

Requirements and Specifications in a Model Based Development World

William P. Milam
Ford Motor Company

Rough Outline

- Why all the fuss?
- Requirements vs Constraints vs Assumptions
- Specifications:
 - Executable Specifications
- The model based development world
 - Where does all this fit and how does it work?
 - Mathworks Automotive Advisory Board(MAAB)
 - SAE Model Based Embedded Systems Engineering(MBESE/J2780)
- Musings on the future..

Why all the fuss?

- Automobiles are the most complex consumer device in the world. Period.
- The automotive industry does aerospace complexity on a very limited budget.
- Requirements are largely done with little more than a spreadsheet.
- F-22 should be a wakeup call....
- It's not a software problem, it's a system engineering problem.

Requirements vs Constraints

- Same coin, different sides
 - When the switch is pressed, the window will be lowered.
 - The system will not use interrupts.
 - If the switch is depressed for more than one second, the window will stop lowering if the switch is released.
 - If the switch is depressed for less than .75 seconds, the window will fully lower.
 - The system will not consume more than 10ma.
- Are constraints a special subset of requirements?

What about assumptions?

- The undocumented constraints.
- Extremely common and totally unmanaged.
- Same challenge for assumptions as there is for requirements and constraints, how to capture, analyze and manage.

Functional vs Para-Functional

- Functional – What it should do. Press the button and the window goes down.
- Para-Functional – Other requirements to ensure expected behavior. i.e.: Must not use more than 10ma. Formerly known as Non-Functional.
- I assume the window mechanism behavior is linear.

Specifications

- Built from requirements and constraints.
 - **Specification for window lowering function:** The system will consist of a button that can be depressed by the user that will cause a window to be lowered in one of two modes. For the easy down function, if the user momentarily depresses the switch, the window will fully lower. For normal operation, if the user keeps the button depressed for more than a second, the window will stop lowering when the button is released. The system will not consume more than 10ma under any circumstances. Interrupts cannot be used to implement this capability.

Executable Specifications

- Oxymoron?
 - If it's executable, it is an implementation.
 - Trade-offs were made in order to get the execution to match desired outcome.
- Mathworks is de-facto standard for automotive industry and very wide spread use in aerospace.
- Abstract Implementation
- Types of abstraction: continuous, discrete, finite state.
- Levels of refinement, or elaboration?

3 level taxonomy

As defined in DARPA MoBIES

- Level 1 – continuous time, no regard for controller sampling or signal naming. Anything goes.
- Level 2 – conforming to style guides with rigorous signal naming and characterization. Model building blocks are restricted to a subset to admit analysis. Controller models are restricted to discrete-time/discrete-event building blocks and have timing information included.
- Level 3– Level 2 model augmented with software design information to admit automatic code synthesis.

4 level taxonomy

As proposed for SAE MBESE

- Level 1 – continuous time, no regard for controller sampling or signal naming. Anything goes.
- Level 2 - Same as Level 1, but can be code generated for rapid prototyping purposes.
- Level 3 – conforming to style guides with rigorous signal naming and characterization. Model building blocks are restricted to a subset to admit analysis. Controller models are restricted to discrete-time/discrete-event building blocks and have timing information included.
- Level 4 – Level 2 model augmented with software design information to admit automatic code synthesis.

Level One

- Unconstrained.
- Think of it as a step better than back of an envelope.
- Unmanaged.
- Lots of assumptions.

Level Two

- Constrained to only contain blocks that can be code generated for rapid prototyping(RP) purposes.
- Sole purpose in life is to support RP.
- Some effort put into defining interface.
 - Especially true for by-pass RP.

