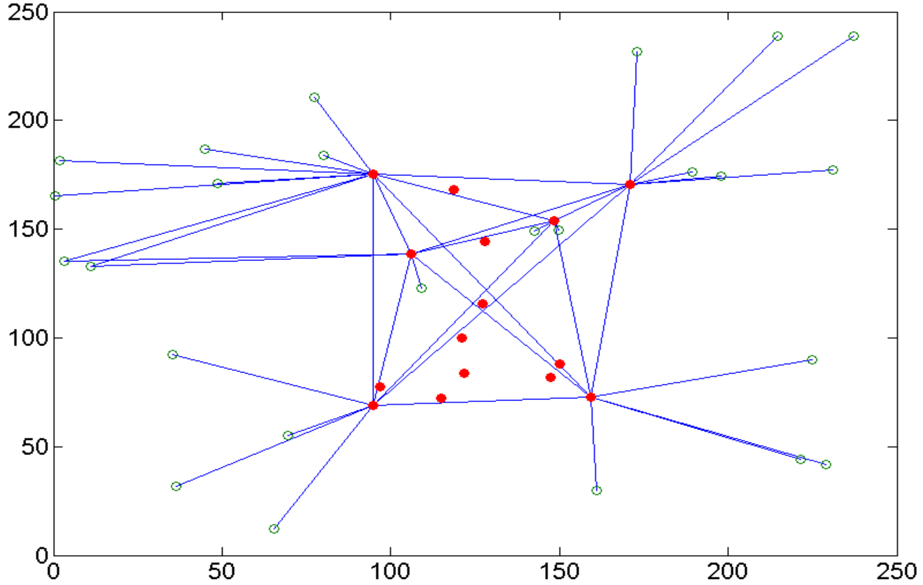


A Domain



To support stakeholder decisions in the domain

A USE CASE: SC DESIGN



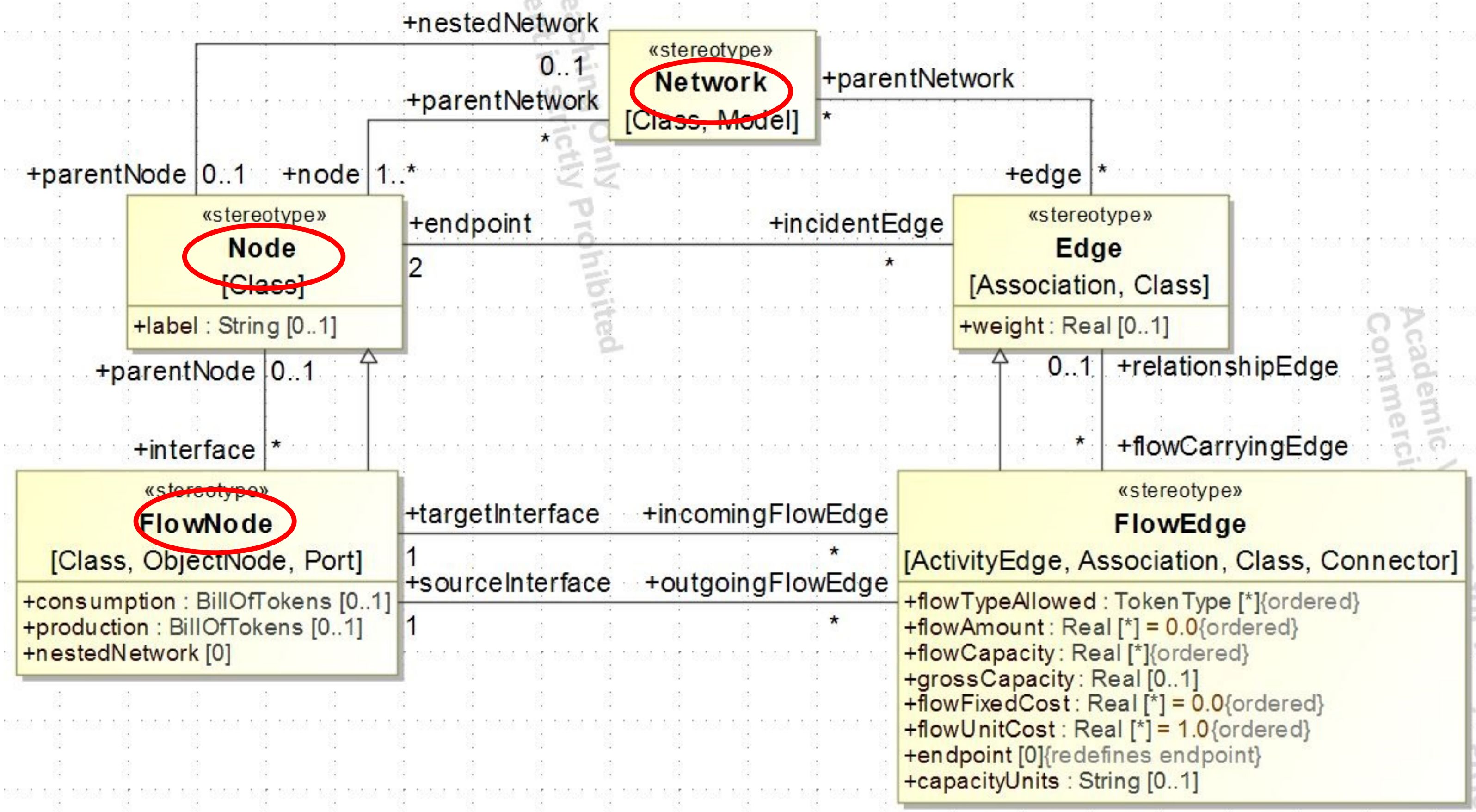
- Many locations where loads originate or terminate
- Many possibilities for distribution center locations
- Many possibilities for fleet configuration at each DC
- Want to guarantee delivery lead time
- Uncertain pickup/drop rates at each customer

If you care about *both cost and service level*, how many DCs should you have, where should they be, how should you configure each DC's vehicle fleet, and how should you dispatch vehicles?

Not just an optimization problem, because of control and uncertainty.

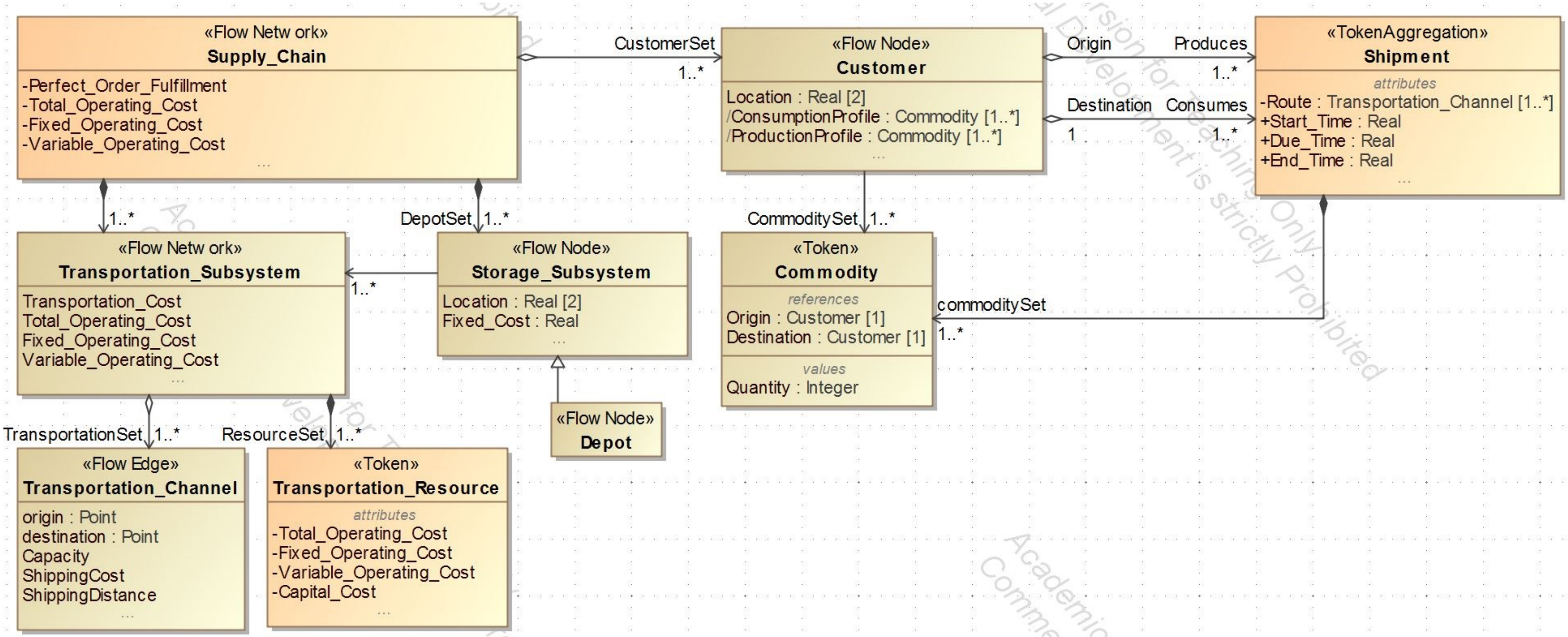
Not just a simulation problem, because of facility and fleet configuration decisions.

