Model Lifecycle Management

Dr. Michael Tiller
VP, Modeling R&D
Emmeskay, Inc.

Background

MBD: STAMP Required

- **S** – Skills
- **T** – Tools
- **A** – APIs
- **M** – Models
- **P** – Processes

Modelica

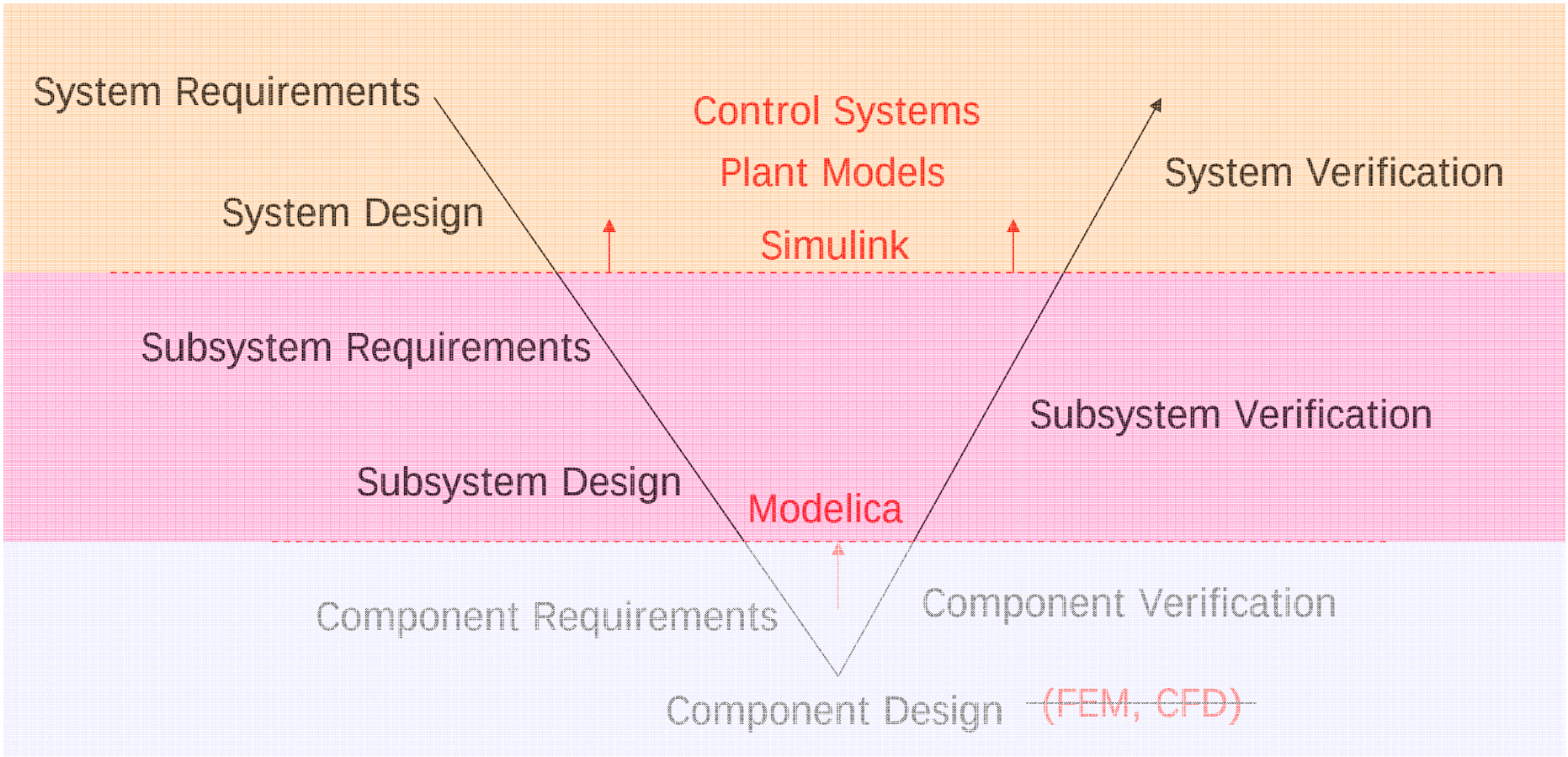
- Non-proprietary language for describing the continuous and discrete behavior of systems.
  - Equations
  - Components
  - Libraries
  - Architectures
- Multi-formalism, multi-domain approach.
- Compelling both technically and from a business perspective.

