Model Management Tools and Technologies

Michael Tiller
Powertrain Research Department
Ford Motor Company

Overview

- Development Tools
  - Version Control
  - Web-based Tools
- Modelica – Open Standard
  - Representing behavior
  - Translation
  - Modelica-based applications

Tools

Choosing Tools

- Tools are like houses:
  - They need a solid foundation to build on
  - You can always add a coat of paint later

Key Principles (for us)

- Cheap
- Tools and technologies should be
  - Open – Non-proprietary and well-documented
  - Proven – Scales for real-world applications
  - Mature – Relatively bug-free
- Treat model development like software development
  - Leverage tools/libraries as appropriate
  - "Redundancy is the root of all evil"

Tools – Version Control

Version Control Requirements

- Archive previous versions
- Archive developer comments on changes
- Allow branching for parallel development
- Allow merging for coherent code base
- Ability to “diff” changes
- Tag (or label) releases
- Retrieve versions by tag, branch or date
- Cross-platform
- Keyword substitution
- Binary file support

Version Control Tools

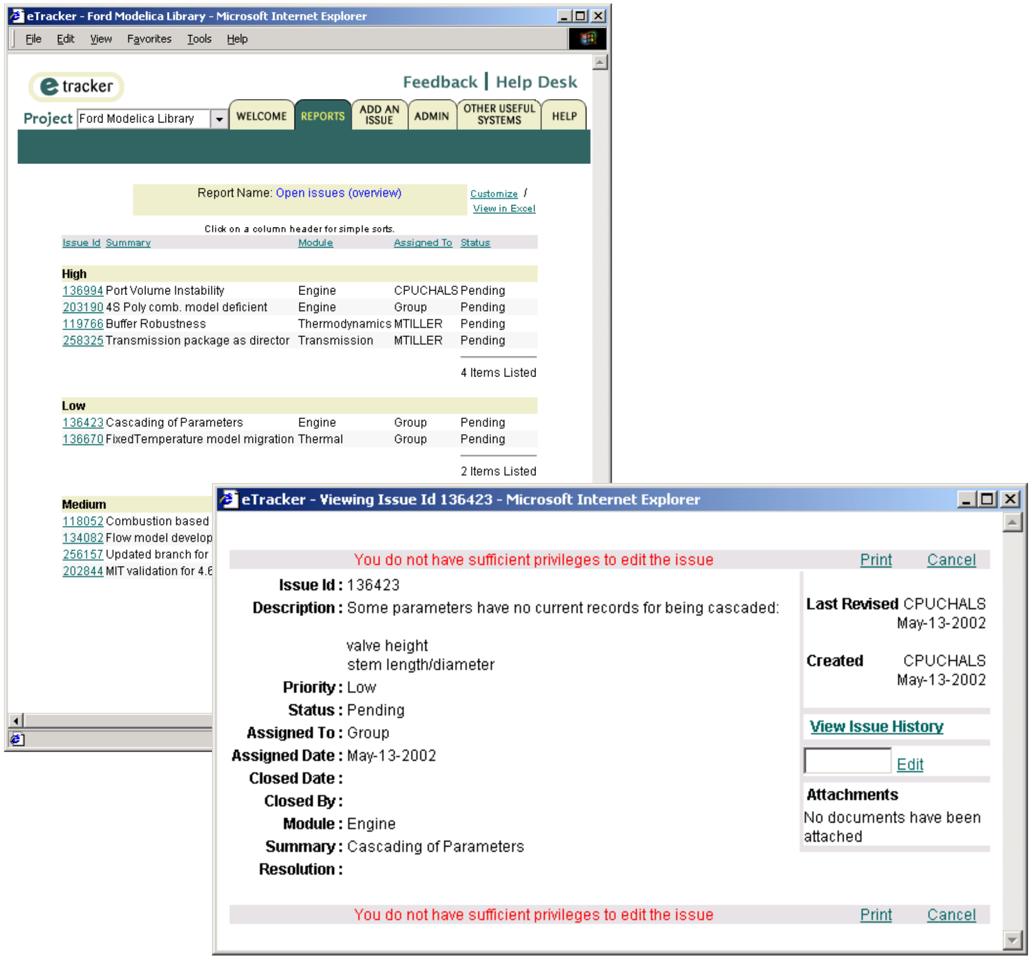
- ClearCase – used extensively
  - Excellent features
  - Expensive
  - High maintenance
- CVS – used extensively
  - Good features (missing a few nice ones)
  - Free
  - Fairly simple to install and maintain
- Perforce – never used
  - Good compromise

CVS User Interfaces

- Command line
- Ancillary Tools
  - TkDiff, CSDiff – Diffing and merging
- Several GUIs
  - WinCVS – primary user interface
  - TkCVS – Works on Windows/Unix
  - CVSWeb – Web based access
- Good example of an open system

Typical Usage

- Create a branch for either a bug fix or enhancement
  - We use eTracker
  - Bug fix branch names use eTracker Ids
- Work in parallel
  - Clearly identifies all changes
  - Avoid “stepping on toes”
- Merge changes



Report Name: Open issues (overview)

Customize / View in Excel

Click on a column header for simple sorts.

Issue Id	Summary	Module	Assigned To	Status
High				
136994	Port Volume Instability	Engine	CPUCHALS	Pending
203190	4S Poly comb. model deficient	Engine	Group	Pending
119766	Buffer Robustness	Thermodynamics	MTILLER	Pending
258325	Transmission package as director	Transmission	MTILLER	Pending
				4 Items Listed
Low				
136423	Cascading of Parameters	Engine	Group	Pending
136670	FixedTemperature model migration	Thermal	Group	Pending
				2 Items Listed
Medium				
118052	Combustion based			
134082	Flow model develop			
256157	Updated branch for			
202844	MIT validation for 4.6			

eTracker - Viewing Issue Id 136423 - Microsoft Internet Explorer

You do not have sufficient privileges to edit the issue — Print — Cancel

Issue Id : 136423

Description : Some parameters have no current records for being cascaded:

valve height
stem length/diameter

Priority : Low

Status : Pending

Assigned To : Group

Assigned Date : May-13-2002

Closed Date :

Closed By :

Module : Engine

Summary : Cascading of Parameters

Resolution :