NETWORK META MODEL



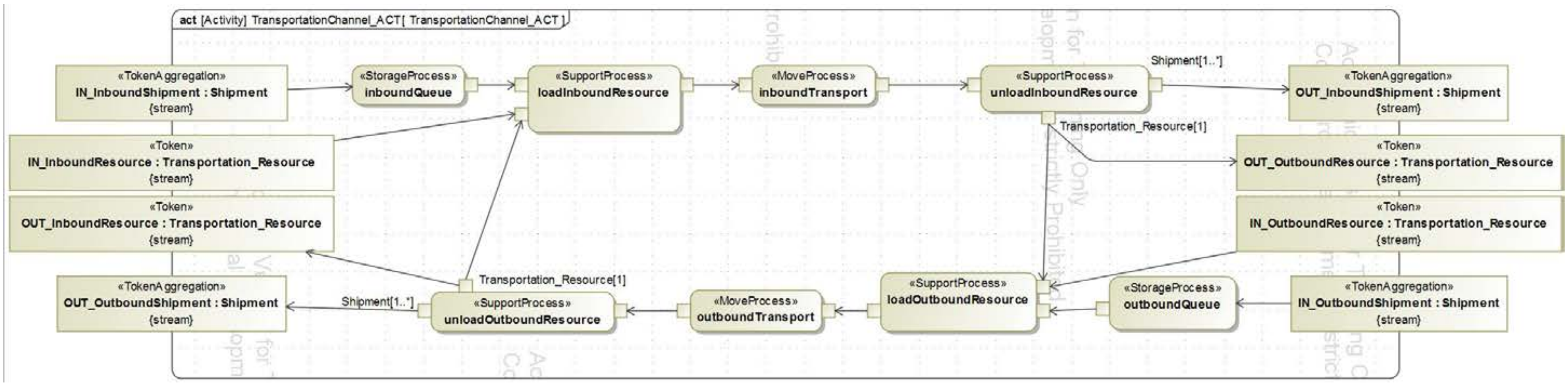
An example of a “meta-model” defining the semantics for creating an instance model of a particular (abstract) network.

SC META MODEL ELEMENTS



Using the meta-model concepts (e.g., <<Flow Network>>, <<Flow Edge>>, etc.) to develop a “domain specific language”, with semantics that are easily understood by the domain experts and stakeholders

TRANSPORT CHANNEL BEHAVIOR



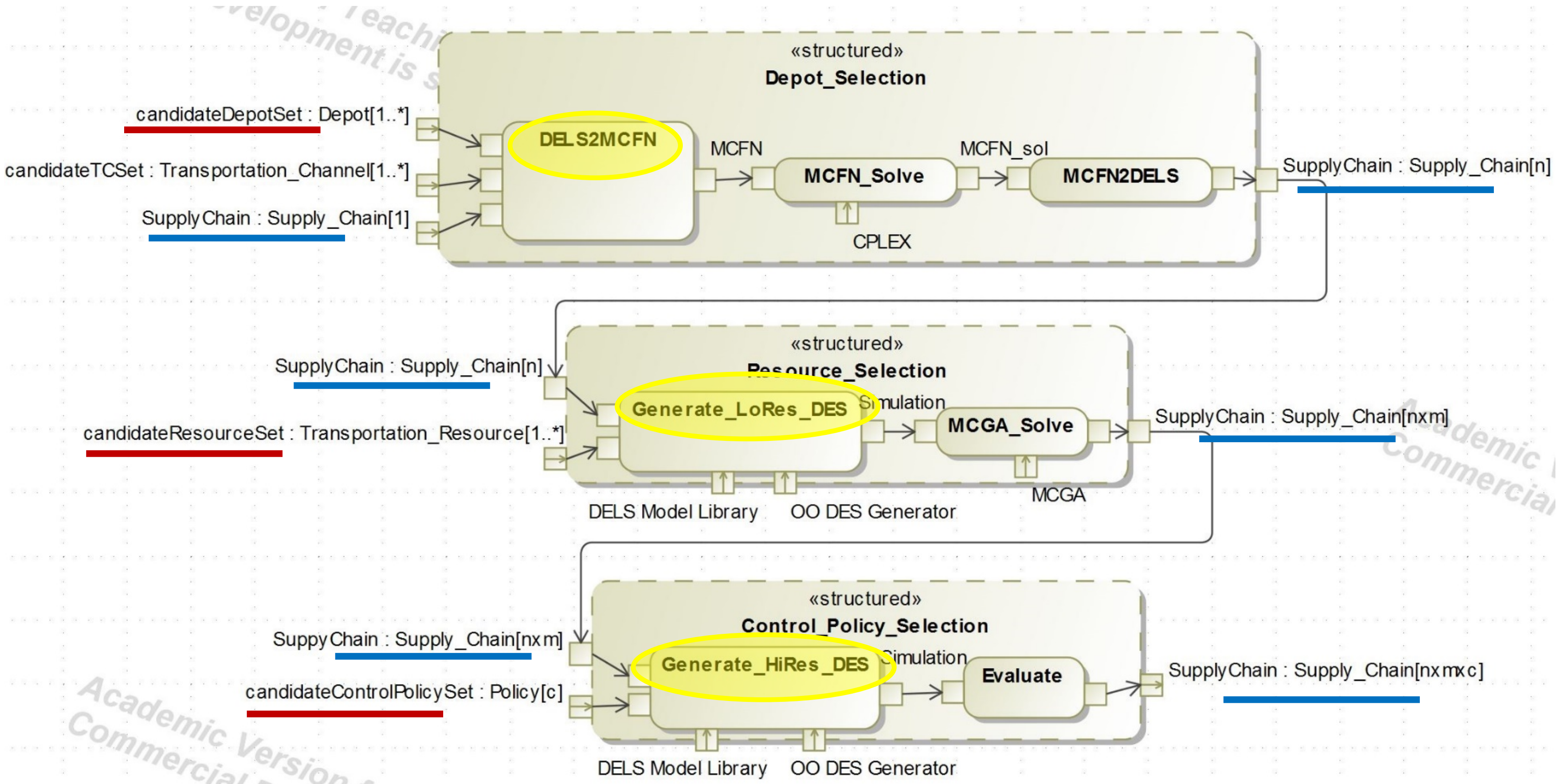
For this to work, we have to be precise—the system instance model cannot be ambiguous, because that will prevent reliable transformation to analysis models.

SC "OBJECT" REFERENCE MODEL

- Includes slots for source-sink flow network
- Includes slots for transportation network
- Includes slots for depots, fleets, and vehicle dispatch control

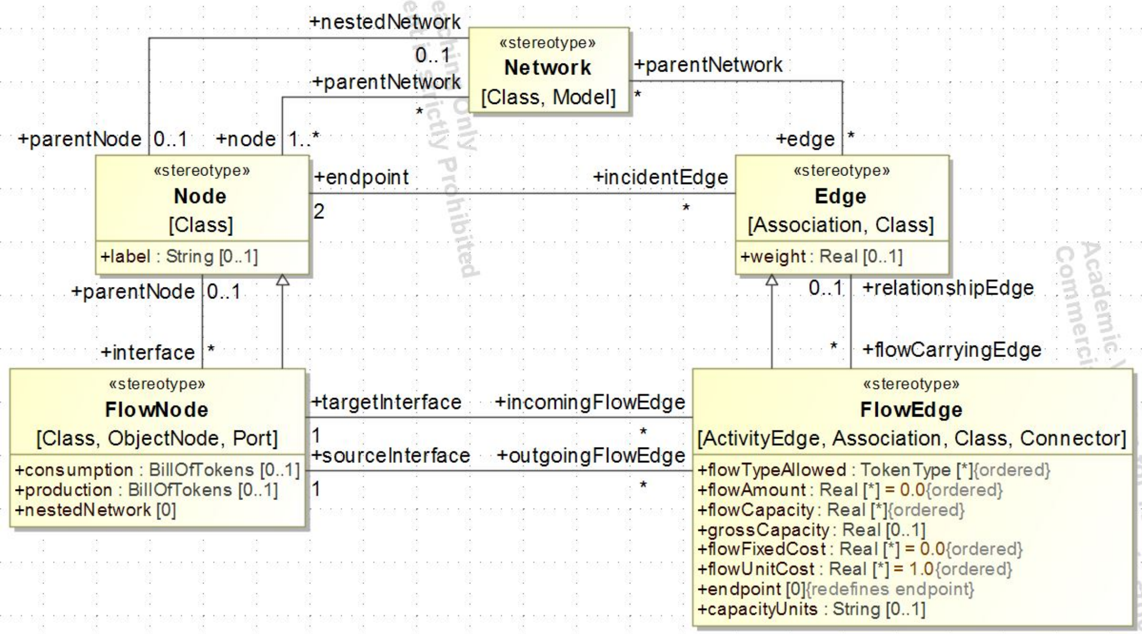
- Create an "instance" of the supply chain "object" which contains all the information you have for a particular supply chain design.

HIERARCHICAL DESIGN ANALYSIS



Each analysis "conforms" to the supply chain reference model, thus works for any "instance" of the supply chain object.

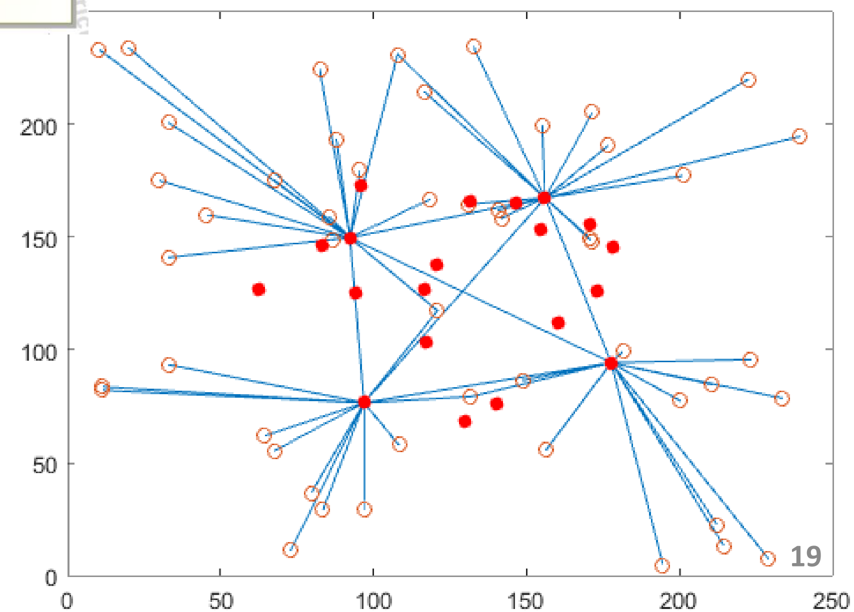
STRUCTURE: DEPOT SELECTION VIA MCFN



Goal: Reduce the computational requirements of optimizing the distribution network structure.

