Enabling Model-Based Design: Robust Collaborative Development of Embedded Systems

William P. Milam, Ford
Eileen Davidson, Ford
John Mills, SimuQuest
Agenda

• Introduction
• Brief overview of MBD
• MBD challenges due to complexity and scale
• Solution
• Questions
Introduction

- Data Definition Management: definitions of variables and parameters in model and resulting code
- Architecture Management: defining, managing instances of a reference architecture
  – interfaces, execution order & rate, etc.
Model-Based Design

- The use of control algorithm models and plant models to develop robust embedded control systems
- Relatively unconstrained solutions to fully implementable solutions via generated code for the target platform
Challenges

• Complexity: Typical powertrain application has 120 different features. Average of 36 applications released per model year.

• Scale: Over 200 developers. Some features/components come from module vendors.

• Not all artifacts are in Simulink/Stateflow
  – Plant models, legacy code, …
Exponential Data Complexity

Engine ECU (Per Model-Year)

- 36 APPLICATIONS
- 120 FEATURES
- 200 DEVELOPERS
- 10,000 SIGNALS
- 30,000 CAL PARAMS

... AND WITH CHANGING ARCHITECTURES, INTERFACES
Solution

• Client/server tool for managing sets of data definitions
  – definitions can be versioned
• Use same tool to define membership and interfaces for applications, sub-systems and features
  – Check instance of sub-system versus reference architecture description for sub-system
    • All inputs and outputs resolve within the sub-system
• Standalone so that it can be used with multiple modeling tools
  – Matlab, Modelica, AMESim, GTPower,…
Data definition management

• Manage data definitions
  – Version control
  – Recreate history of objects, releases
• Share data definitions across tools and developers enabling collaboration
• Fewer defects due to coordination of definition changes
Conceptual Overview

- Global Cooperative Editing
- Standalone Operation
Editing Workflow Overview

For Each Commit
• Who?
• What?
• Why?

Data Type: uint8 vs uint16

UniPhi Server

commit
update
update
revert
Pool Overview

Everything you need to simulate or generate code.

Server: 0..n Pools

API

Signals
Parameters
Models

Data Types
Abstractions / Reports
Feature / Rate Definitions

Tags
Links
Database Features Overview

See exactly what changed across multiple points in time.

MYxy PCM released

15 years ago

now
• Problem Statement for Architecture Process:
  – Feature interfaces not well documented outside of actual c-code or model implementation
    • No precise way to communicate interface requirements
    • No easy way to analytically predict interface incompatibilities prior to final build
  – Features can access any variable or parameter from any other feature
    • Organizations support multiple controls architectures, each containing a different set of features
    • Difficult for feature engineers to understand the various architectures they must deal with
A tool that provides a central repository for architecture artifacts, allowing engineers to make informed architecture decisions during the implementation of their features and subsystems.

These artifacts are:
- Reference Application Architectures
- Reference Subsystems (Reference Application Architectures are composed of Reference Subsystems)
- Reference Features (Reference Subsystems are composed of Reference Features)
Visual Aid for Architecture

Run-Time Support

Application

Sub-System

F F F F F

Sub-System

F F F F F

Sub-System

F F F F F

Sub-System

F F F F F
How does it work?

• The architecture team defines a *reference* representation of an architecture. The properties of a reference architecture are:
  – The set of Subsystems/Features defined in the architecture
  – The set of control signals (time and event-based triggers) available in the architecture
  – The set of I/O signals exchanged between subsystems/features

• The developers can then create instances of the reference architectures and compare to ensure compliance.
Reference vs Instance

Reference: Air-Path Management
- Throttle
- VCT
- Wastegate
- EGR

Instance: Air-Path Management A
- Throttle
- VCT
- Wastegate

Instance: Air-Path Management B
- Throttle
- VCT
- EGR
Illustration of Sub-System Analysis

Reference Subsystem
- Subsystem A
- Features
- Inputs
- Outputs
- Triggers

UniPhi

Feature Implementations
- Feature X
- Inputs
- Outputs
- Triggers
- Feature Y
- Inputs
- Outputs
- Triggers
- Feature Z
- Inputs
- Outputs
- Triggers

UniPhi

"Shaker Box"

Pass / Fail Report:
- Are feature(s) included in the subsystem?
- Are Feature subsystem interfaces satisfied?
UniPhi: Complexity Visualization

VISUALIZE WHAT I WANT ANY WAY I NEED TO SEE IT

e.g. SHOW ME ALL DEPENDENCIES FOR THIS SIGNAL
• The ability to manage and share sets of data definitions is essential to successfully support implementation of large scale MBD and eliminate errors.

• Architecture management is essential to enable early checking of interfaces and assemblies of features while in the modeling phase.

• Ultimately this leads to assembling models from components, architecture descriptions and data definitions.