Model Lifecycle Management

Maximizing the impact and value
of modeling throughout the
product development process.

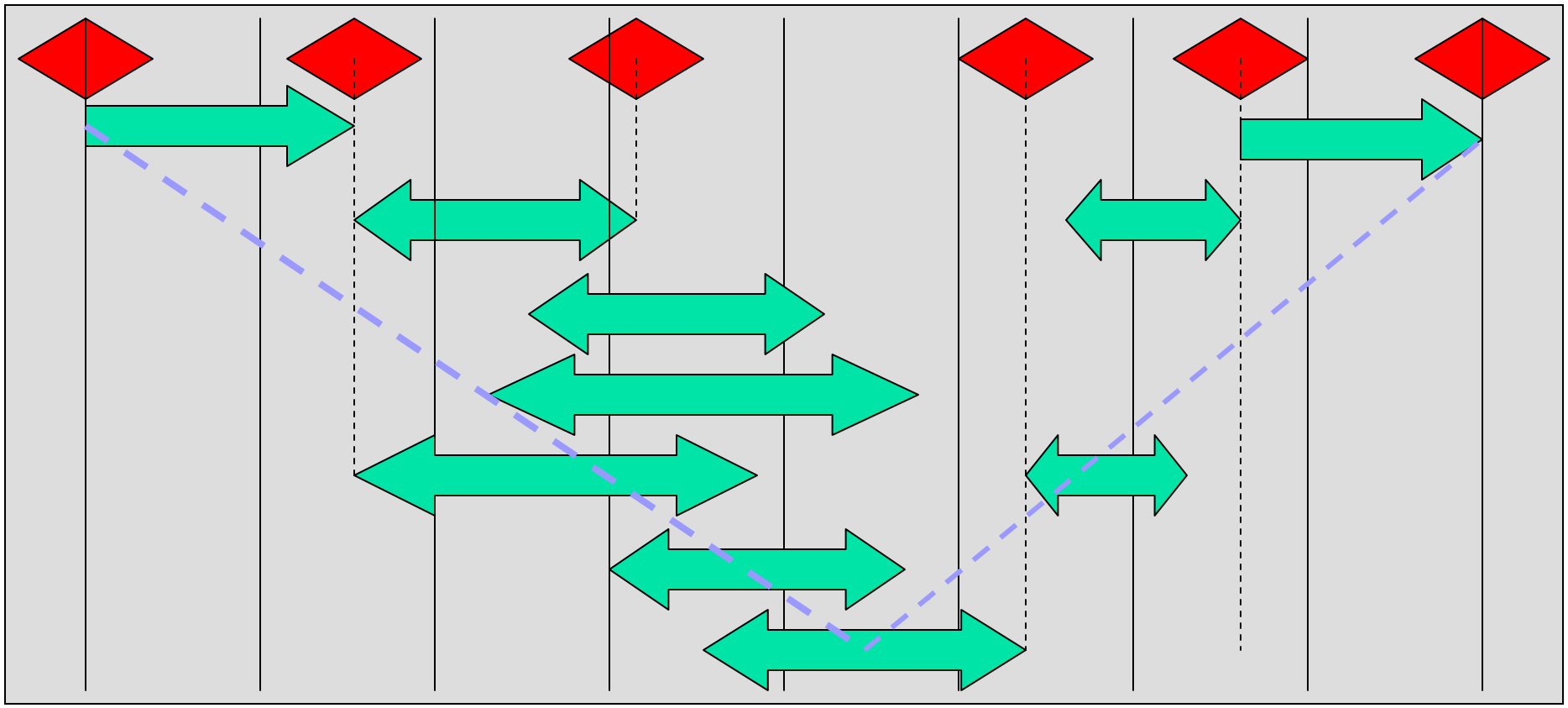
Kinds of Models

- Tools, Formalisms and System Engineering



Product Development

- Where does modeling fit in the product development process?



Model Lifecycles

Why management is needed…

“Mayfly” Lifecycle

- Created spontaneously
  - Minimal planning
  - Time pressure
  - Often by somebody without much experience.
- Limited validation but still used to make important decisions.
- Thrown away when done.
- Redundant and surprisingly common.

“Dinosaur” lifecycle

- Created by forward thinking organizations.
- Represents the cutting edge for some point in time.
- Languishes when vision and innovation fall out of favor.
- Devolves into extinction management.

