Managing Software Interfaces of On-Board Automotive Controllers

Anthony Tsakiris
Ford Motor Company
atsakiri@ford.com
Introduction

Today’s cars rely heavily on software-intensive electronic controllers that share information.

A typical Ford vehicle has 20 controllers. Our high-end vehicles have about 40 controllers.

Increasingly, controllers must cooperate to deliver functions none can provide alone.
Hybrid Electric Power Pack

- High Voltage Battery + Controller
- Engine + Controller
- Motor + Generator + Gear Set + Controller
Business Goals

• Avoid duplication of engineering work
• Easily deploy new features and technologies
• Be quicker to market
• Reduce product and development cost
Scenario 1

Offer different power packs in a single vehicle
Scenario 2

Reuse a power pack in different vehicles
The Problem

We tried to reuse the hybrid electric power pack of a North American SUV in a European sedan that original had a conventional power pack. (scenario 2)

• Many unanticipated interface mismatches
• Lots of re-engineering
• Slow progress
• Project eventually cancelled

Two big causes:
• Interface variations
• Organizational constraints
Cause 1 - Variation

Optional features across large product line

Legacy designs from many brands

Ford  Mercury  Lincoln  Mazda  Volvo  Jaguar  Land Rover
Cause 2 - Organizational Constraints

We were not architecturally driven.

- Non-functional requirements have low priority
- Decisions often driven by immediate needs of an individual project
- Incremental change favored to reduce risk
- Collective mental model of system overwhelmingly focused on hardware view
- Different brands have different business models
A new project ... to eliminate the problems that caused the first project’s failure ...

- Six workstreams, one devoted to control architecture and interfaces
- A committed team of subject matter experts who can make change happen
- A new framework to guide our thinking
- Commonized control architecture and signal interfaces
Solution - A New Framework - Function

Vehicle Control Functions

Subsystem Control Functions

Internal details not shown for confidentiality reasons.

= function(s)
→ = data flow(s)
Solution - Interface Commonization

1160 interface signals collected from legacy designs

We created three categories:

- **standardized** for common, long-term use
- **restricted** for short-term use only
- **prohibited** to eliminate variation

- id
- name
- description
- range
- resolution
- enumerated values

- update rate
- initial value
- aliases
- transmitters
- receivers
- etc.
Results

393 redundant or undesirable signals eliminated

<table>
<thead>
<tr>
<th>Signal Category</th>
<th>Number of Signals</th>
<th>Fraction of All Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardized</td>
<td>392</td>
<td>34%</td>
</tr>
<tr>
<td>Restricted</td>
<td>99</td>
<td>8%</td>
</tr>
<tr>
<td>Prohibited</td>
<td><strong>393</strong></td>
<td><strong>34%</strong></td>
</tr>
<tr>
<td>TBD*</td>
<td>276</td>
<td>24%</td>
</tr>
<tr>
<td>Total</td>
<td>1160</td>
<td>100%</td>
</tr>
</tbody>
</table>

* Note: We have since addressed the TBD signals.
Established corporate data dictionary for all network interface signals, not just those affecting the power pack.

That means about 800 more signals.

<table>
<thead>
<tr>
<th>Signal Category</th>
<th>Number of Signals</th>
<th>Fraction of All Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardized</td>
<td>1200</td>
<td>60%</td>
</tr>
<tr>
<td>Restricted</td>
<td>311</td>
<td>15%</td>
</tr>
<tr>
<td>Prohibited</td>
<td>506</td>
<td>25%</td>
</tr>
<tr>
<td>Total</td>
<td>2017</td>
<td>100%</td>
</tr>
</tbody>
</table>
The Lessons Learned

#1 Illuminate the entire product line constantly. Establish a framework for thinking about the system that covers the entire product line. Make it the basis for integrating new features. This helps balance new features and old features.

#2 Keep the team. A permanent cross-functional team can provide continuous attention to non-functional quality attributes. Do not disband the team unless other mechanisms are established to take its place.
The Lessons Learned

# 3 Evolution works too.
Many organizational habits are deeply rooted. It is far more effective to evolve existing work products and processes instead of trying to revolutionize them.

#4 Do not wait for a complete architecture description.
A multi-view architecture description makes great sense but is difficult to establish in an organization that is not architecture-driven. Add value incrementally by working from implementation to architecture.
The Lessons Learned

#5 Educate.
Plan time and resources for education about architecture. Most of the engineering community does not think in terms of quality attributes, stakeholder concerns, architectural strategies, and multiple views.

#6 Be persistent; have patience.
Some changes take a very long time. Keep at it. Light pressure is better than no pressure.