Last Revised CPUCHALS May-13-2002

Created CPUCHALS May-13-2002

View Issue History

Edit

Attachments

No documents have been attached

You do not have sufficient privileges to edit the issue — Print — Cancel

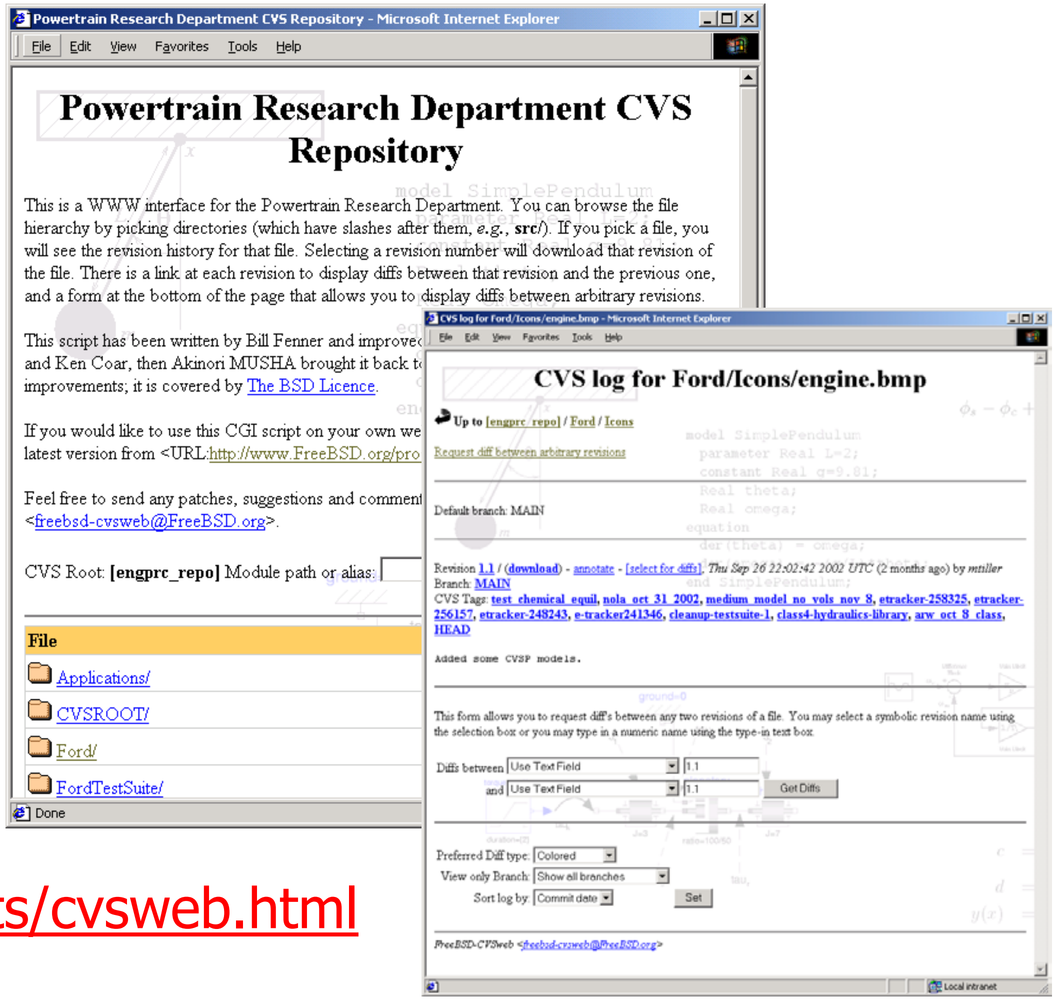
WinCVS

- Create branches and tags
- View local changes and/or conflicts
- View “version tree” for a given file
- Review check-in comments
- “Sandbox” approach
- http://www.wincvs.org/

Tools - Utilizing the Intranet

CVS Web

- Overview of the filesystem
- Download specific versions of files
- View differences between files



Powertrain Research Department CVS Repository - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Powertrain Research Department CVS Repository

This is a WWW interface for the Powertrain Research Department. You can browse the file hierarchy by picking directories (which have slashes after them, *e.g.*, **src/**). If you pick a file, you will see the revision history for that file. Selecting a revision number will download that revision of the file. There is a link at each revision to display diffs between that revision and the previous one, and a form at the bottom of the page that allows you to display diffs between arbitrary revisions.

This script has been written by Bill Fenner and improve and Ken Coar, then Akinori MUSHA brought it back t improvements; it is covered by The BSD Licence.

If you would like to use this CGI script on your own we latest version from <URL:http://www.FreeBSD.org/pro

Feel free to send any patches, suggestions and comment <freebsd-cvsweb@FreeBSD.org>.

CVS Root: **[engprc_repo]** Module path or alias:

File

- Applications/
- CVSROOT/
- Ford/
- FordTestSuite/

Done
CVS log for Ford/Icons/engine.bmp - Microsoft Internet Explorer
File Edit View Favorites Tools Help

CVS log for Ford/Icons/engine.bmp

Up to [engprc_repo] / Ford / Icons

Request diff between arbitrary revisions

Default branch: MAIN

Revision 1.1 / (download) - annotate - [select for diffs], *Thu Sep 26 22:02:42 2002 UTC* (2 months ago) by *mtiller*
Branch: MAIN
CVS Tags: **test_chemical_equil**, **nola_oct_31_2002**, **medium_model_no_vols_nov_8**, **etracker-258325**, **etracker-256157**, **etracker-248243**, **e-tracker241346**, **cleanup-testsuite-1**, **class4-hydraulics-library**, **arw_oct_8_class**, **HEAD**

```
Added some CVSP models.
```

This form allows you to request diff's between any two revisions of a file. You may select a symbolic revision name using the selection box or you may type in a numeric name using the type-in text box.

Diffs between Use Text Field 1.1
and Use Text Field 1.1 Get Diffs

Preferred Diff type: Colored
View only Branch: Show all branches
Sort log by: Commit date Set