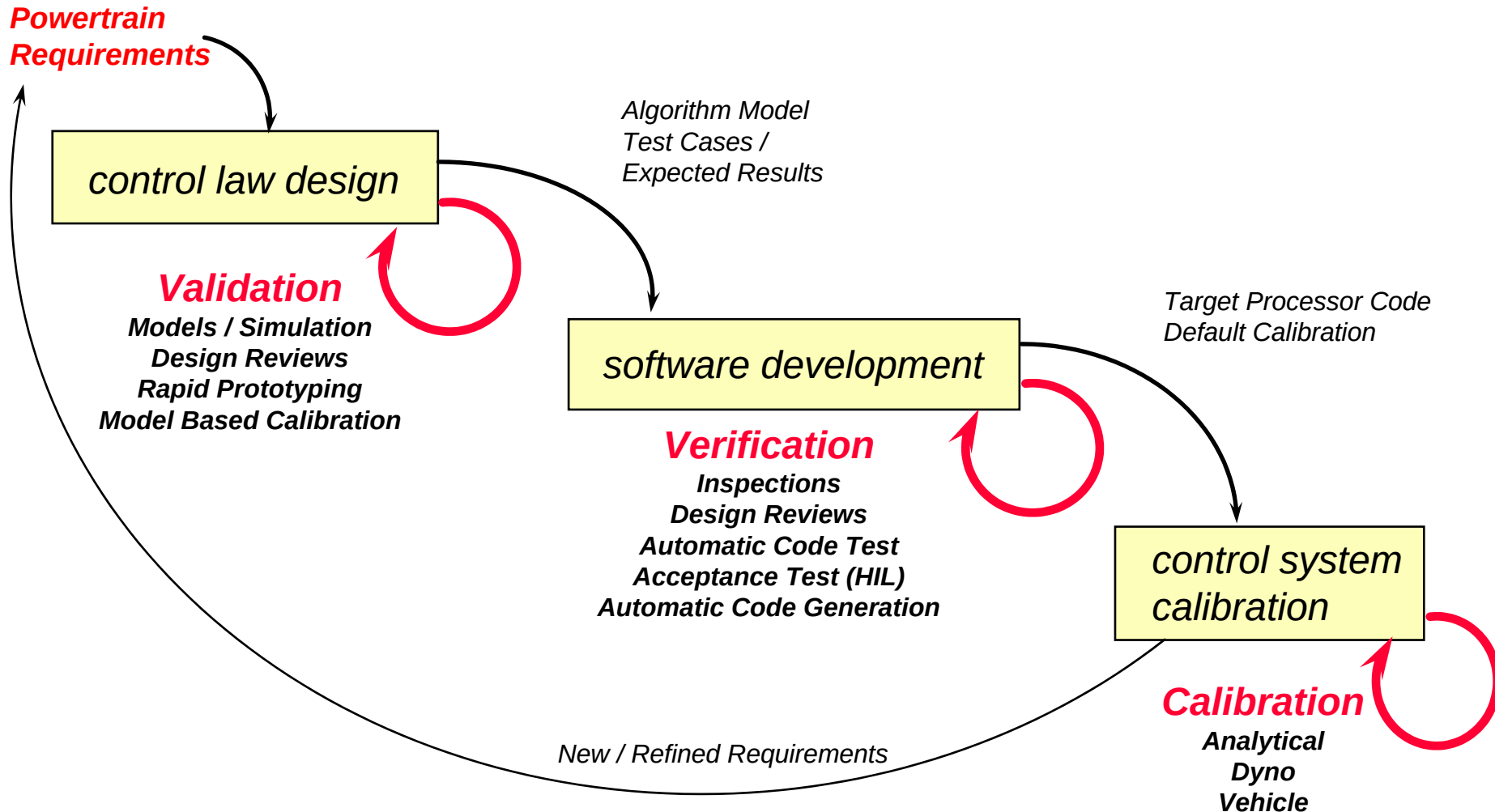
Level Three

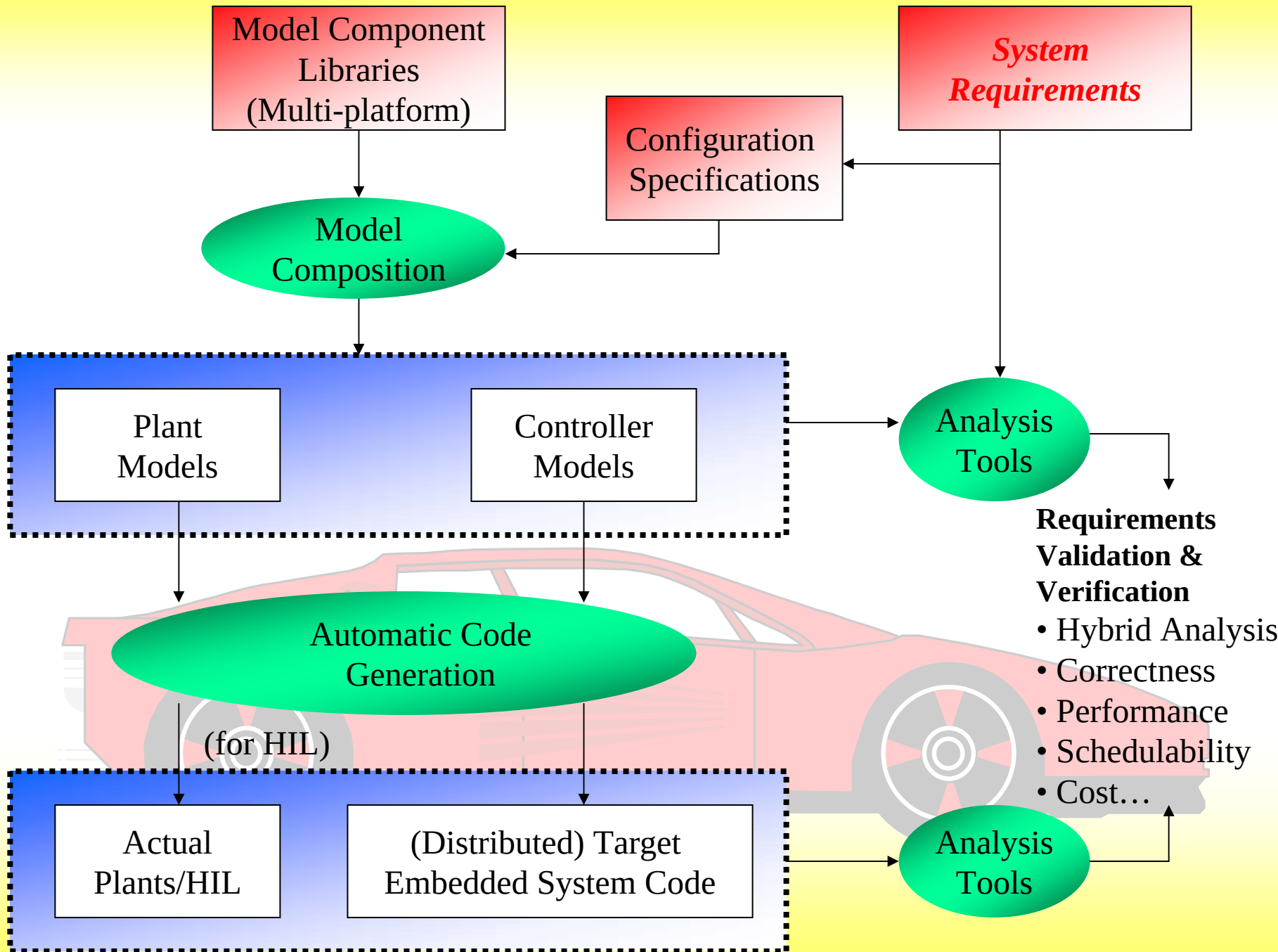
- Conforms to style guide, local or MAAB
- Think of it as a well written program using good programming style. Still short on comments.
- Managed.
- Use datatyping and well defined interfaces
- Naming standards for all signals, parameters and sub-systems.

Level Four

- Push Button ready.
- Managed.
- Models managed as sub-systems which map to procedures.
- Fully discrete.
- Have to deal with certification requirements.
- Support for data acquisition and calibration.

Where Does it All Fit?





Mathworks Automotive Advisory Board

- Started in 1998 with Ford, Daimler-Benz and Toyota
- Grown to worldwide organization with annual meetings alternating between Dearborn and Stuttgart.
- Two main sub-groups:
 - Japan MAAB(J-MAAB)
 - North American MAAB(NA-MAAB)
 - Europe decided to combine with NA

NA-MAAB Working Groups

- Data Management – Mainly data dictionary functions
- Code Generation – Enhancements to support production code generation.
- Style Guide – First update for MAAB style guide in 5 years.
- Physical Modeling – Integrating plant models from other languages and tools.
- Systems/Software Architecture – Enhancements to support systems level modeling

SAE J2780

Model Based Embedded Systems Engineering

- Started in 2004.
- Part of SAE Electronics Design Automation(EDA) Technical Committee.
- Members from both EDA and Embedded Software committees.
- Part of ground vehicle part of SAE, but have liaison with SAE Aerospace embedded group.