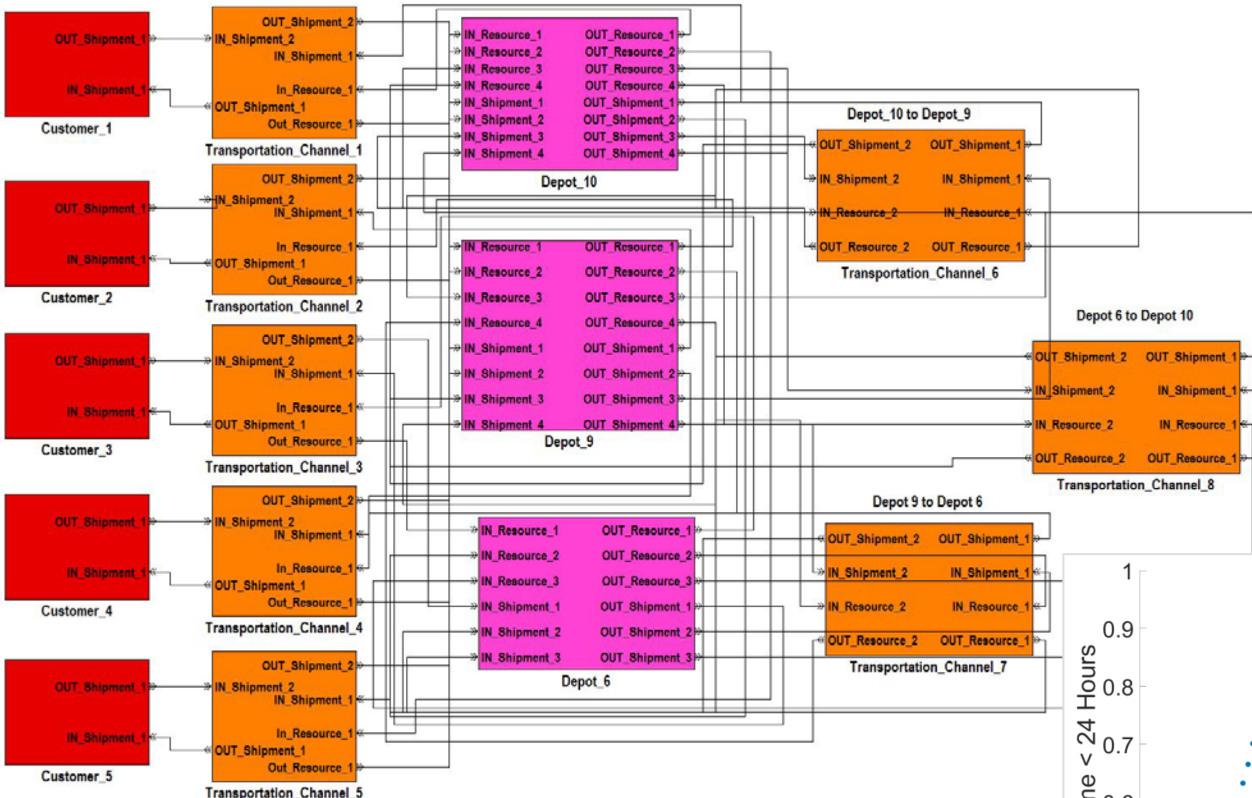
Strategy: Formulate and solve a corresponding multi-commodity flow network and facility location problem.

- Aggregate and approximate the flows and costs
- Solve MCFN using a COTS solver (CPLEX)
- Apply a “leave one out” strategy to generating several feasible candidate network structures.
- In this case, generate 5 candidates



BEHAVIOR: RESOURCE SELECTION

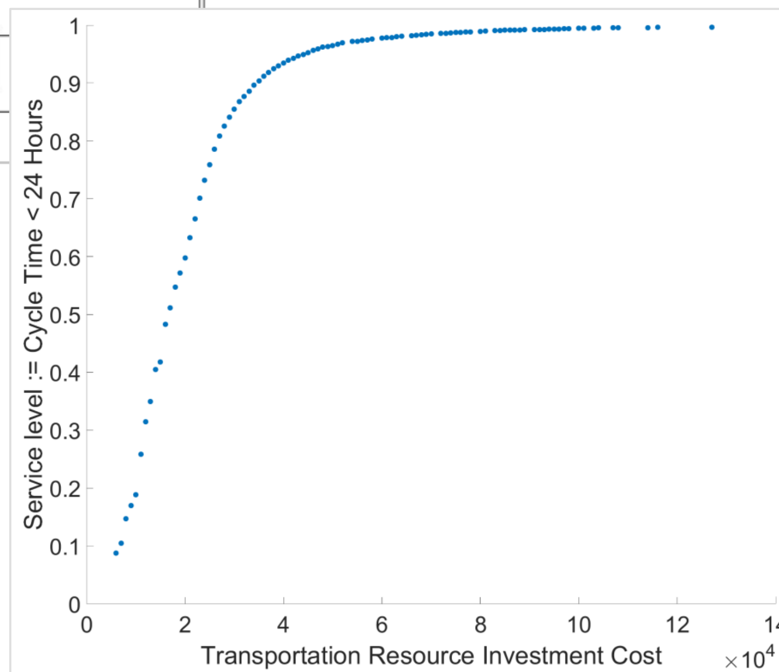
Georgia Tech



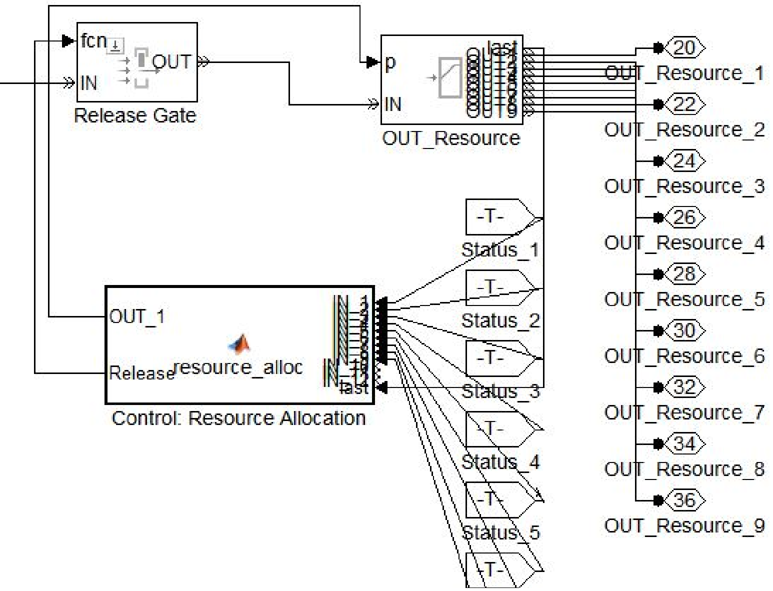
- For each candidate supply chain network structure, generate a portfolio of solutions to the fleet sizing problem
- Trade-off cycle time/service level and resource investment cost

Goal: Capture and evaluate the behavioral aspects of the system using discrete event simulation.

Strategy: Generate a DES that simulates a probabilistic flow of commodities through the system.



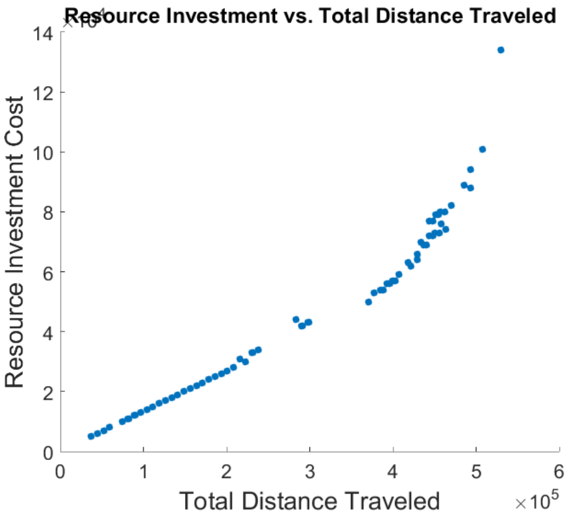
CONTROL: RESOURCE ASSIGNMENT

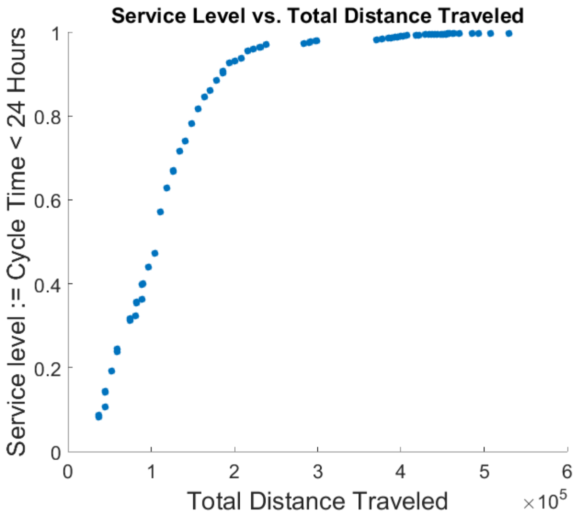


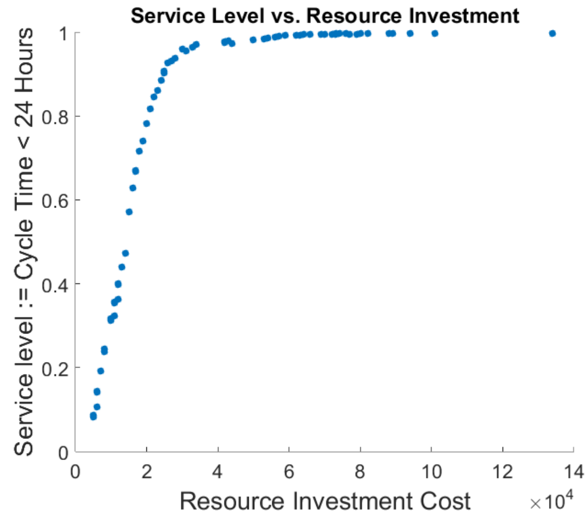
Goal: Select and design a detailed specification of the control policies for assigning trucks to pickup/dropoff tasks at customers.