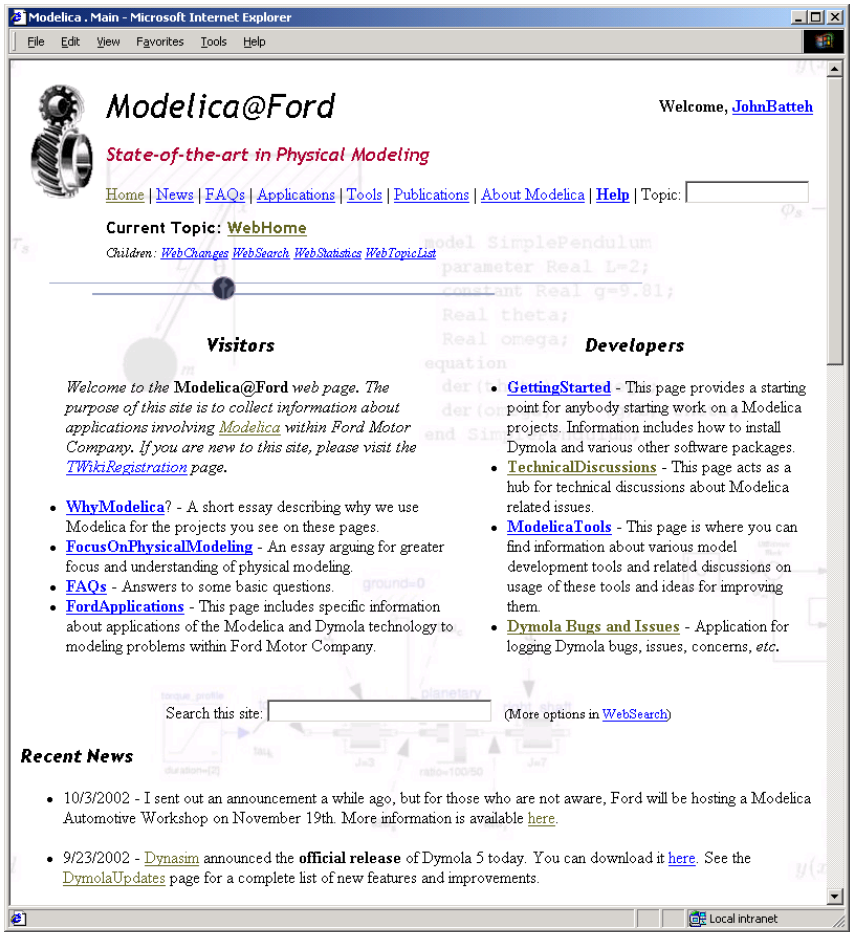
FreeBSD-CVSweb <freebsd-cvsweb@FreeBSD.org>

http://www.freebsd.org/projects/cvsweb.html

Wiki – Collaborative Tool

- Collaborative environment
  - FAQs, user and developer information
  - Technical discussions
- Web-based
- Searchable
- Active server-side scripting
- Customizable
- Free
- Change tracking, version control and diff-ing
- http://twiki.org/

Wiki Screen Shots



Modelica . Main - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Modelica@Ford

Welcome, JohnBatteh

State-of-the-art in Physical Modeling

Home | News | FAQs | Applications | Tools | Publications | About Modelica | **Help** | Topic:

Current Topic: WebHome

Children: WebChanges WebSearch WebStatistics WebTopicList

Visitors

Welcome to the **Modelica@Ford** *web page. The purpose of this site is to collect information about applications involving Modelica within Ford Motor Company. If you are new to this site, please visit the TWikiRegistration page.*

- **WhyModelica**? - A short essay describing why we use Modelica for the projects you see on these pages.
- **FocusOnPhysicalModeling** - An essay arguing for greater focus and understanding of physical modeling.
- **FAQs** - Answers to some basic questions.
- **FordApplications** - This page includes specific information about applications of the Modelica and Dymola technology to modeling problems within Ford Motor Company.

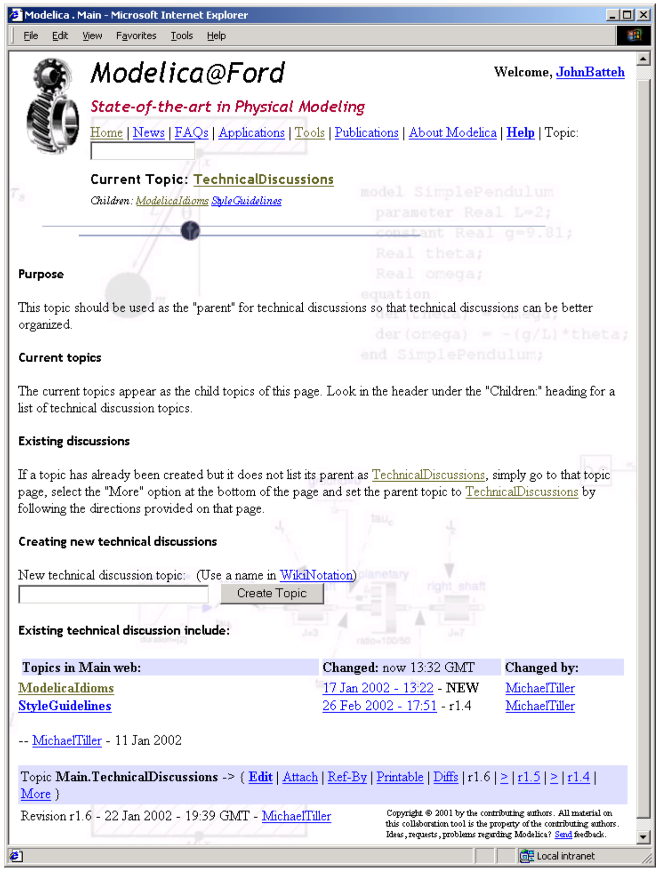
Developers

- **GettingStarted** - This page provides a starting point for anybody starting work on a Modelica projects. Information includes how to install Dymola and various other software packages.
- **TechnicalDiscussions** - This page acts as a hub for technical discussions about Modelica related issues.
- **ModelicaTools** - This page is where you can find information about various model development tools and related discussions on usage of these tools and ideas for improving them.
- **Dymola Bugs and Issues** - Application for logging Dymola bugs, issues, concerns, *etc.*

Search this site: (More options in WebSearch)

Recent News

- 10/3/2002 - I sent out an announcement a while ago, but for those who are not aware, Ford will be hosting a Modelica Automotive Workshop on November 19th. More information is available here.
- 9/23/2002 - Dynasim announced the **official release** of Dymola 5 today. You can download it here. See the DymolaUpdates page for a complete list of new features and improvements.



Modelica . Main - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Modelica@Ford

Welcome, JohnBatteh

State-of-the-art in Physical Modeling

Home | News | FAQs | Applications | Tools | Publications | About Modelica | **Help** | Topic:

Current Topic: TechnicalDiscussions

Children: ModelicaIdioms StyleGuidelines

Purpose

This topic should be used as the "parent" for technical discussions so that technical discussions can be better organized.

Current topics

The current topics appear as the child topics of this page. Look in the header under the "Children:" heading for a list of technical discussion topics.

