Background

GMPT EI&S develops software for embedded powertrain controllers
- Numerous engine and transmission combinations
  - Freewheel and clutch to clutch transmissions
- Different system architectures
  - ECMs, TCMs, and PCMs (About 50 in total)
- Various vehicle platforms (Over 100 in total)

Background Continued

- 2 Mb of S/W in Controller
- 500,000 Lines of C Code
- 5000 subroutines/functions
- 64K RAM
- Controllers Are Power PC Based
GMPT Definition of Software Architecture

Software Architecture is the backbone on which algorithm functionality can be partitioned. It includes functional decomposition, interfaces, and how to integrate those pieces.
The foundation of the software architecture is based upon the observation that there are two basic methods by which to decompose (i.e., partition) the software:

- **Functional decomposition**
  - The software may be decomposed into functional areas (e.g., exhaust gas recirculation, theft deterrent, torque converter clutch, etc.), which vary across applications based upon feature content.

- **Structural decomposition**
  - The software may be decomposed into structural classifications (e.g., I/O, communications, control, etc.), which are consistent across all applications.

The software is initially functionally decomposed into functional components (a.k.a. "rings")

- Relatively strong data coupling and functional cohesion between different structural classifications within a single functional area.
- Relatively weak data coupling and functional cohesion within a single structural classification across different functional areas.

Each functional component is subsequently structurally decomposed into structural classifications

- Results in a common structure (i.e., architecture) for all functional components.
- Facilitates the development of templates and utilities.
Software Architecture - Variants

Variants of a functional component may be required due to requirements for different levels of functionality in different controllers

- One controller reads a sensor (and transmits the value over a communication link)
- Another controller simply receives the value over a communication link
- Redundant processing performed in different controllers

All variants support the same set of interfaces

Traditional Process
-- Communication through Static Documentation --

Modeling and Simulation Process
-- Communication through Executable Specifications --
**Benefits of Modeling and Simulation**

- Reduces time from concept to production
  - Faster innovation cycles for algorithm development
- Documentation, Verification and Coding time
- Clear Depiction of Algorithms via Executable Specifications
  - Dynamic rather than static documentation
  - Abstraction of algorithm to a higher level
- Reduces the opportunity for translation errors
  - Direct relationship between initial model and production system
  - Verification can be performed at each design stage

---

**Model-Based Algorithm Development Process**

1. Algorithm Model Development
2. Model Simulation and Verification
3. Model-Based ADD Development
Algorithm Model Development

Supporting Items

- **GM Powertrain Standard Modeling Block Set**
  A standard set of blocks comprised of typical PCM functions commonly used in control algorithms which can be used as basic elements for model development regardless of tool. Engineers will be able to recognize these blocks because of their identical graphical representation and know the underlying functionality.

- **MAAB Modeling Guidelines**
  MathWorks modeling guidelines developed by the MathWorks Automotive Advisory Board made up of representatives from GM, Ford, Chrysler and Toyota.

- **GM Powertrain Modeling Guidelines**
  Further Mathworks modeling guidelines developed within GM Powertrain building on the MAAB guidelines.

- **Algorithm Modeling Template**
  A Mathworks model template that provides the structure and layout for a model-based Algorithm Description Document.

Model-Based Algorithm Development Process

1. Algorithm Model Development
2. Model Simulation and Verification
3. Model-Based ADD Development

Model Simulation and Verification

Objective:
Create an environment to allow engineers to conveniently perform the proper simulation method for a control algorithm and quickly transition between these methods to verify algorithm functionality.

- **Model Simulation on Engineer’s Desktop**
  - Open Loop Simulation
    - Simulation of control algorithm only. No plant model.
    - Data usually captured from vehicle operation and used as input stimulus for control algorithm model.
  - Closed Loop Simulation
    - Simulation of both control algorithm and vehicle plant model.

- **Model Execution in Vehicle on Rapid Prototyping Hardware**
  - Algorithm models are executed outside of the PCM on a high-speed controller.
Model-Based Algorithm Development Process

1. Algorithm Model Development
2. Model Simulation and Verification
3. Model-Based ADD Development

Model-Based ADD Generation

Objective:
- Extract information contained in the algorithm model and supplement any additional information required to generate a complete Algorithm Description Document.
  - Automatic generation from algorithm model
    - Algorithm model must conform to the Algorithm Modeling Template

Traditional ADD Context Level Diagram
Model-Based Context Level Diagram

Traditional ADD -- Sequential Execution and Activation

"Activate each of the processes of the pump model at the appropriate rate and in the appropriate order."

Model-Based ADD -- Sequential Execution and Activation
### Traditional ADD -- Control PSPEC

1.1.1 Determine Prime Pump Request

This process requests the pump state based upon a request. The input Variable: Input_1, the output of this process.

Variable: Output_2 is then arbitrated with the other pump requests to determine whether the pump should be turned ON or OFF.

**IF**

**THEN**

**ELSE**

**ELSE**

**ENDIF**

**ELSE**

**ENDIF**

**ELSE**

**ENDIF**

**ENDIF**

**ELSE**

**ENDIF**

**ENDIF**

### Model-Based ADD -- Process Specification

![Diagram](image1.png)

### Other Modeling Tools and Standards

- Both within our organization and within other parts of GM, various modeling tools and standards are employed.

- Regardless of particular tool and model type employed, standards for their use are of the utmost importance.
GM Powertrain Guidelines -
Example

- Guidelines have been written to address allowable and unacceptable model constructs, cleanliness, behavior, configuration management, etc.
- Guidelines ease transition from upstream and to downstream process executors (to model's "customers")
- Various applications (printed copy, rapid prototyping, verification against code, hand coding, autocoding) need to be considered

EXAMPLE:
- Blocks should be resized so their icons are visible and recognizable. Any text in the icon must be readable.

Algorithm Modeling
-- Without Using Algorithm Modeling Library Blocks / Guidelines --

Algorithm Modeling
-- Using Algorithm Modeling Library Blocks / Guidelines --
What is Rapid Prototyping?

An embedded controller consisting of a high speed processor(s) with additional I/O capability for advanced signal processing.

dPRAM
CAN
J1850
Ethernet (?)

Rapid Prototyping Vehicle Configuration

Control/Monitor Interface  Algorithm Development  Data Logging Interface

Control Desk  Variable Graphics  Control Desk Logging

Dual-port RAM HW  PC

Controller Interface  PCM  Real-Time Code Generation  Control Desk Logging

PCM  RPC  Ethernet  Parallel Port  Serial Port