Strategy: Generate a high-fidelity simulation that is detailed enough to fine-tune resource and control behavior.

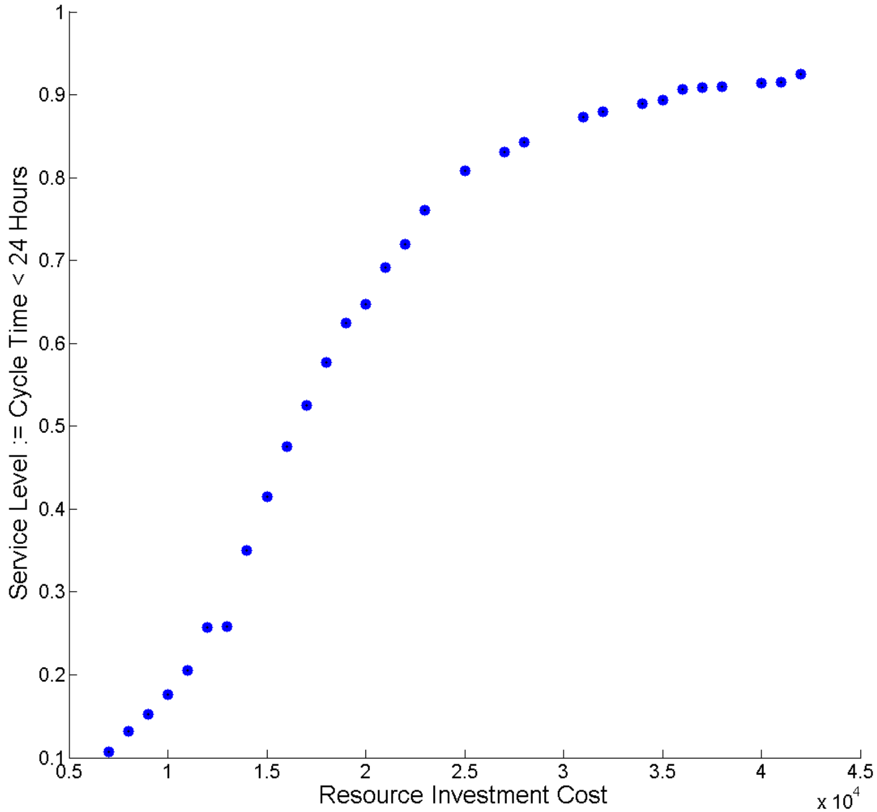
Generate a Pareto set of solutions that trade-off Service Level, Capital Costs, and Travel Distance





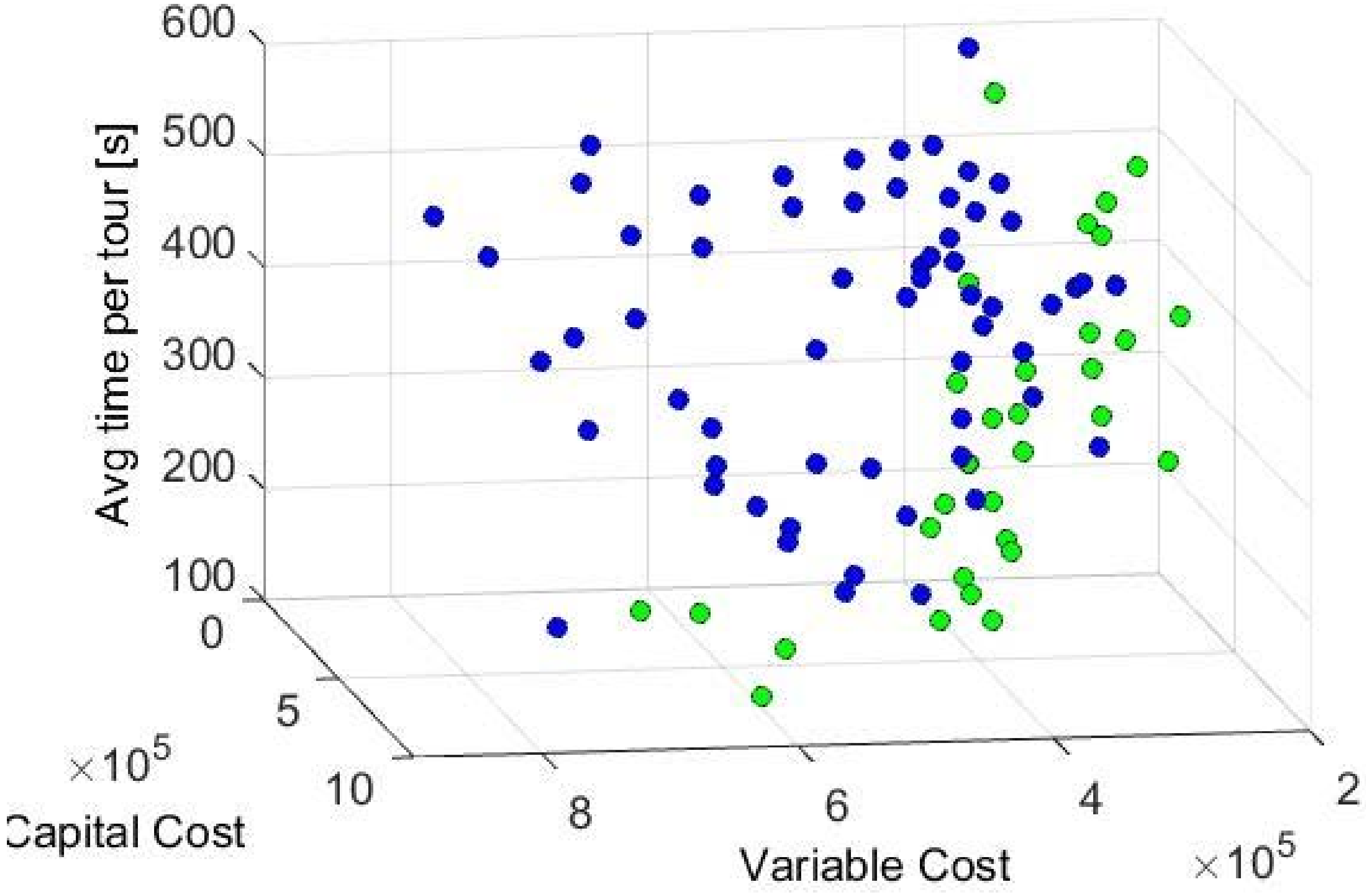


KINDS OF RESULTS



- These are Pareto optimal designs
- Decision makers make trade-offs
- Hundreds, perhaps thousands of simulation runs, with varying depot location decisions, varying fleet configurations, varying control policies—all generated algorithmically