Existing discussions

If a topic has already been created but it does not list its parent as TechnicalDiscussions, simply go to that topic page, select the "More" option at the bottom of the page and set the parent topic to TechnicalDiscussions by following the directions provided on that page.

Creating new technical discussions

New technical discussion topic: (Use a name in WikiNotation)

Existing technical discussion include:

Topics in Main web:	**Changed:** now 13:32 GMT	**Changed by:**
ModelicaIdioms	17 Jan 2002 - 13:22 - **NEW**	MichaelTiller
StyleGuidelines	26 Feb 2002 - 17:51 - r1.4	MichaelTiller

-- MichaelTiller - 11 Jan 2002

Topic **Main.TechnicalDiscussions** -> { **Edit** | Attach | Ref-By | Printable | Diffs | r1.6 | > | r1.5 | > | r1.4 | More }

Revision r1.6 - 22 Jan 2002 - 19:39 GMT - MichaelTiller

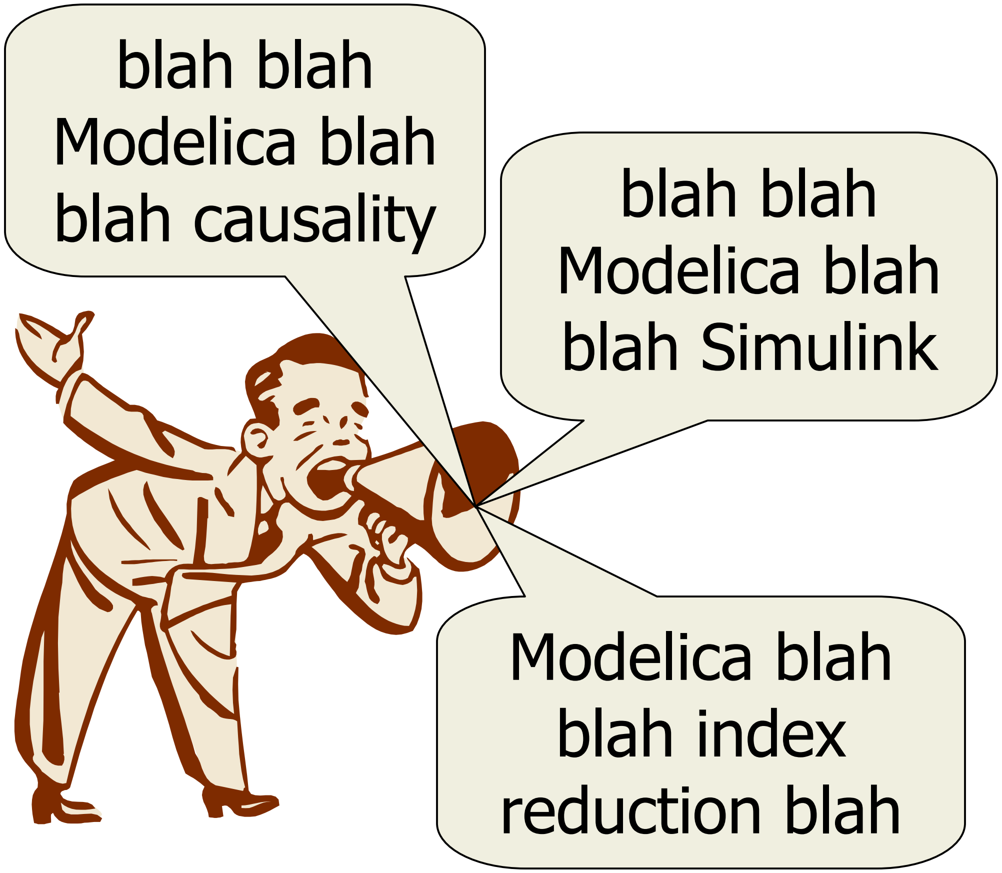
Copyright © 2001 by the contributing authors. All material on this collaboration tool is the property of the contributing authors. Ideas, requests, problems regarding Modelica? Send feedback.

CORBA

- Used to allow programs to communicate and interact over networks
- Open standard
- Cross-platform
- Supports every major language
- Have not needed this functionality yet
- Typical example of thought out solution

Modelica

Unclear Message


blah blah Modelica blah blah causality
blah blah Modelica blah blah Simulink
Modelica blah blah index reduction blah

- Not (strictly) about physical modeling
- Interest in an open specification
- Format for model storage and exchange
- Features beyond behavioral descriptions
  - Data handling
  - Code maintenance
- Non-proprietary
- Scalable

Development Effort

- It requires a considerable effort to identify all the issues and then develop solutions
- We wanted to leverage work already done
- The diverse composition of the Modelica Organization provided many views

Capturing Information

- Encapsulation `(protected, public, encapsulated)`
- Built-in types capture
  - Units, min & max values, nominal values
- Annotations – Meta Data
  - Non-behavioral (*e.g.* graphical)
  - All tools must preserve this data
  - Structured/Hierarchical (conceptually like XML)
  - Avoids “kitchen sink” approach
  - Useful for other applications (*e.g.* FMEA)

Meta-Data

- Annotations
  - Currently used for several things including
    - "Masks"
    - Laying out components and connections
    - Documentation
  - Completely open-ended
  - Can be associated with models, functions, packages, model instances, *etc*
- Structured comments
- Possible to store arbitrary data files in the Modelica model hierarchy