“Tree of Life” Lifecycle

- Models are not centrally controlled.
- Users copy and modify models unfettered.
- Redundancy and chaos create a confusing landscape of options.

Technical Aspects

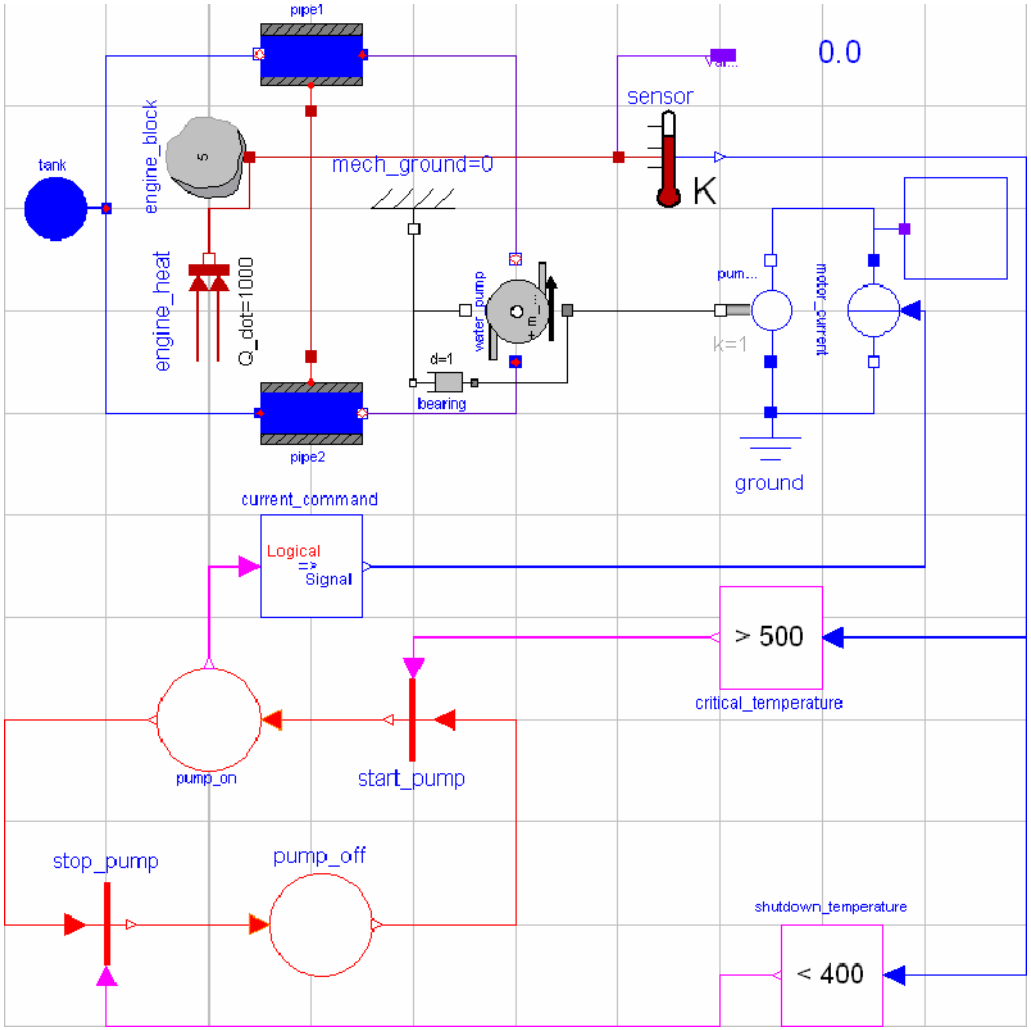
- Formalisms
- Reuse
- Configuration Management
- Version Control
- Testing and Validation

Formalisms

- Many types of modeling formalisms
  - Block diagrams
  - Bond-graphs
  - Acausal formalisms (a.k.a. Isomorphic, Schematic)
  - Petri nets/State charts
- Controls vs. Plant

"If the only tool you have is a hammer, you tend to see every problem as a nail."