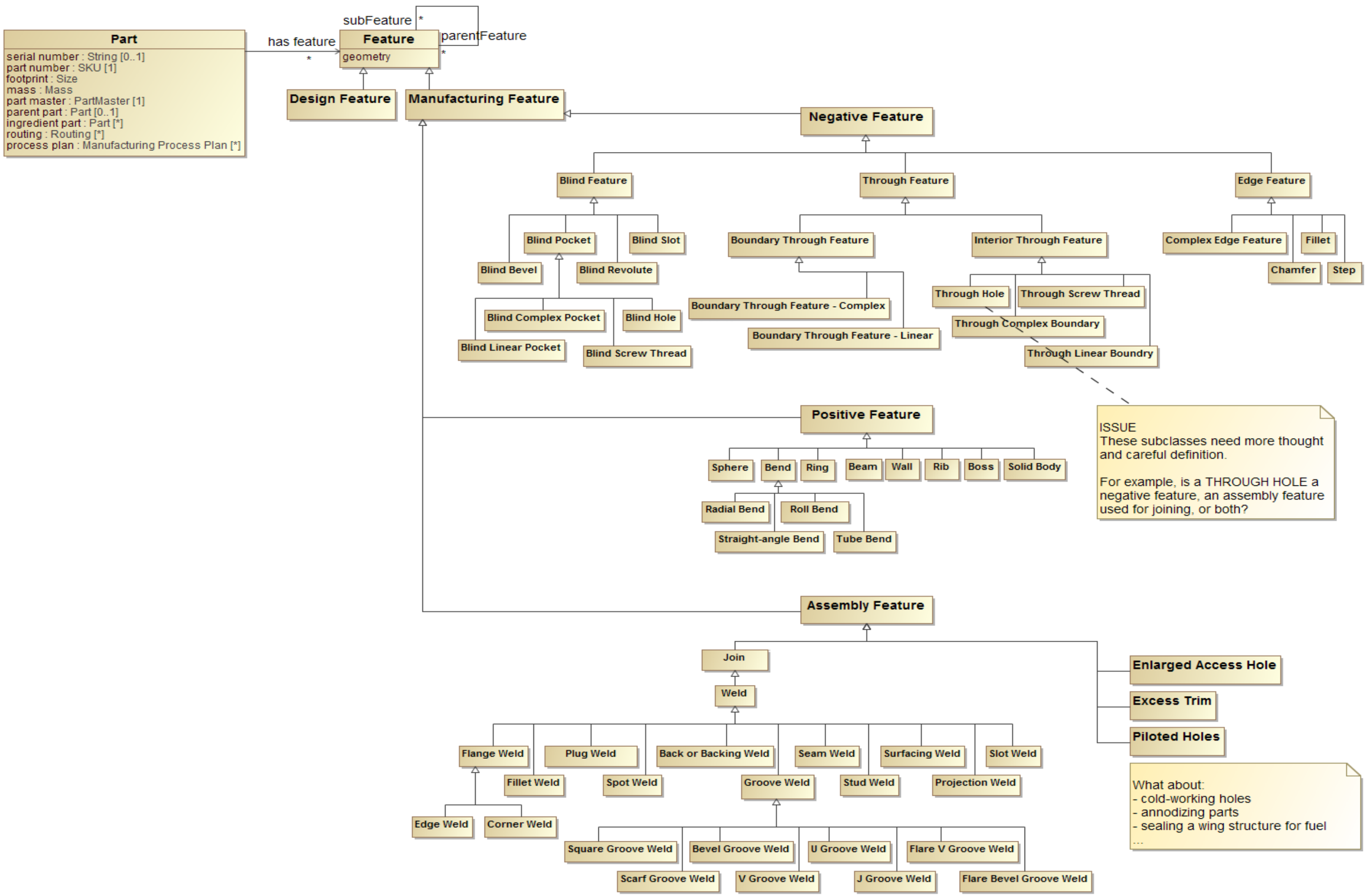
VISUALIZATION CHALLENGES



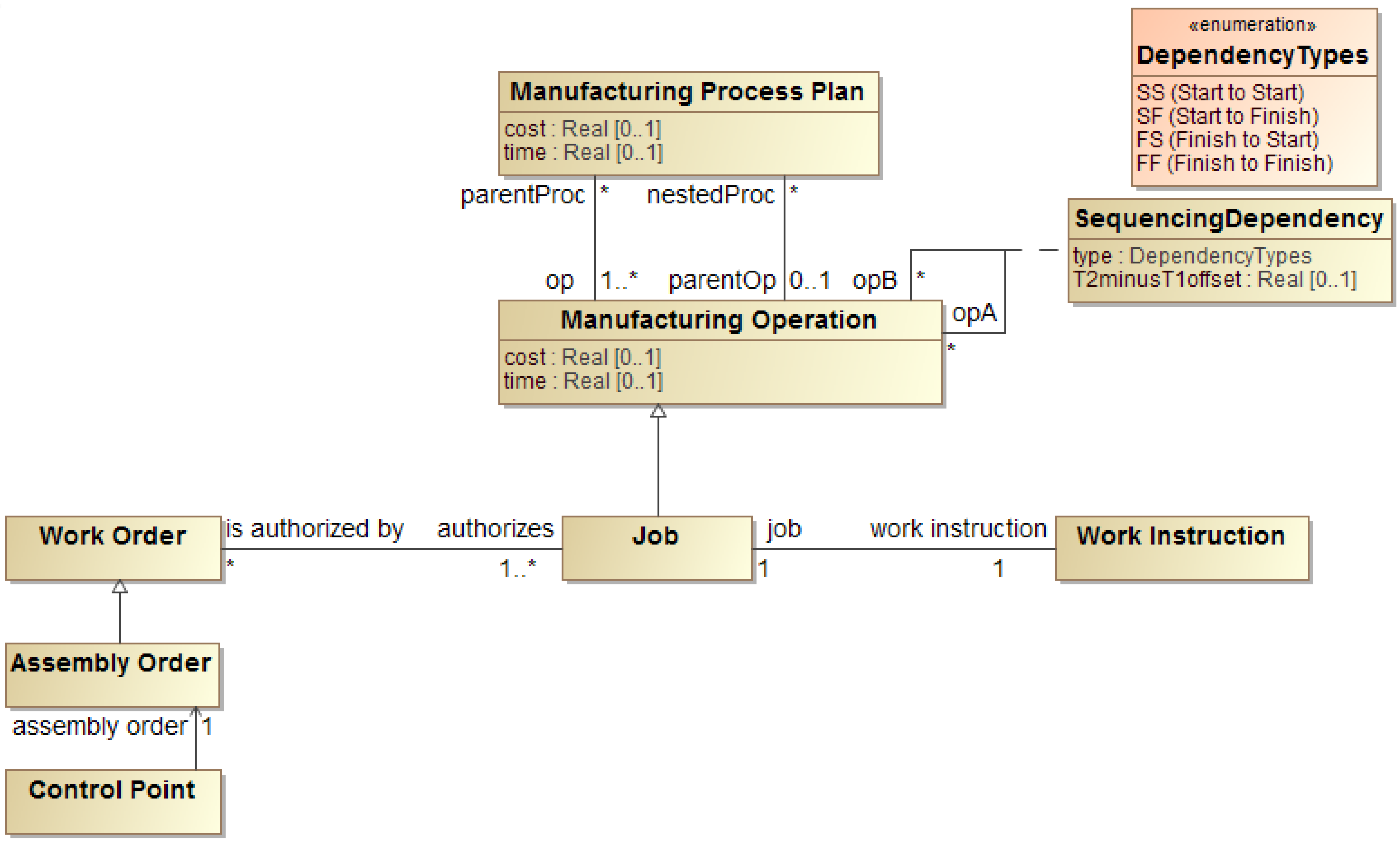
WHAT IF?

- You want to look inside a node and evaluate in more detail how it will perform, i.e., you want to model its production processes?
- Flow nodes can <u>*nest*</u> a flow network
- Need additional semantics
  - Underlying *network* structures
  - Semantics for *product, process, resource, facility*
  - Semantics for *control*

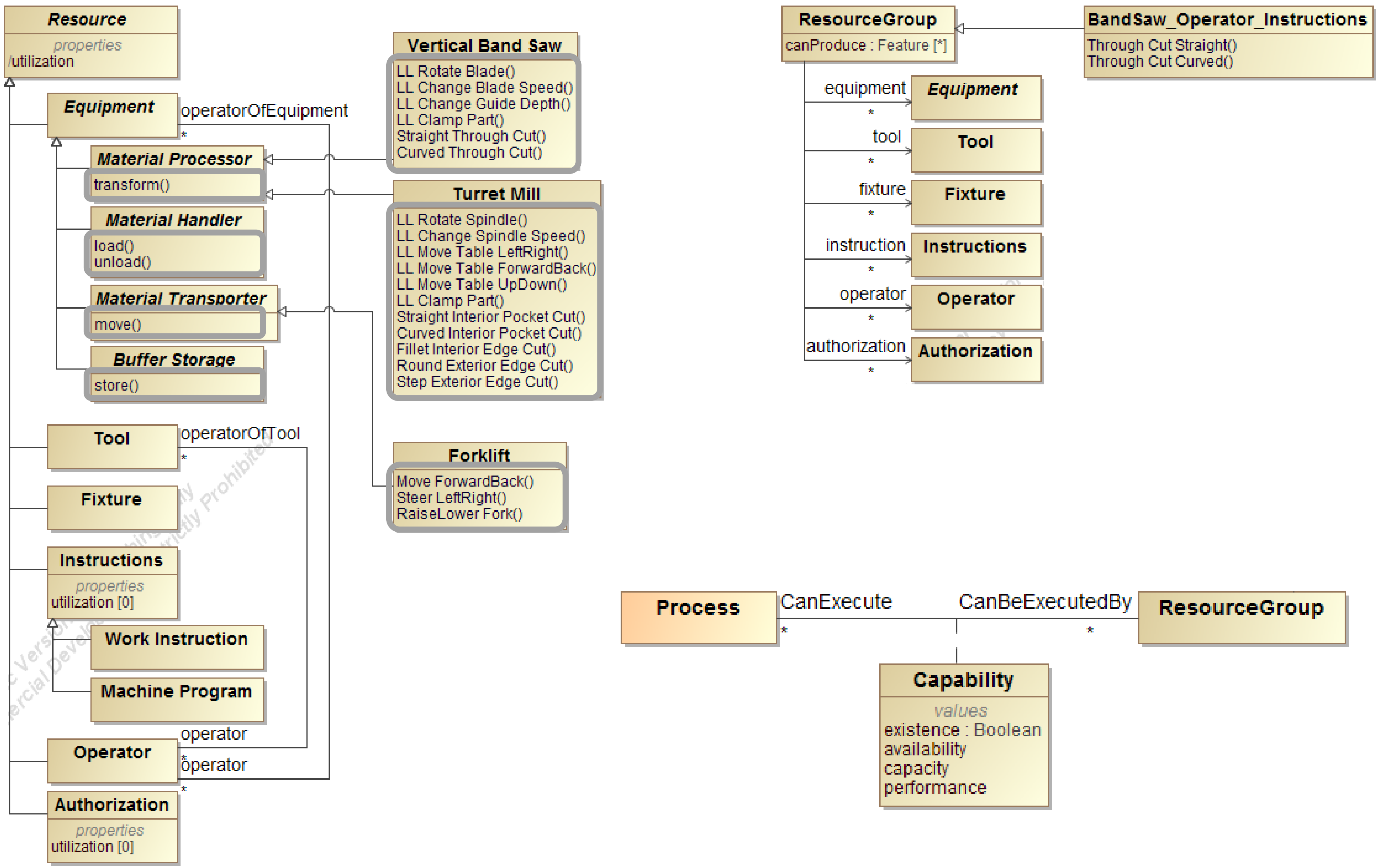
DEFINE “PRODUCT”