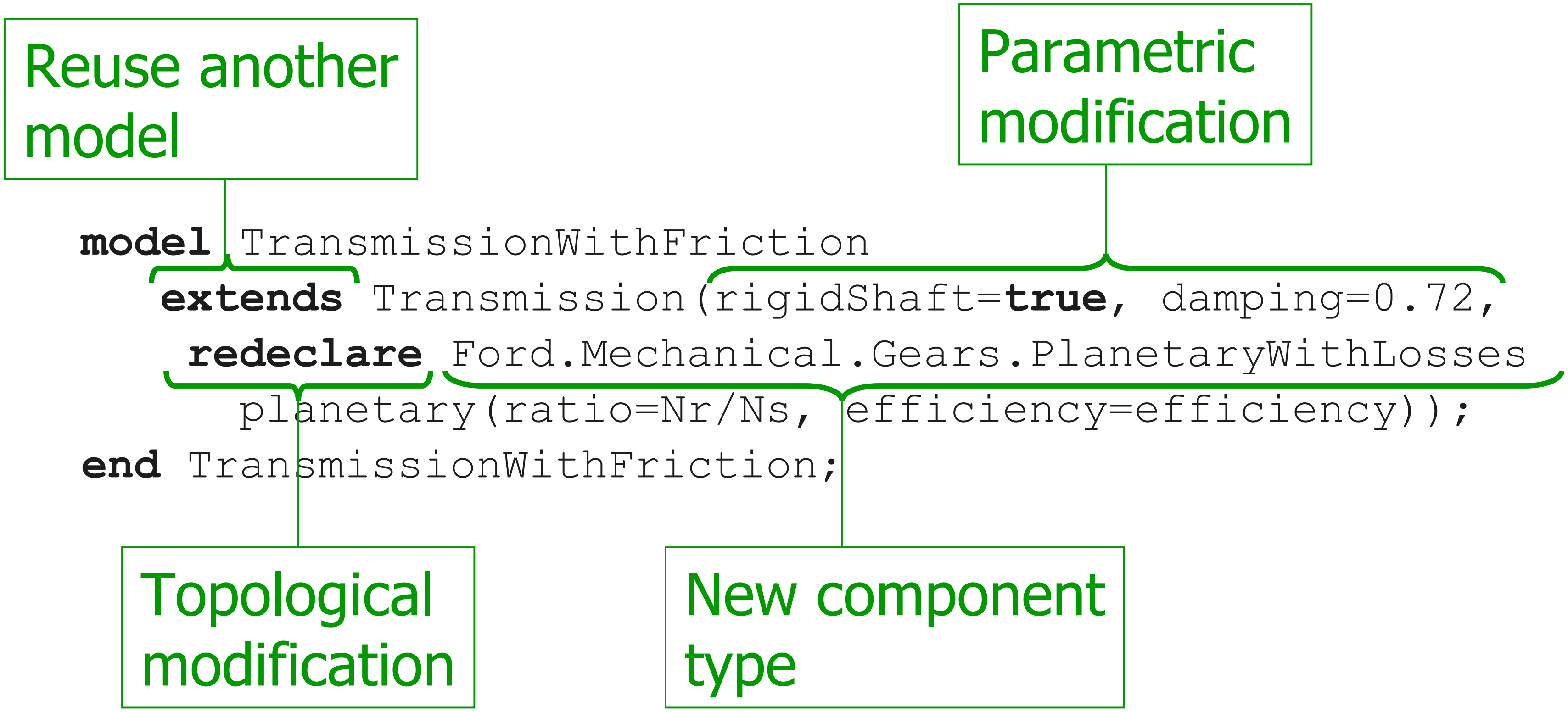
Configuration Management

- Example: Level of fidelity
- “Redundancy is the root of all evil”
- Version control provides different dimension sometimes misused, e.g.
    - Version 1.3 – Model with base halfshafts
    - Version 1.4 – Model with stiff halfshafts
    - Version 1.5 – Model with slip clutch
- We build this into the model
    - Parametrically change whether shaft is rigid or compliant
    - Swap one component for another with a comparable interface

“Variant” Control

- Using “Save As” is problematic
- Variants are nice because they
  - Reference other classes (not copy)
  - Do not alter the “base” model
  - They explicitly list exactly what has changed (called modifications)
- Type safe

Example



Reuse another model

Parametric modification

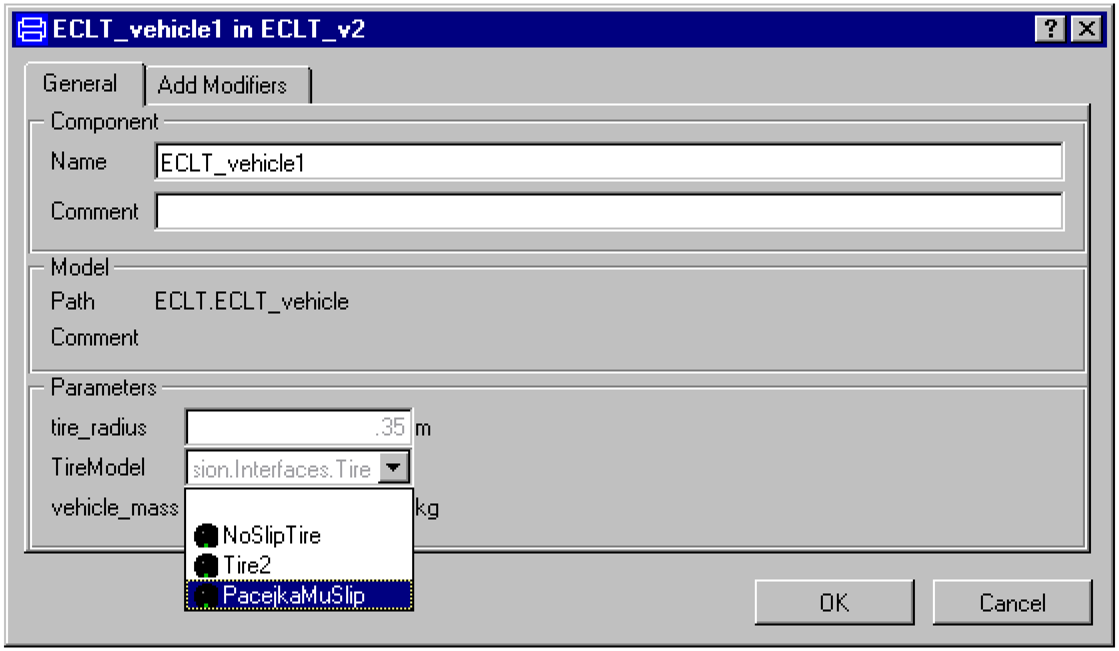
```
model TransmissionWithFriction
   extends Transmission(rigidShaft=true, damping=0.72,
    redeclare Ford.Mechanical.Gears.PlanetaryWithLosses
      planetary(ratio=Nr/Ns, efficiency=efficiency));
end TransmissionWithFriction;
```