-- Abraham Maslow

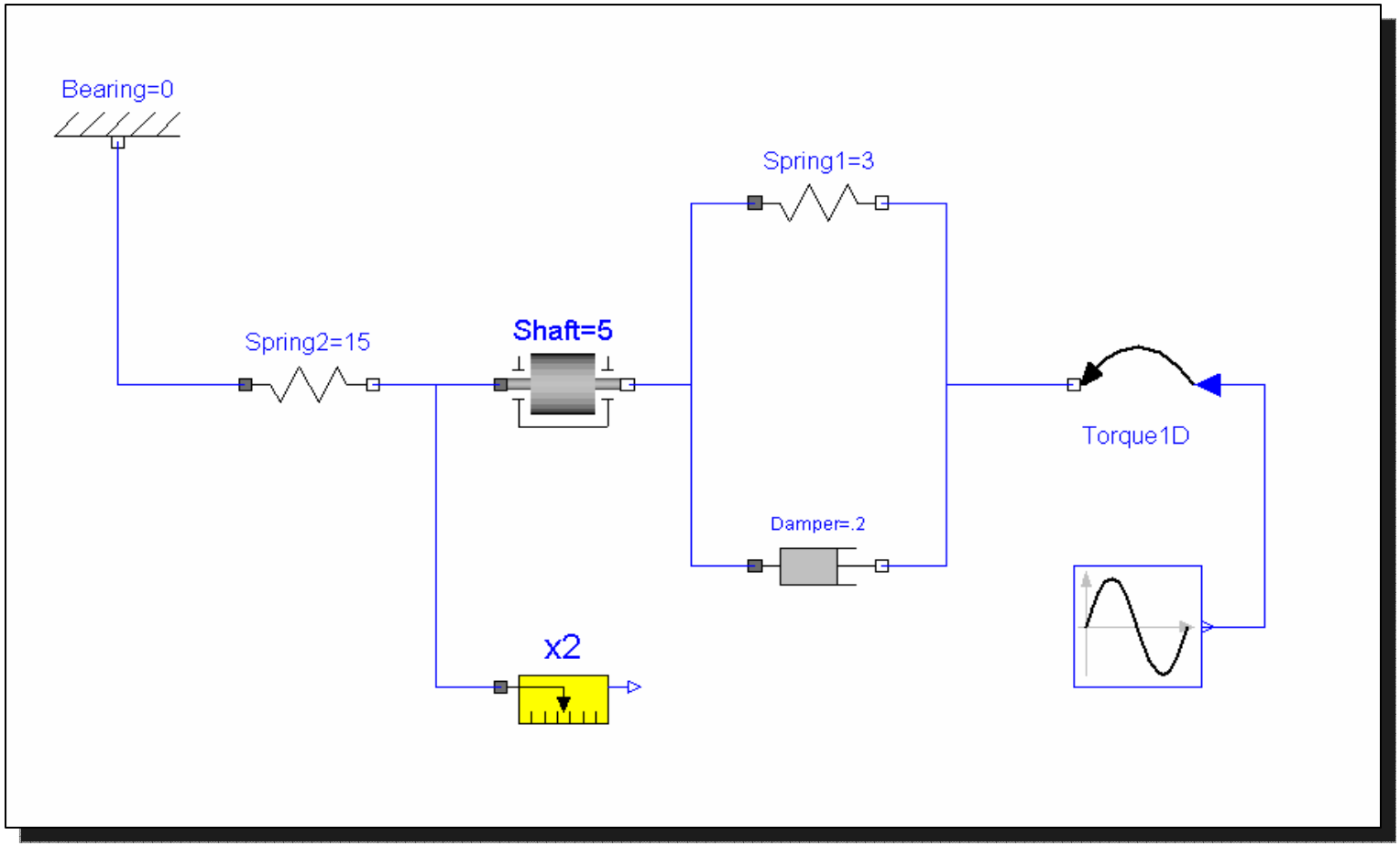


Reuse

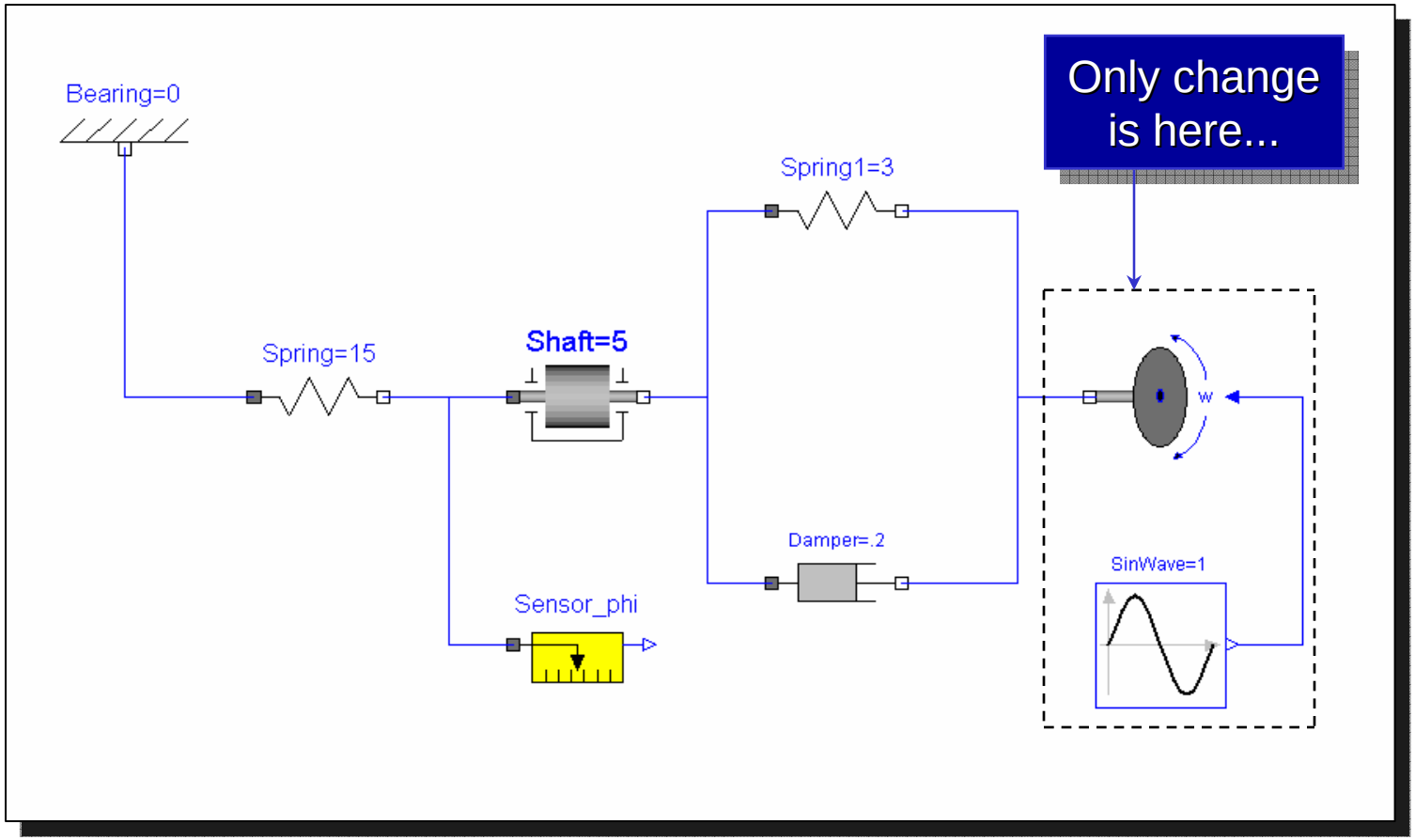
- Fundamental for efficiency and quality
  - Minimize redundant/repeated effort
    - "Redundancy is the root of all evil"
  - Make sure that models that have been validated get reused instead of recreated.
- Support for Inheritance
  - Common in software engineering (Java, C++)
  - Rarely seen in modeling tools, e.g. Simulink, VHDL-AMS, *etc*.
  - Designed into Modelica from the start.

Reuse (cont.)