DEFINE "PROCESS"

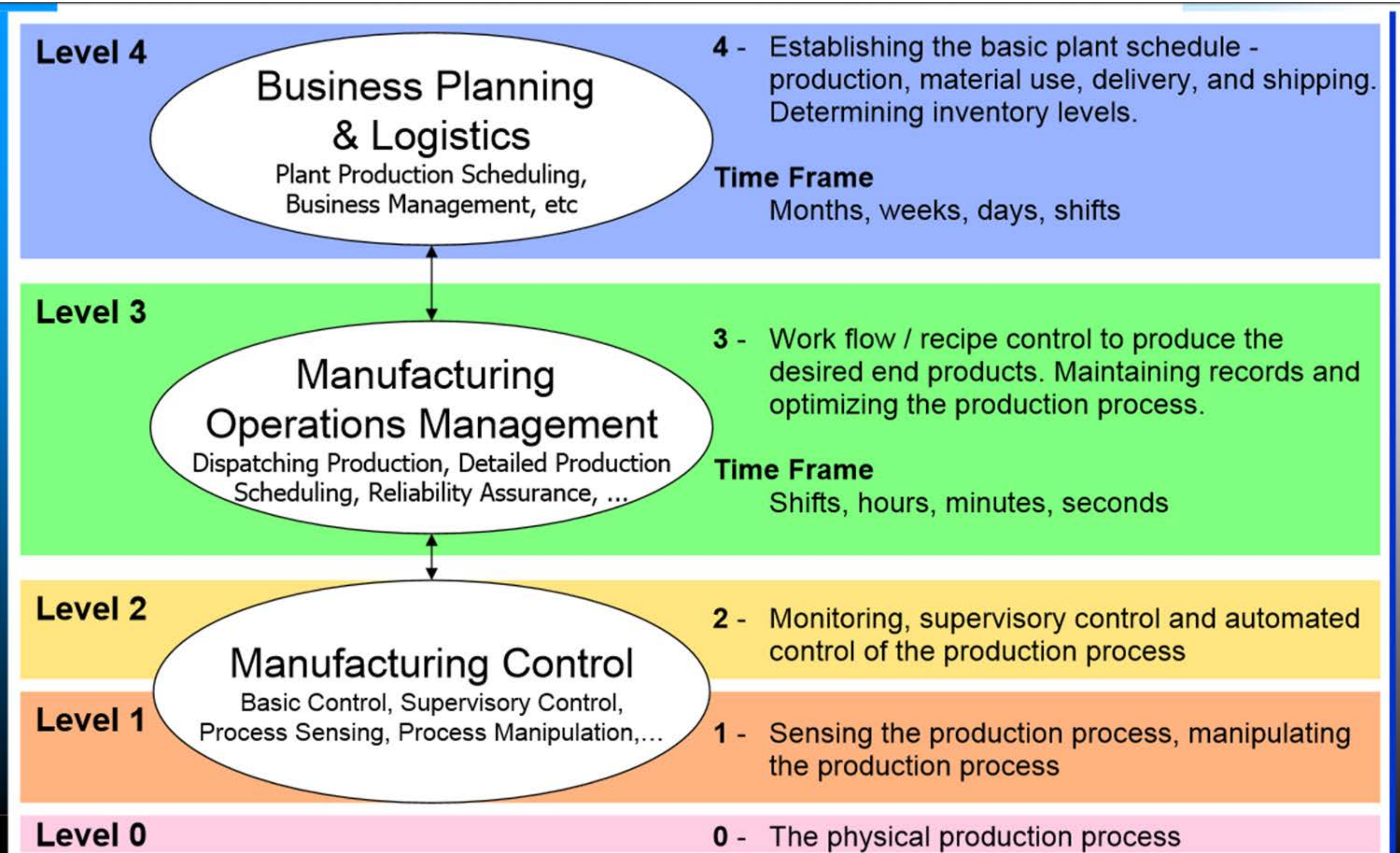


DEFINE "RESOURCE"



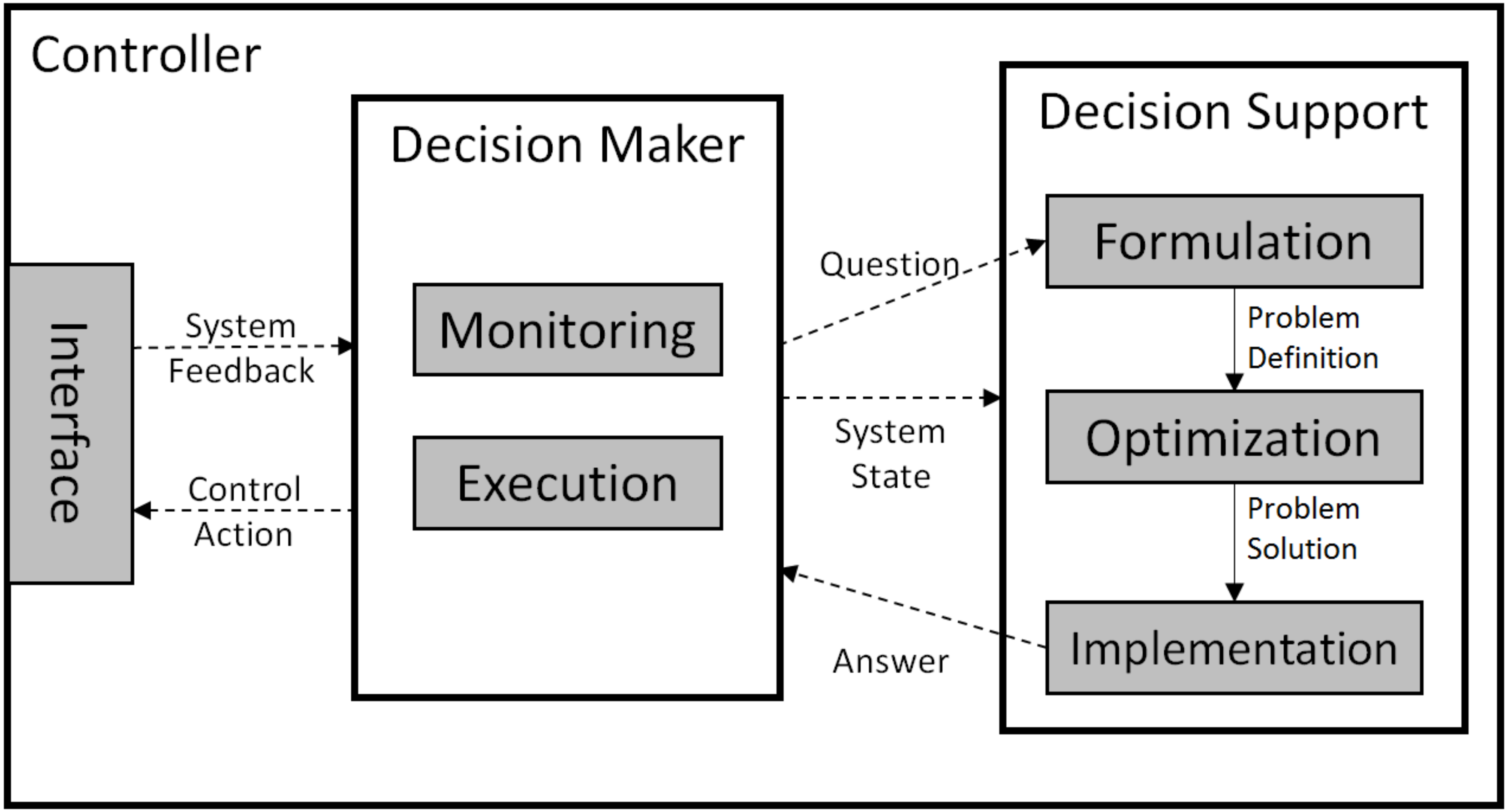
DEFINE "CONTROL"