Topological modification

New component type

GUI Handling

- Variations are handled similarly to LSM



ECLT_vehicle1 in ECLT_v2

Component
Name ECLT_vehicle1
Comment

Model
Path ECLT.ECLT_vehicle
Comment

Parameters
tire_radius .35 m
TireModel sion.Interfaces.Tire
vehicle_mass kg

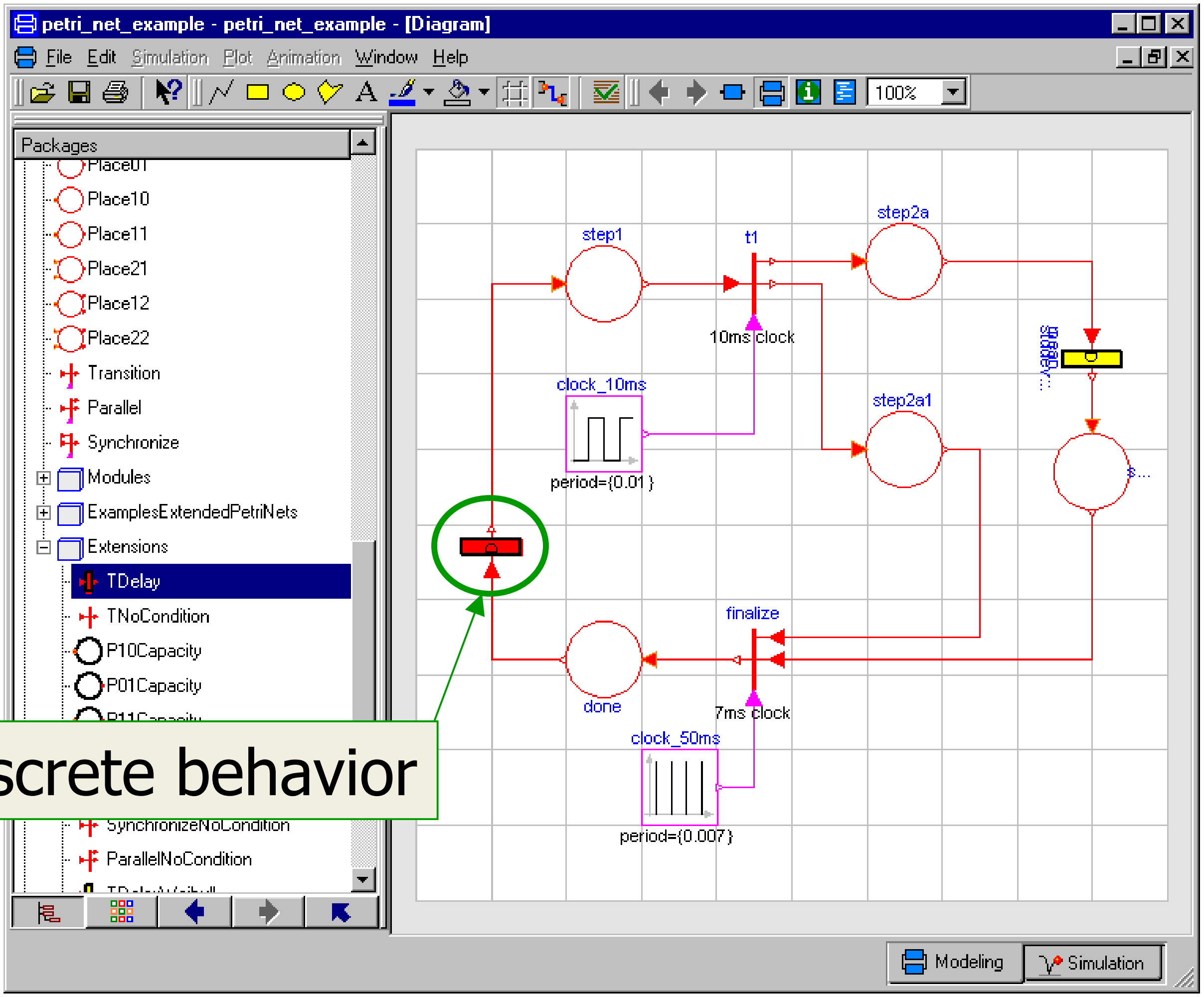
NoSlipTire
Tire2
PacejkaMuSlip

- Annotations are used for choices

Representing Behavior

- Modelica includes both declarative and procedural semantics
    - Can be used for scripts and functions
- Can handle both continuous and discrete behavior
    - Can be used for things like Petri nets, digital circuits and state charts
- Different GUIs might be required to build models
- Different simulators may provide different performance for different problem types

