Impact of Functional Safety on Software Architecture

Michael Turner – Software Architecture Technical Fellow

May 7, 2019
Agenda

• Personal Introduction
• Corporate Introduction
• Presentation Goals
• Functional Safety Primer
• Software Architecture Safety
• Conclusion
• 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

Rapidly growing in infotainment and domain controller solutions

Source: Rankings from 2016 ABI Research and IHS Markit.

Visteon Market Position

<table>
<thead>
<tr>
<th>Top 5</th>
<th>Connected car Tier 1 supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Digital clusters</td>
</tr>
<tr>
<td>#2</td>
<td>Center stack displays</td>
</tr>
<tr>
<td>#3</td>
<td>Head-up displays</td>
</tr>
</tbody>
</table>

Source: Rankings from 2016 ABI Research and IHS Markit.
Presentation Goals
Presentation Goals

- Introduction to functional safety
- Relating functional safety to software architecture
Functional Safety Primer
What is Functional Safety?

- Specifics are dependent upon the industry
  - ISO 26262 - Road Vehicles Functional Safety
  - IEC EN 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems
  - Ministry of Defence Defence Standard 00-56 - Safety Management Requirements for Defence Systems
  - US RTCA DO-178C - Software Considerations in Airborne Systems and Equipment Certification
  - US RTCA DO-254 - Design Assurance Guidance for Airborne Electronic Hardware
  - ISO 25119 - Tractors and machinery for agriculture and forestry -- Safety-related parts of control systems
  - IEC 62304 - Medical device software – Software life cycle processes
  - IEC 61513:2013 - Nuclear power plants – Instrumentation and control important to safety – General requirements for systems
  - IEC 61226:2009 - Nuclear power plants – Instrumentation and control important for safety – Classification of instrumentation and control functions
  - IEC/TR 61838:2010 - Use of Probabilistic Safety Assessment for the Classification of Functions
What is Functional Safety?

- Absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems (ISO 26262-1:2018, Definition 1.51)
- Unreasonable risk is risk judged to be unacceptable in a certain context according to valid societal moral concepts (ISO 26262-1:2018, Definition 1.136)
- Risk is the combination of the probability of occurrence of harm and the severity of that harm (ISO 26262-1:2018, Definition 1.99)
- Harm is physical injury or damage to the health of persons (ISO 26262-1:2018, Definition 1.56)
- A hazard is the potential source of harm caused by malfunctioning behavior of the item (ISO 26262-1:2018, Definition 1.57)
- Malfunctioning behavior is failure or unintended behavior of an item with respect to its design intent (ISO 26262-1:2018, Definition 1.73)
ISO 26262 Concepts

• Automotive Safety Integrity Level (ISO 26262-3:2018)
  • ASIL = f(Severity, Exposure, Controllability) (ISO 26262-3:2018, Table 4)
  • Thresholds for ASIL D, C, B, and A are not prescriptive in standard
  • QM (Quality Management) – No safety mechanisms are required

• Freedom from Interference (ISO 26262-6:2018, Annex D)
  • Interference is the presence of cascading failures from a sub-element with no ASIL assigned, or a lower ASIL assigned, to a sub-element with a higher ASIL assigned leading to the violation of a safety requirement of the element
  • A cascading failure is the failure of an element of an item causing another element or elements of the same item to fail.
  • Fault categories
    • Timing and Execution
    • Memory
    • Exchange of Information
ISO 26262 Concepts

• Timing and Execution (ISO 26262-6:2018, Section D.2.2)
  • Faults
    • Blocking of execution, deadlocks, livelocks, incorrect allocation of execution time
  • Protection Mechanisms
    • Cyclic execution scheduling, fixed priority based scheduling, time triggered scheduling, monitoring of processor execution time, program sequence monitoring and arrival rate monitoring
ISO 26262 Concepts

• Memory (ISO 26262-6:2018, Section D.2.3)
  • Faults
    • Corruption of content, stack overflow/underflow, read or write access to memory allocated to another software element and for registers
  • Protection Mechanisms
    • Memory protection, parity bits, error-correcting code (ECC), cyclic redundancy check (CRC), redundant storage, restricted access to memory, static analysis of memory accessing software and static allocation
ISO 26262 Concepts

• Exchange of Information (ISO 26262-6:2018, Section D.2.4)
  • Faults
    • Repetition/loss/delay/insertion/incorrect addressing of information
  • Protection Mechanisms
    • Communication protocols, information repetition, loop back of information, acknowledgement of information, appropriate configuration of I/O pins, separated point-to-point unidirectional communication objects, unambiguous bidirectional communication objects, asynchronous data communication, synchronous data communication, event-triggered data buses, event-triggered data buses with time-triggered access, time-triggered data buses, mini-slotting and bus arbitration by priority
ISO 26262 Concepts

- **Modeling and coding guidelines** (ISO 26262-6:2018, Section 5)
  - Goals: Unambiguous/comprehensive definitions, modularity/abstraction/encapsulation, structured constructs
  - Methods: Low complexity, language subsets, strong typing, defensive programming, style guides, naming conventions

- **Notations for software architecture design** (ISO 26262-6:2018, Section 7.4.1)
  - Goals: Comprehensibility, consistency, simplicity, verifiability, modularity, abstraction, encapsulation, maintainability
  - Methods: Use of natural language, informal notations, semi-formal notations, formal notations

- **Principles for software architecture design** (ISO 26262-6:2018, Section 7.4.3)
  - Goals: Comprehensibility, consistency, simplicity, verifiability, modularity, encapsulation, maintainability
  - Methods: Component structure/size/complexity/cohesion/coupling/spatial isolation, interface size, scheduling

- **Mechanisms for error detection at the software architecture level** (ISO 26262-6:2018, Section 7.4.12)
  - Goals: Detect dependent failures in support of freedom from interference
  - Methods: Data range checks, plausibility checks, error detection codes, temporal/logical program execution monitoring, design redundancy

- **Mechanisms for error handling at the software architecture level** (ISO 26262-6:2018, Section 7.4.12)
  - Goals: Address the effects of failures in support of freedom from interference
  - Methods: Safe state, graceful degradation, access permission

- **Methods for the verification of the software architecture design** (ISO 26262-6:2018, Section 7.4.14)
  - Goals: Architecture design satisfies ASIL requirements with evidence
  - Methods: Design walk-through/inspection/simulation, control flow analysis, data flow analysis, scheduling analysis
Software Architecture Safety
Safety as a Quality Attribute

• One possible view is that functional safety has a bi-directional dependency with other attributes of the architecture. For example, software that is not available cannot be safe. [1]

• Another view is that safety is a quality attribute with tactics that overlap with availability. [2]

• Regardless of the view, the tactics are the same.


Safety Architectural Tactics

• Using ISO 26262 as a reference, the architectural tactics to achieve functional safety include:
  • Monitor (Availability)
  • Heartbeat (Availability)
  • Active Redundancy (Availability)
  • Rollback (Availability)
  • Degradation (Availability)
  • Increase Cohesion (Modifiability)
  • Reduce Coupling (Modifiability)
  • Introduce Concurrency (Performance)
  • Maintain Multiple Copies of Computations (Performance)
  • Maintain Multiple Copies of Data (Performance)
  • Schedule Resources (Performance)
  • Verify Message Integrity (Security)
Conclusion
Conclusion

• Functional safety expands the notion of safety already known to software architects by applying safety to a system.
• The software architects’ toolbox already contains the tools (i.e., tactics) needed to address requirements of functional safety.