Evaluation of Legacy Software Architecture

Michael Turner
May 9, 2018
Agenda

• Personal Introduction
• Corporate Introduction
• Automotive Software Primer
• Legacy Software Architecture Review
• Evaluation
• Conclusions
• Summary
• Q&A
Personal Introduction
Michael Turner’s Introduction

Education

- The Graduate Excellence Award Recipient, M.S.E. in Computer and Electrical Engineering, University of Michigan
- Graduated with Distinction, B.S.E. in Computer Engineering, Purdue University

Work experience

- Technical Fellow/Software Architect - Design of Cockpit solution architectures
- North American Regional Software Architect - Design of Driver Information products
- Driver Information Lead Software Engineer for instrument cluster projects
- Senior Software Engineer/Team Lead, Lear Corporation, for body control modules, generic electronic modules, and smart junction boxes
- Project Engineer, Stanley Air Tools, responsible for communication protocol research, development, and deployment

Core Expertise Areas:

- Solution and Software Architecture Design, Analysis, and Evaluation
- Agile Software Architecture Principles and Practices
- Risk and Cost Driven Architectures
- Technical Debt Control
- Architecture Deployment
- Architecting for Continuous Integration
Corporate Introduction
Industry-Leading Cockpit Electronics Product Portfolio

Only pure-play in automotive cockpit electronics

Visteon Market Position

<table>
<thead>
<tr>
<th>Top 5</th>
<th>Connected car Tier 1 supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>Instrument cluster displays</td>
</tr>
<tr>
<td>#2</td>
<td>Automotive display systems</td>
</tr>
<tr>
<td>#3</td>
<td>Head-up displays</td>
</tr>
</tbody>
</table>

Source: Rankings from 2016 ABI Research and IHS Markit.
Instrument Clusters

Shift toward connected cars and autonomous vehicles driving transition to all-digital instrument clusters

- Global market leader in digital instrument clusters
- Flexible and customizable OEM application co-development approach
- Platform approach delivers advanced technologies needed for an autonomous future
- Driver monitoring, ADAS integration and a virtualized instrument cluster domain
- Embedded functionality such as camera systems and ambient lighting allow for all new driver interaction experiences
Next-Gen Vehicle Cockpit Solution

Highly Integrated Cockpit Controller

- Single cockpit computer drives all displays and functions
- Integration of instrument cluster and infotainment as basic functions
- Advanced systems integrate HUD, driver monitoring and augmented reality
- Complex system integration of multiple operating systems on single multicore system-on-chip

Market Growth Rate
(Units in millions)

Visteon’s Market Leading Position

Data Source: Strategy Analytics, September 2017.

Visteon leads industry in fast-growing cockpit controller solutions
Leveraging Autonomous Driving Ecosystem

Technology Collaborations

Customer Partnerships

Strategic Investments

- DEEPScale
- STEER
- StradVision

Establishing an ecosystem with key partners

GAC MOTOR

Strategic cooperation agreement

American Center for Mobility

Only Tier 1 founder partner

Disciplined approach to R&D and strategic investments
Visteon by the Numbers

- $3.15B: 2017 annual sales
- 10,000: Employees
- 18: Countries
- 19: Manufacturing locations
- 18: Technical centers

Van Buren Township, Michigan, United States

A global leader in automotive cockpit electronics and software
Global Manufacturing Footprint

Europe
- Portugal: Palmela
- Russia: Vladimir
- Slovakia: Namestovo
- Tunisia: Bir El Bey

Asia Pacific
- China: Chongqing x2, Shanghai x2, Changchun x2, Xuzhou, Shaoxing
- India: Chennai
- Indonesia: Jakarta
- Japan: Hiroshima
- S. Korea: Yesan
- Thailand: Rayong

Americas
- Brazil: Manaus
- Mexico: Chihuahua, Reynosa

19 Manufacturing Locations
21,000 Unique Components
1 Million Products Per Week
1,000 Customer Locations
Global Engineering Footprint

**Europe**
- Bulgaria: Sofia
- France: Cergy
- Germany: Karlsruhe, Kerpen
- Portugal: Palmela
- UK: Chelmsford

**Asia Pacific**
- China: Shanghai x3
- India: Bangalore, Chennai, Pune
- Japan: Hiroshima, Yokohama
- S. Korea: Seoul

**Americas**
- Brazil: São Paulo
- Mexico: Chihuahua, Queretaro

- More than 50% Software Engineers
- 70% of Resources in Growth/Emerging Markets
- 9 Global Centers of Competence
- 134,000 Lines of Code Per Week
Automotive Software Primer
Deployment Perspective