Control Logic



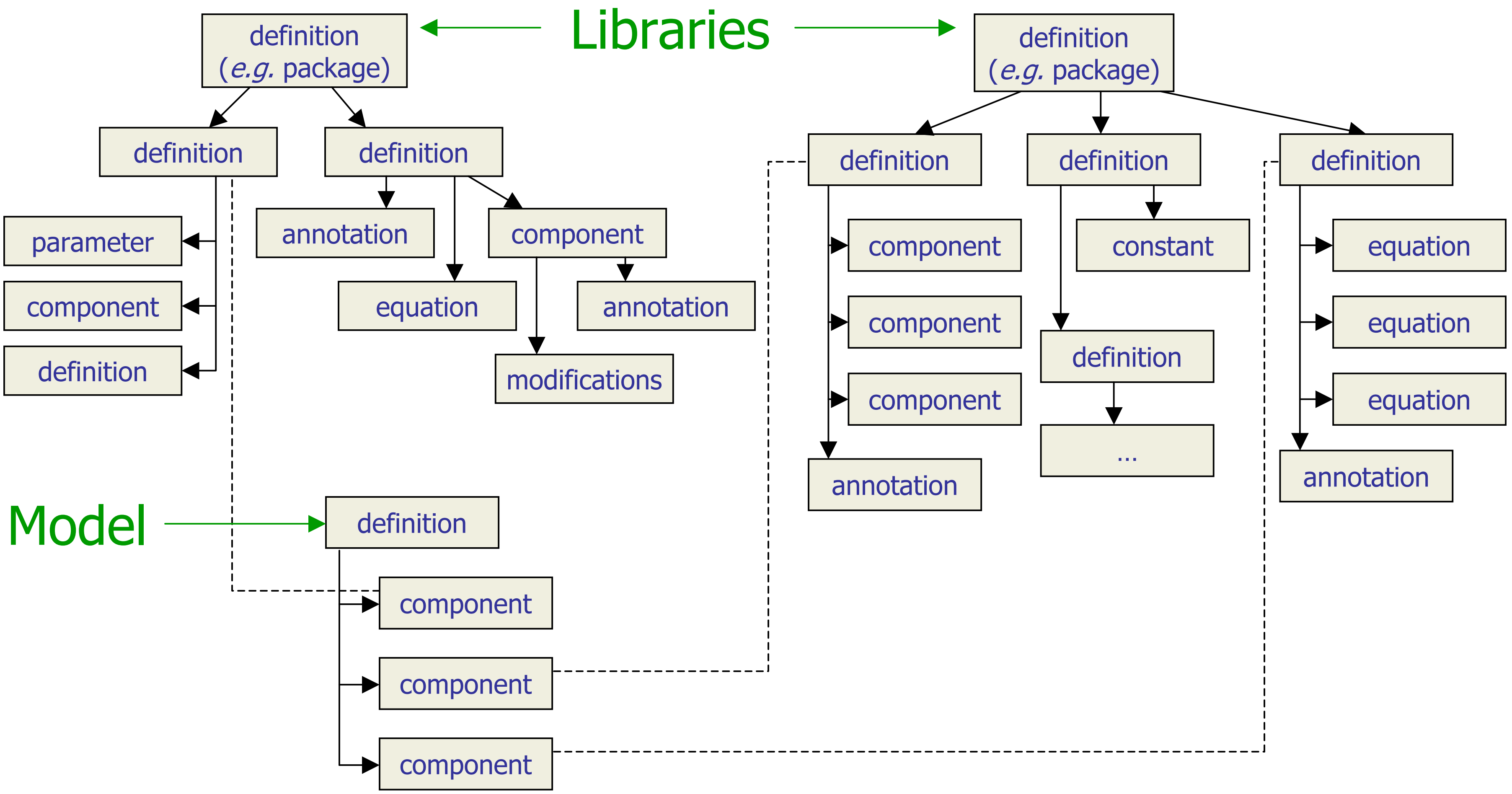
Discrete behavior

Control Logic (cont.)

```
model Tdelay "Transition with one input and one output connection"
  parameter Real delay=5;
protected
  Boolean activated;
  Boolean delay_passed;
  Boolean fire;
  Real last_activation_time;
equation
  //activation of transition
  activated = inTransition.state;
  //set activation time
  when activated then
    last_activation_time = time;
  end when;
  //activated and delay passed
  delay_passed = activated and ((time - delay) > last_activation_time);
  //fire command
  fire = activated and delay_passed and not outTransition.state;
  //propagate firing to in and output places
  inTransition.fire = fire;
  outTransition.set = fire;
end TDelay;
```

Modelica Applications

Sample Package Structure

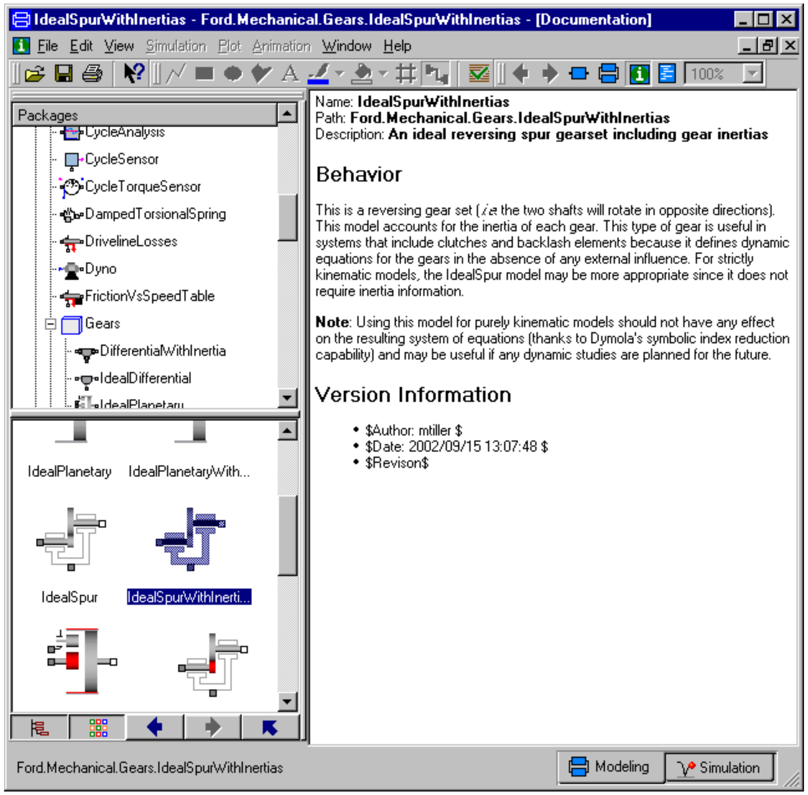


Modelica Tools

- Simulate models
- Generate documentation
- Bundle models for distribution
- Style guideline reporting
- Other possibilities
  - Specialized diffing/merging
  - Code generator
  - Obfuscation/shrouding
  - Conversion/transformation utilities
  - Formal verification

Example: Dymola Layers

- Dymola uses annotations for:
  - Icon graphics
  - Diagram layout (and drawing elements)
  - Documentation
- Result
  - All captured in the model
  - Viewable in Dymola
  - Used to generate HTML documentation



Name: **IdealSpurWithInertias**
Path: **Ford.Mechanical.Gears.IdealSpurWithInertias**
Description: **An ideal reversing spur gearset including gear inertias**

Behavior

This is a reversing gear set (*i.e* the two shafts will rotate in opposite directions). This model accounts for the inertia of each gear. This type of gear is useful in systems that include clutches and backlash elements because it defines dynamic equations for the gears in the absence of any external influence. For strictly kinematic models, the IdealSpur model may be more appropriate since it does not require inertia information.

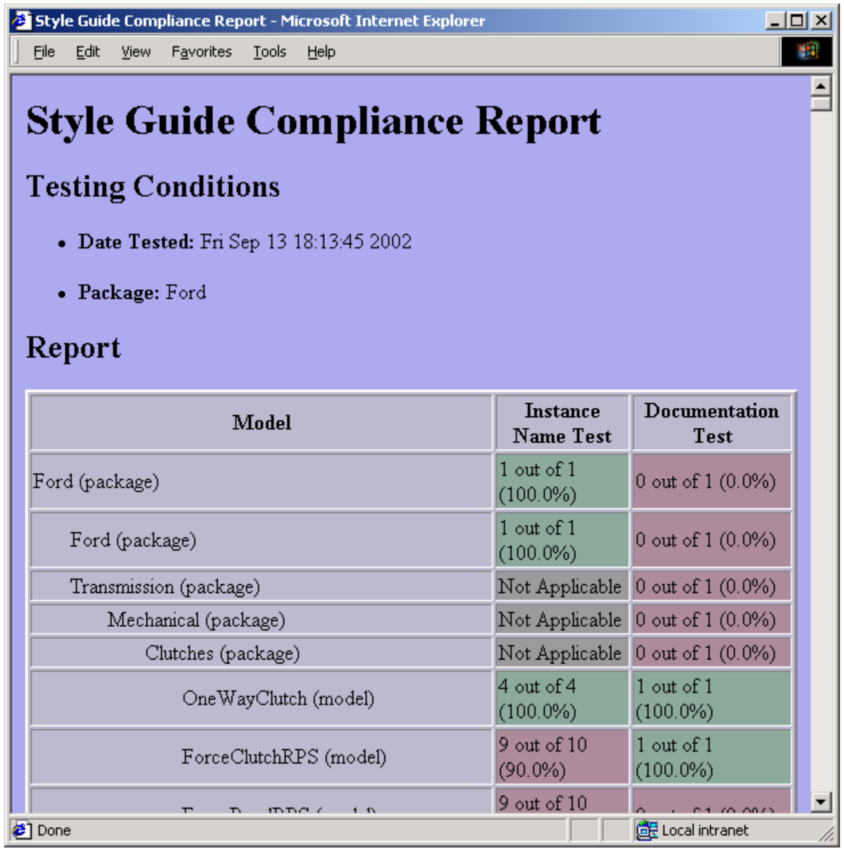
Note: Using this model for purely kinematic models should not have any effect on the resulting system of equations (thanks to Dymola's symbolic index reduction capability) and may be useful if any dynamic studies are planned for the future.