Words from SAE announcement

- Interest has been expressed in creating a Working Group to develop a recommended practice on model based systems/software engineering. This would be restricted primarily to in-vehicle control systems such as powertrain, chassis, body and so on. It would include control system design, plant modeling of vehicle systems and sub-systems, requirement capture and analysis and other things. It would likely feed into what the Embedded Software people, SAE J2516, are working on.
-
- The plan would be to work at defining the umbrella global process that would utilize all of the piece parts that the committee typically works on. It's not defined anywhere else at present and we have interest from other OEMs in forming such a group. We would like to start with primarily OEMs and suppliers. The initial focus is to address the global issue and not be tied to any particular tool, method or process.
-
- This is an outgrowth of the DARPA/MoBIES work. While many individual pieces may exist, there is no real guidance for how and when to use them. Nor is there anything resembling a road map of how we should connect the various pieces to implement a particular process to develop a particular system.

J2780 MBESE

- Developing Recommended Practice for model based development of embedded systems used in ground vehicles.
- Core group of participants from NA-MAAB
- Using lessons learned from MoBIES

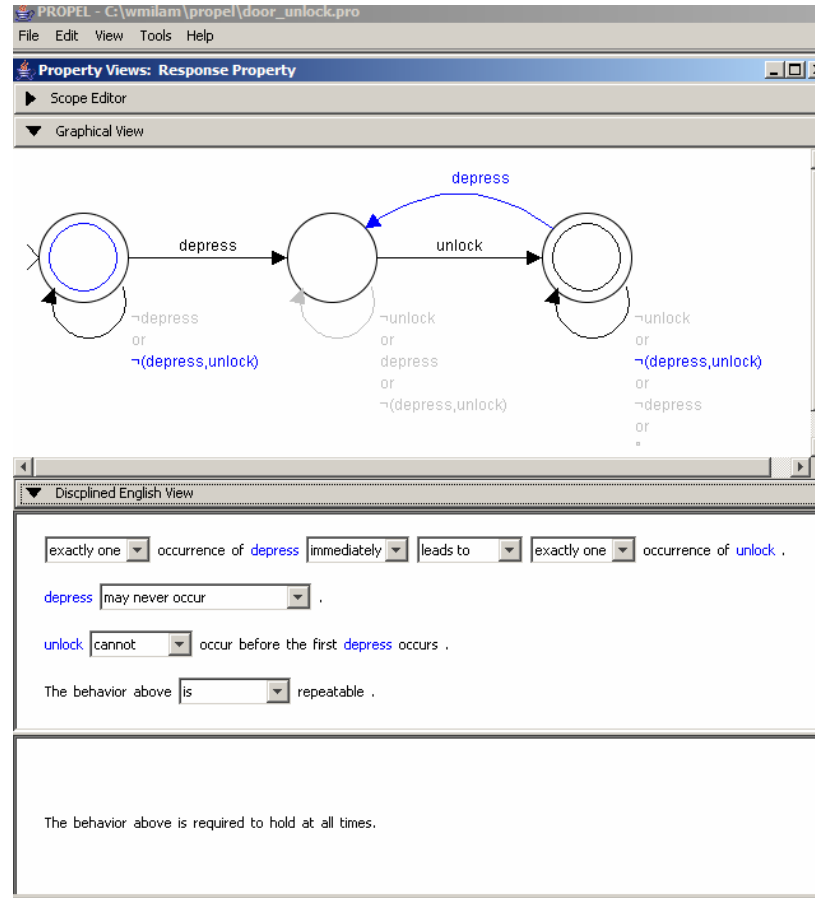
Musings

- We seem to have a handle on modeling, so what don't we have?
 - Requirements capture
 - Requirements analysis
 - Requirements management
- Common problem for all of the above is it has to work with existing model based paradigms. We cannot afford revolutions so we prefer evolution.

Requirements Capture and Analysis

- Related problems.
- Requirements capture today is basically natural language.
- Prefer capturing requirements as some form of automata or in some formal language.
- With requirements captured in a form we can manipulate with a computer, analysis is achievable.
- Analysis does not stop at the particular artifact under consideration.
 - As there is a composition of model components to create a system model, there is also a composition of the requirements.
 - Are there conflicts?
- Might consider this invisible formal methods.

Requirements Capture?



Requirements Management

- More than just tracking changes.
- Linkage to related requirements for a single artifact(sub-system, feature, etc)
- Forward and backwards reference.
 - From any requirement, should be able to navigate to final implementation.
 - From any implementation, should be able to trace back to all requirements that affect the implementation.
- Requirements used as properties and behaviors that models are verified against.