<table>
<thead>
<tr>
<th>30 - 100+ ECUs in car</th>
<th>Consolidation of ECUs into domain controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reduces cost, weight, and power consumption</td>
</tr>
<tr>
<td></td>
<td>Leverages silicon and software innovations</td>
</tr>
</tbody>
</table>
Feature Trends

• Security
• Functional Safety
• OTA Reprogramming
• 3D Graphics
• Low Current
• High Speed Networking
• Driver Awareness
• Driver Monitoring
• Analytics
David McCandless, https://informationisbeautiful.net/visualizations/million-lines-of-code/
Product Deployment - Then and Now
Driver Information Software Architecture History

• Static View
• Data Flow
• Reusable Components
• Application Construction
• Hardware Deployments
• Architecture Principles
Static View

Application Feature Servers

Input Servers and Receive Servers

Infrastructure Feature Servers (Customer and Platform)

Infrastructure

Output Servers and Transmit Servers

Drivers
Reusable Components

• Software components assembled to build end-item software systems
• Components are implemented as feature servers which are configured to meet project requirements
• Components exist for numerous categories:
  • Startup
  • Checksum calculation
  • Input processing
  • NVM management
  • Manufacturing support
  • Stepper motor control
  • Fuel level processing
  • Chime management
  • Odometer handling
  • Telltale Management
  • Power Mode Management
  • Display Management
  • Warnings Management
  • Networking
  • Diagnostics
Application Construction

• Product software consists of:
  • Reusable components
  • Hardware-specific configuration of reusable components
    • MCU configuration
    • Product hardware configuration
  • OEM-specific configuration of reusable components
  • Application components

• Application components are comprised of servers and services to meet product functional requirements
Hardware Deployments

• Hardware deployment achieved via modification or configuration of infrastructure components

• Deployed to numerous hardware platforms:
  • Kepler I - Freescale MPC56xx MCU family
  • Einstein 2.x VIP – Freescale MPC56xx MCU family
  • Maxwell II – TI TMS470 MCU family
  • Newton 1.5 – Freescale S12 MCU family
  • Newton 2.0 – Renesas RL78 MCU family
  • Raman – Renesas 78K0 MCU family
  • Yukawa – Freescale S12X MCU family
  • McKinley 2 – Freescale S12
  • Matterhorn 2 – Freescale S08
  • K2 – Renesas V850
• Pre-platform deployments
  • Freescale HC11
  • Freescale HC12
Architecture Principles

• Simple Software System Design
  • Collection of servers, an infrastructure, and the kernel
  • Limited server types: input, output, receive, transmit, and feature.

• Pseudo-Client/Server Paradigm
  • Gen 1 component which supplies the service is the server.
  • Gen 1 component which uses the service is the client
  • Services are function-based.

• Task Scheduling
  • Non-preemptive, cooperative

• Usability
  • Stable and documented protocol specification through programmers guides
Evaluation
Evaluation – Entropy and Debt Analysis

• Gen 1 architecture was cornerstone of Driver Information software for 25 years.

• Over this time period, software entropy has increased and technical debt has accrued within the architecture, the core bookshelf, and the resultant applications.
  
  • Software Entropy
    • “the maintainability of a system may degrade over time due to continuous change” [1]
  
  • Technical Debt
    • “a metaphor for the trade-off between writing clean code at higher cost and delayed delivery, and writing messy code cheap and fast at the cost of higher maintenance efforts once it's shipped” [2]


Evaluation Metrics

• Entropy Metrics
  • File-based impact
  • Interface-based impact
  • CM-based impact

• Debt Metrics
  • Compatibility of current architecture, design, or implementation
  • Effort to meet with proposed architecture, design, or implementation
  • Stakeholder priorities
Evaluation – Entropy Categories

• **Architecture Entropy**
  - Integration of graphics support
    - 1900 files affected
    - 2100 interfaces affected
  - Integration of network framework
    - 110 files affected
    - 550 interfaces affected

• **Infrastructure Entropy**
  - Design entropy
    - Introduction of memory management components for QSPI and DMA
      - 498 files affected
      - 451 interfaces affected
  - CM Package entropy
    - Packages with narrow functional focus (15%) 
    - Packages with limited project integrations (10%)
    - Obsolete packages (10%)