Version Information

- $Author: mtiller $
- $Date: 2002/09/15 13:07:48 $
- $Revison$

Style Guidelines

- Typical requirements
  - Naming conventions
  - Documentation requirements
  - Avoid confusing constructs
  - Make code more reusable/maintainable
- Possible to automate the process of checking these guidelines
  - Report Generator



Style Guide Compliance Report

Testing Conditions

- **Date Tested:** Fri Sep 13 18:13:45 2002
- **Package:** Ford

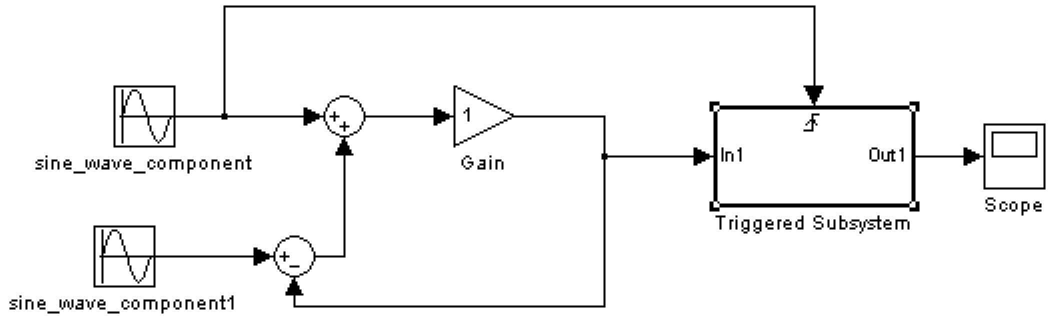
Report

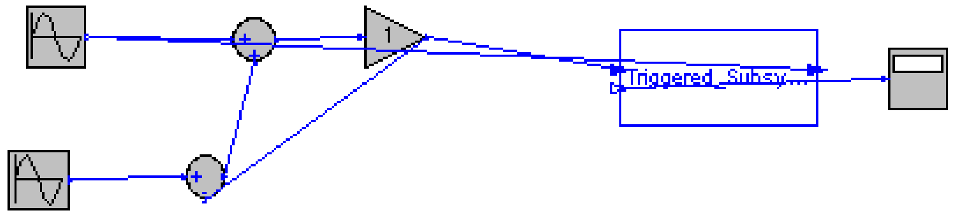
Model	Instance Name Test	Documentation Test
Ford (package)	1 out of 1 (100.0%)	0 out of 1 (0.0%)
Ford (package)	1 out of 1 (100.0%)	0 out of 1 (0.0%)
Transmission (package)	Not Applicable	0 out of 1 (0.0%)
Mechanical (package)	Not Applicable	0 out of 1 (0.0%)
Clutches (package)	Not Applicable	0 out of 1 (0.0%)
OneWayClutch (model)	4 out of 4 (100.0%)	1 out of 1 (100.0%)
ForceClutchRPS (model)	9 out of 10 (90.0%)	1 out of 1 (100.0%)
ForceDrumRPS (model)	9 out of 10	0 out of 1 (0.0%)

Translating Representations

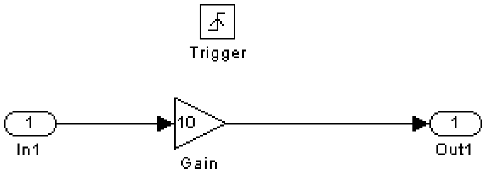
- Example: (simple) Simulink diagrams
- Read “.mdl” file, generate an AST
- Process AST and then generate Modelica code
- Components involved:
  - Continuous: Sum, Sin, Gain
  - Discrete: Triggered Subsystem

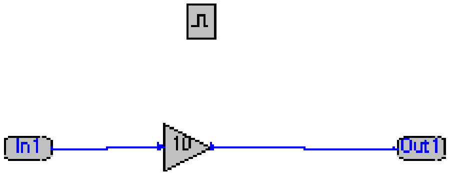
Toplevel System

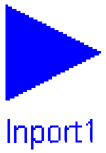


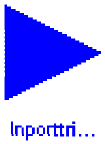


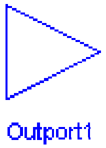
Subsystem


Trigger
1
In1
10
Gain
1
Out1


In1
10
Out1


Inport1


Inporttri...


Outport1

Preserving Information

```
Block {
  BlockType       Sin
  Name            “sine"
  Position        [120, 70, 150, 100]
  SineType        “Time-based"
  Amplitude       "1"
  Bias            "0"
  Frequency       "10"
  Phase           “0“
  Samples         “10"
  Offset          "0"
  SampleTime      "0"
  VectorParams1D  on
}
```

```
Simulink.Sin sine(
  Bias=0, Amplitude=1, Phase=0,
  Frequency=10) annotation(
     Simulink(Offset="0",
              SampleTime="0",
              VectorParams1D=on,
              Samples="10",
              SineType="Time-based"),
     extent=[-82,74;-72,84]);
```

Conclusions

Summary – Tool Philosophy

- Handle version control and configuration management differently
- Use open and mature tools and technologies as a foundation
- Be prepared to customize, this requires
  - Open standards
  - Application Programmer Interfaces (APIs)

Summary – Modelica

- Captures all kinds of behavior
- Fairly mature and scalable
- Handles meta-data
- Avoids the “Save As” syndrome
- Room for more than simulation tools
  - Doesn’t have to be a monolithic system