Georgia Tech



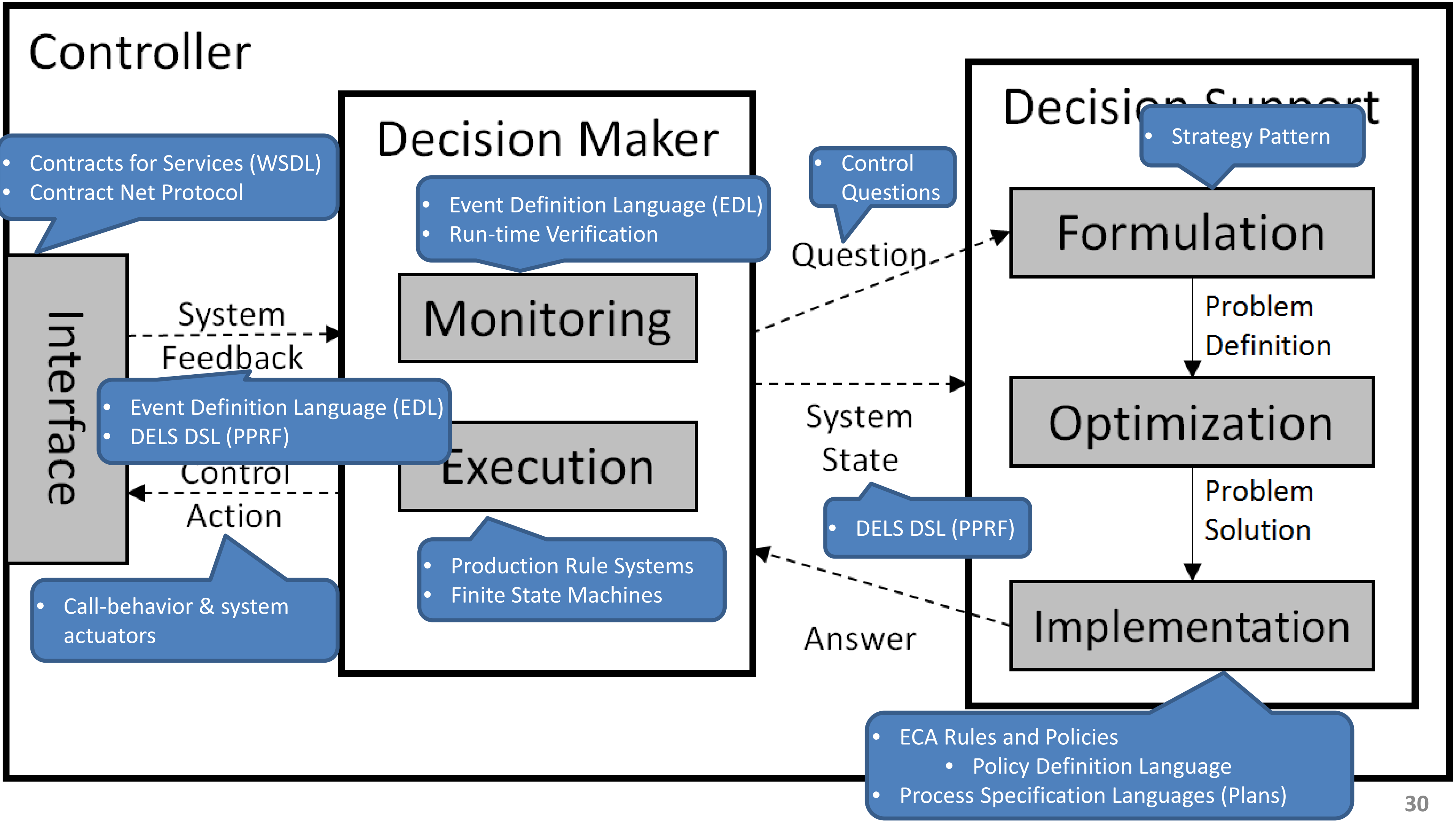
FUNCTIONAL SPECIFICATION OF DELS CONTROLLER

If the ISA-95\L3 architecture is going to be implementable, it needs to be generic.



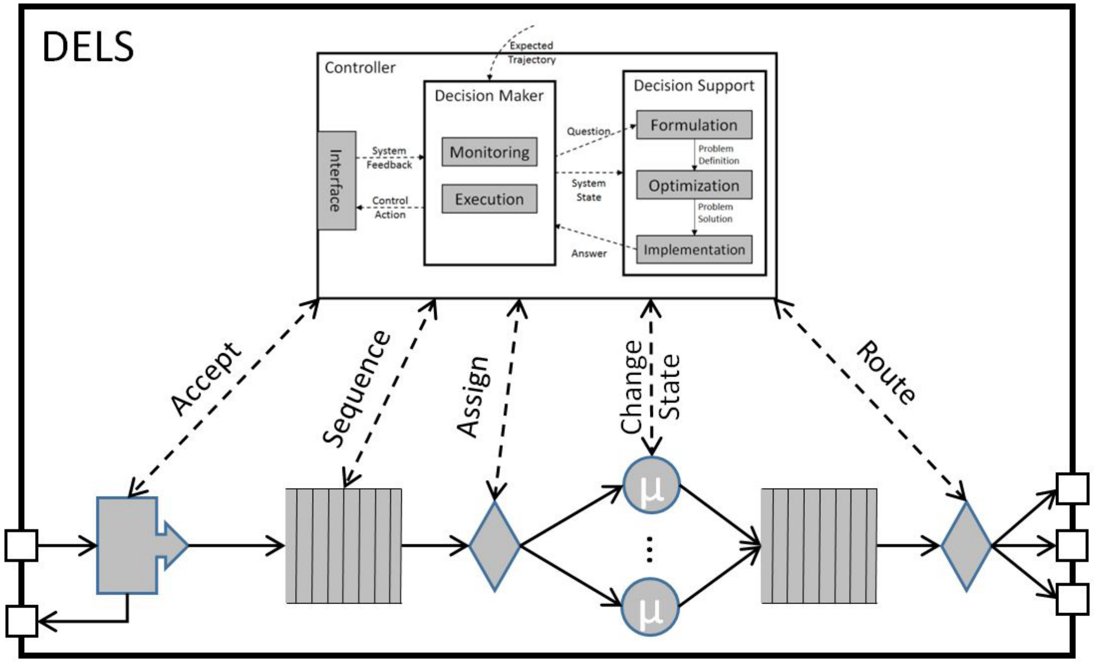
METAMODEL OF OPERATIONAL CONTROL

This research lives at the interfaces with many other disciplines, and it cannot be done without integrating ideas from all of these communities: IE, OR, SysE, SwE, CS.



CONTROL QUESTIONS

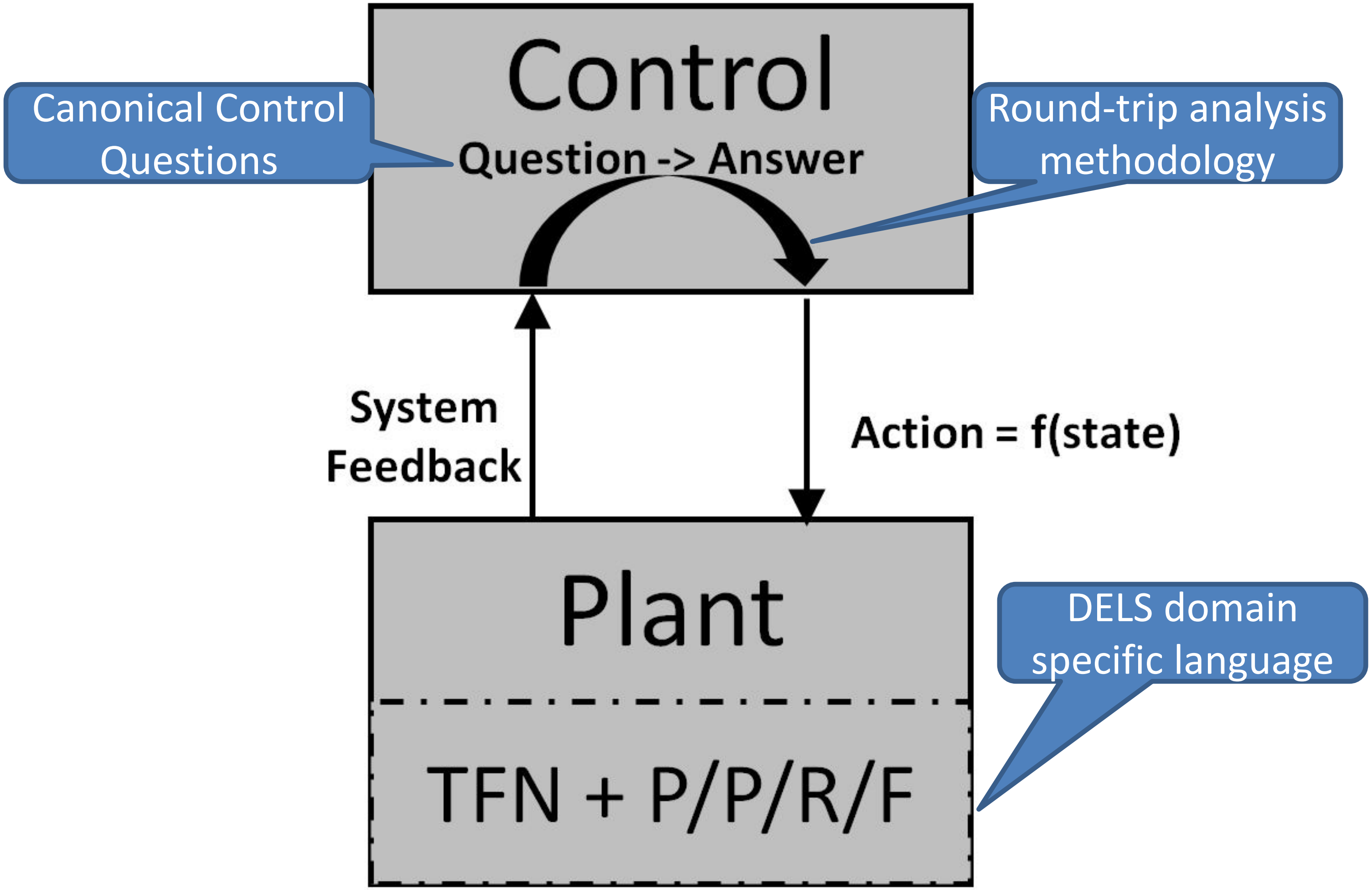
Control questions provide a mapping from a formal functional definition of control activities for DELS to formal (math programming) analysis models.