- Acausal modeling encourages reuse…

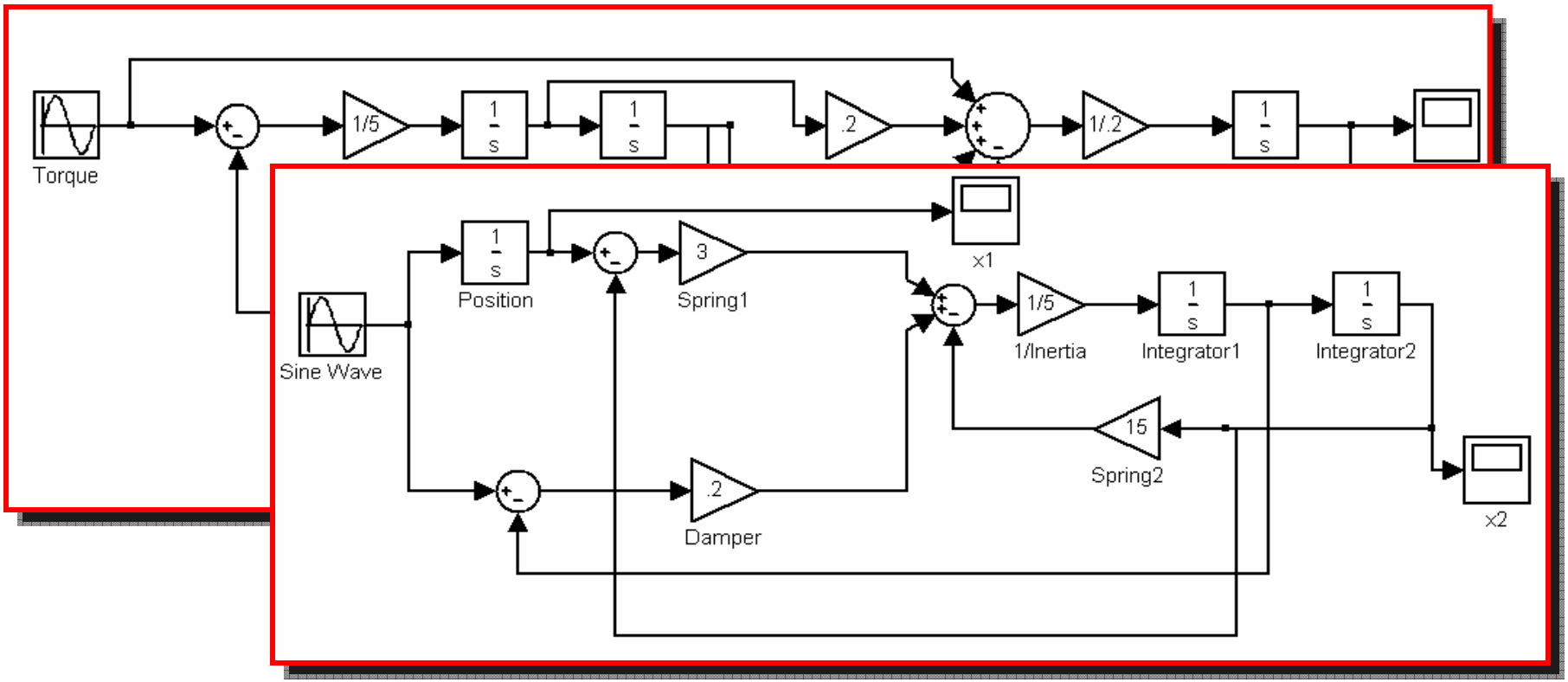


Reuse (cont.)



Reuse (cont.)