• **Application Entropy**
  - New applications have migrated away from server and services concepts
  - 5% adhere to design paradigm
## Stakeholder Requirements Analysis

<table>
<thead>
<tr>
<th>Requirement #</th>
<th>Requirement Description</th>
<th>ASR</th>
<th>Platform Req</th>
<th>Stakeholder Priority</th>
<th>Met with in-house Gen 1</th>
<th>Effort to meet completely with modified Gen 1</th>
<th>TD Factor</th>
<th>Met with SWA Gen2</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Horizontal/Logical Scalability</td>
<td>Y</td>
<td>Y</td>
<td>L</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>002</td>
<td>Vertical/Physical Scalability</td>
<td>Y</td>
<td>Y</td>
<td>L</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>003</td>
<td>Graphics Development Tool support</td>
<td></td>
<td>Y</td>
<td>M</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>004</td>
<td>Graphics framework</td>
<td></td>
<td>M</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>UI logic support</td>
<td>L</td>
<td>Y</td>
<td>L</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>006</td>
<td>Diagnostics Logging</td>
<td>Y</td>
<td>H</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>007</td>
<td>Diagnostics Tracing</td>
<td>Y</td>
<td>H</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>Testability of software</td>
<td>Y</td>
<td>H</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>009</td>
<td>Reliability of software</td>
<td>Y</td>
<td>H</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>Reusability of software</td>
<td>Y</td>
<td>H</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>Abstraction from MCU HW</td>
<td>Y</td>
<td>H</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>012</td>
<td>Abstraction of components from component communication protocol</td>
<td>Y</td>
<td>L</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>013</td>
<td>Abstraction of gauge software from stepper/UI</td>
<td>Y</td>
<td>L</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>014</td>
<td>Tasking Model support</td>
<td>Y</td>
<td>H</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>Shafted stepper motor support</td>
<td>Y</td>
<td>M</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>Shaftless stepper motor support</td>
<td>Y</td>
<td>M</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017</td>
<td>2D animations at 30 fps</td>
<td>Y</td>
<td>M</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using classification concepts from [1], determined that 62 of 74 improvements would be considered as technical debt. Others would be considered as “non-TD” improvement actions.

Allowed the organization to consider the proper priority and team decomposition.

Evaluation – Technical Debt Categories

• Architecture Debt
  • Debt analysis based upon stakeholder requirements
  • Principal incurred due to:
    • Cooperative scheduling
    • Point-to-Point communication (defined as services in Gen 1)
    • Tight coupling of components
    • Lacking attributes:
      • Testability
      • Monitorability
      • Scalability
      • Portability
  • Interest incurred with increase in product complexity:
    • Graphics requirements
    • 3rd party software integration
    • Security
    • Safety
Evaluation – Technical Debt Categories (continued)

• Infrastructure Debt
  • Principal
    • Tight coupling to Gen 1 architecture
    • Tight coupling to kernel
  • Interest
    • Ambiguous boundary between infrastructure and application components
      • It is not sufficient to define infrastructure based only upon reuse
      • Effort to port to a new application should be considered
    • Functional decomposition granularity does not meet expectations of modern products
    • Limited use of model-driven development
Evaluation – Technical Debt Categories (continued)

• Application Debt
  • Principle
  • Design
    • Application components do not follow a consistent design pattern
    • Applications functionality is decomposed to match functional requirements, not a software design specification
  • Performance
    • Run-time performance of application components is not considered until test phase
  • Size
    • Memory footprint of application components is not considered until build phase
  • Interest
    • Extensibility
      • New requirements are difficult to efficiently implement or distribute
        • Synchronization between the display, chimes, and telltales
        • Power Management
Conclusions
Conclusions

• Gen 1 could not be extended to meet the objectives of the next generation architecture:
  • Achieve the stakeholder requirements
  • Possess the derived quality attributes
  • Reduce the software entropy
  • Reduce the technical debt

• Revolution, not evolution, was required.
Summary
Summary

• Analysis of legacy systems requires systematic approach to avoid opinion-based conclusions
  • Team of architects, designers, and developers must contribute in the analysis to ensure proper parameters and techniques are selected
  • Product knowledge is crucial in evaluating criteria
    • For example, for some systems, the number of files affected may not be significant for entropy.

• Not all debt and entropy are the same
  • Categories and classifications of debt and entropy can assist in assigning teams to the actions

• Helpful thought: Entropy is often the side affect of debt

• We must strive to ask why debt or entropy exists before deciding how to resolve.
Thank you