- Which tasks get serviced? (Admission/Induction)
- When {sequence, time} does a task get serviced? (Sequencing/Scheduling)
- Which resource services a task? (Assignment/Scheduling)
- Where does a task go after service? (Routing)
- What is the state of a resource? (task/services can it service/provide)

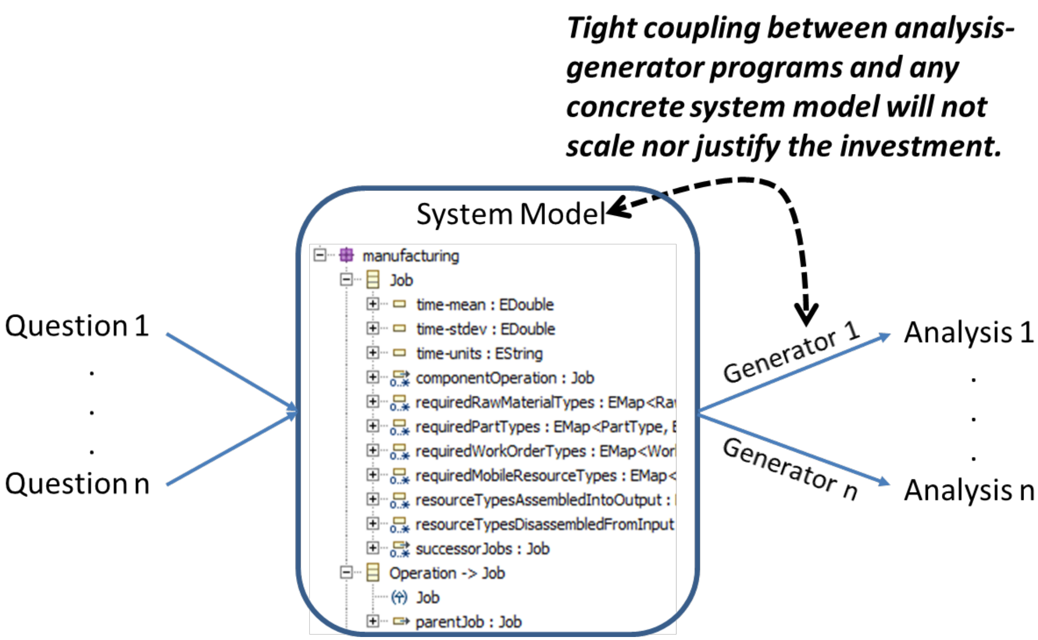
SEPARATION OF PLANT AND CONTROL

The prevailing paradigm in the literature neglects to separate the model of the plant from the model of the control of that plant:

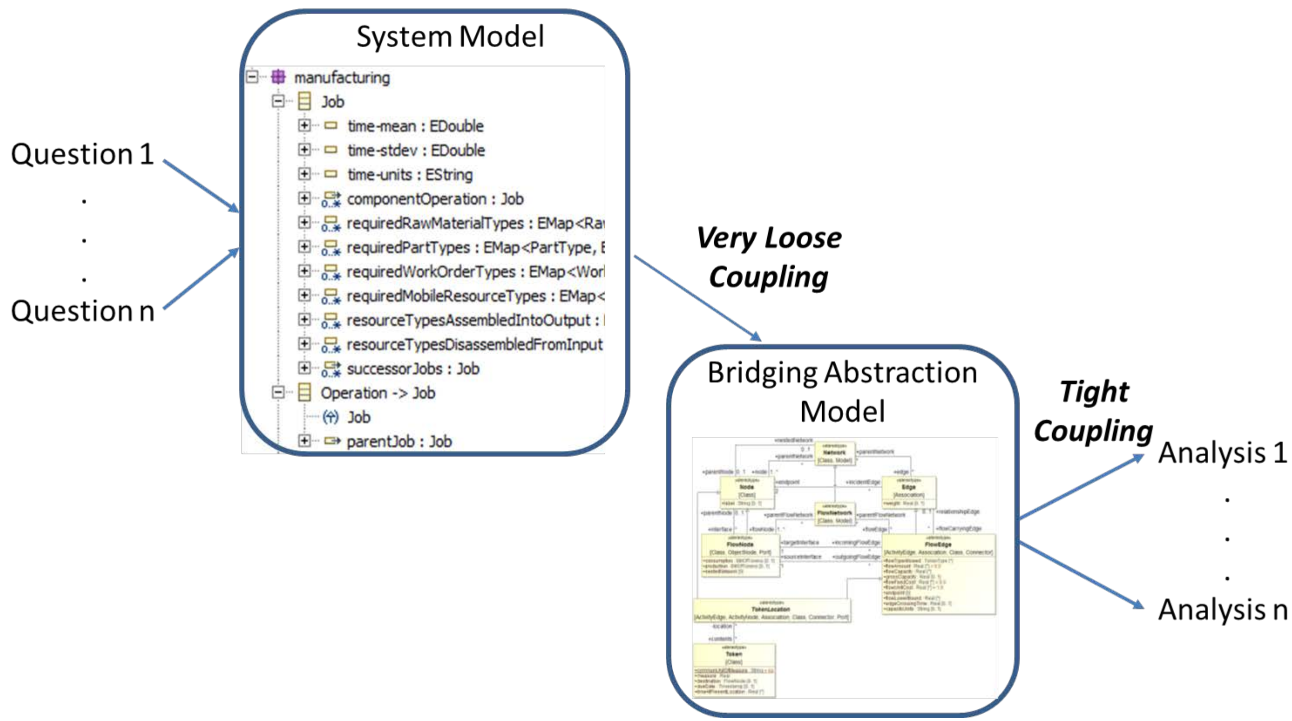


KEY LEARNING

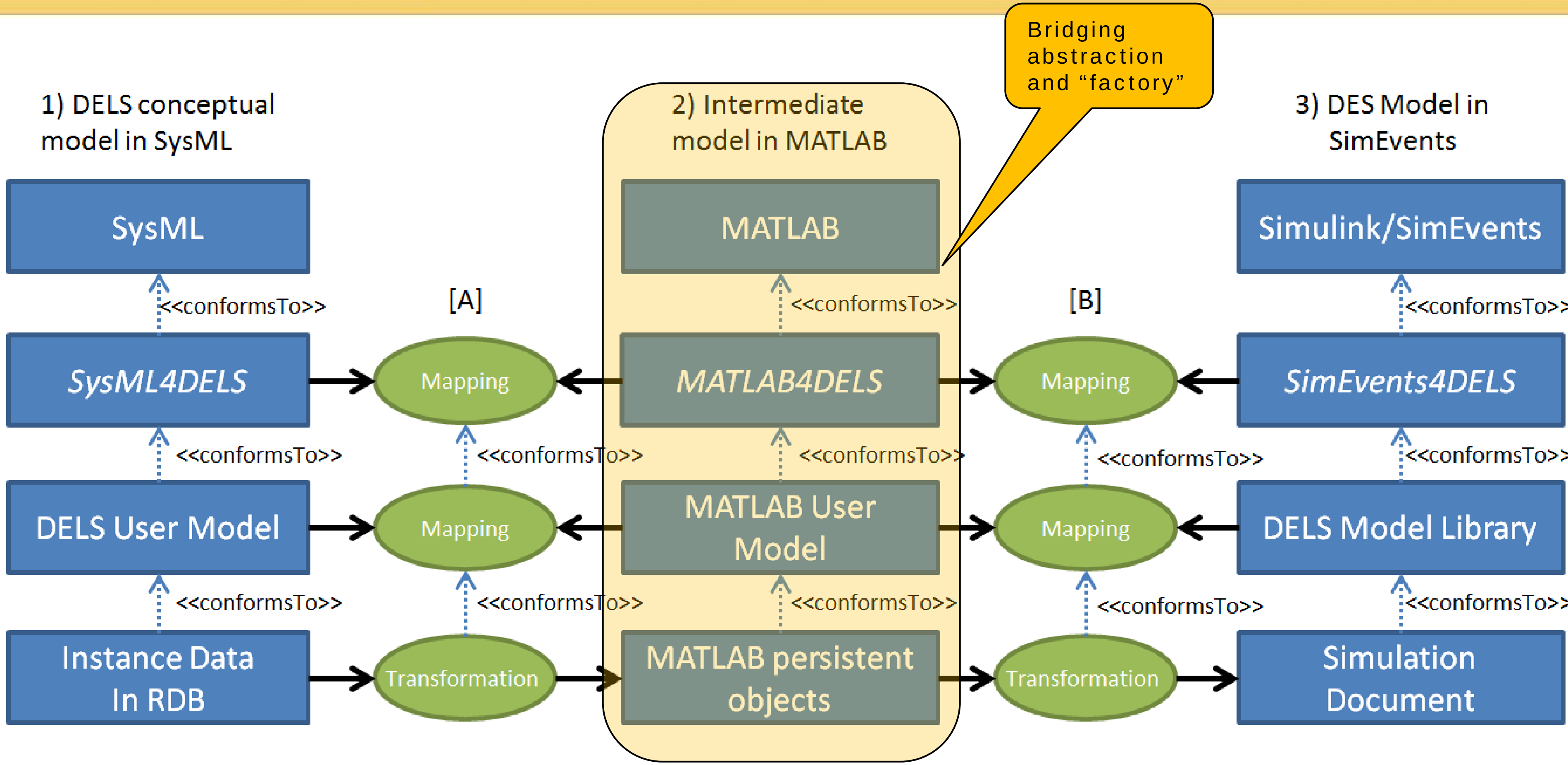
- Need "concrete" modeling for acceptance by domain stakeholders
- Need "abstract" modeling to support modeling automation
- A consequence of the need to be simultaneously abstract and concrete is that no perfect generic DELS model exists. Any simulation-generation strategy must accommodate a variety of system models, each of which may regularly change and evolve



We solve this problem by introducing a bridging abstraction model, one of our biggest innovations. It’s an abstract model capturing the underlying commonalities of all DELS, and is robust and stable enough for analysis-generator programs to rely on.



ONE IMPLEMENTATION



To accomplish the transformation seamlessly, we need three things:

1. Relational Database (and instance data) that conforms to Reference Architecture (SysML)
2. MATLAB class definitions (classdefs) that conform to Reference Architecture (SysML)
3. SimEvents Model Library objects that conform to Reference Architecture (SysML)

ARE WE THERE YET?

We need "standards" for a DELS reference model, or DSL

We need to elaborate the bridging abstraction so that it's complete and rigorous

We need a better discrete event simulation platform, because no COTS tool is up to the task of modeling & simulating control processes

BTW, we need more than simulation

We need a common s/w platform so that we can collaborate on achieving this vision (as you find in the optimization world)

We need to focus on "round-trip analysis"

Scott's right—we need test suites