- Causal modeling (for physical systems) discourages reuse…



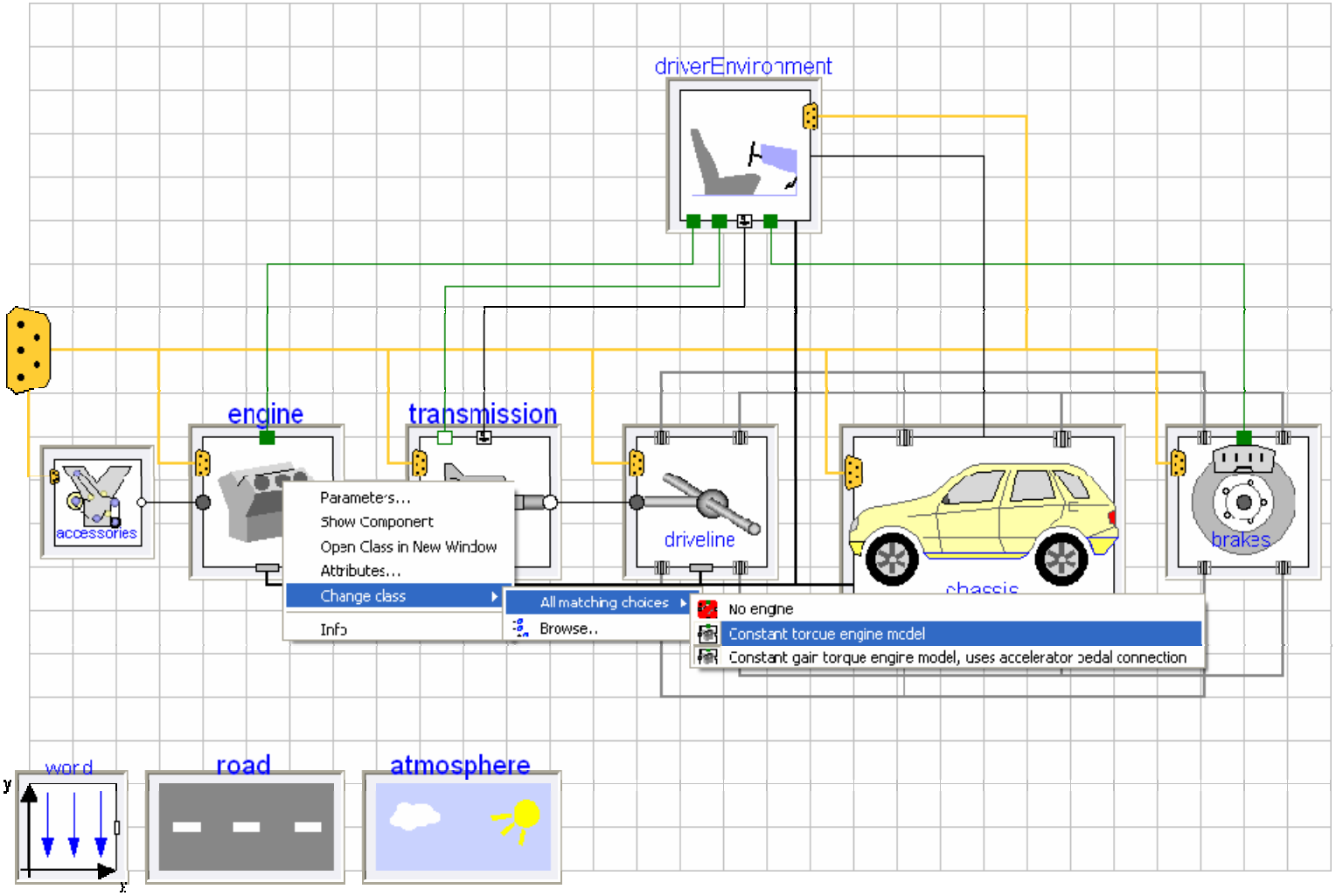
Configuration Management

- Supported through polymorphism in software engineering, but again rarely seen in modeling tools.
- Useful for developing architectures.
- Strong synergy with inheritance, e.g.

```
model DieselExplorer
  extends BaselineExplorer(
     redeclare DieselEngine engine(bore=0.080),
     cylinders=8);
end DieselExplorer;
```

- No “copy and paste” required.

Configuration Management (cont.)



Version Control

- Provides a basic “backup” mechanism.
- Useful features for any developer
  - “diffing”
  - Log messages
  - “Blame” functionality (who/what)
- Supports concurrent development
  - Branching – Creating new capabilities.
  - Merging – Folding them back in.
    - Trimming the “Tree of Life”.

Testing and Validation

- Models as institutionalized knowledge
- Unit testing
  - Don’t just test the system, test the components
  - Establish baseline results.
- Coverage analysis
  - Are all models being tested?
  - Are all conditions and decisions being tested?
- Conservation analysis.
  - Verify balance equations.

Final Thoughts

Troubling Indicators…

- “Skip the details about the model, just show me the results.”
- “Why do you need to spend more time/money on the models when they already work?”
- “I’m just going to make a simple change.”
- “We’ve already spent money on this other tool, can’t you use that?”
- “Have the new guy build a model of it.”

MBD: STAMP Required

- **S** – Skills
  - Tools are not enough.
  - College curriculum needs to include “collaboration” skills.
- **T** – Tools
  - Ideally based on standards.
- **A** – APIs
  - Key to integrate processes and provide “intuitive” interfaces.
  - Open APIs helps avoid vendor lock-in.
- **M** – Models
  - All models are not equal.
  - Need to have the right model at the right time.
- **P** – Processes

Conclusions

- Modeling is…
  - Hard (need more than tools and processes)
  - Costly (don't be a dinosaur)
  - Valuable…
    - Intellectual property
    - Competitive advantage
    - Support decision making
    - Faster time to market
    - Better products
- Impose checks on quality and behavior.
- Model development process
  - Make sure models are there when needed.
  - Make sure models get reused and improved.