OVERVIEW OF GM RESEARCH OF METHODS AND TOOLS FOR TIMING/DEPENDABILITY/COST ANALYSIS

Paolo Giusto, Arkadeb Ghosal
GM ECI Lab – Advanced Technology Office
Palo Alto, CA, USA

Mark McKelvin
UC Berkeley – EECS
Outline

- Background
- Methods and Tools for Dependability Analysis
- Methods and Tools for Cost Analysis
- GM Example of Application of Methods and Tools for Timing Analysis and Optimization
Background
Active Safety Applications

Forward Vision System
- Lane tracking
- Object detection
- Far IR capability

Rear Vision System
- Object detection
- Far IR capability

Enhanced Digital Map System

Short-Range Sensors
- Short-Range Blind-Spot Sensors

Long-Range Sensors
- Wireless: V2X communication

Camera
- Image Process
- Sensor Fusion
- Feature 1
- Feature 2
- Feature 3
- Feature 4

GM
Latency and Jitter Constraints

Lane Keeping Feature Decomposition

90ms < L < 100 ms
Issues w/ Current Design

Processes

- Current and near-future in-vehicle architectures are CAN-based
  - Cost Effective
  - Relatively Low Bandwidth
  - Non-Deterministic Timing Behavior (Safety??)

- FlexRay Time-Triggered architectures are coming
  - Relatively High Bandwidth (Ethernet ?)
  - Quasi-Deterministic Timing Behavior (non-deterministic failures are possible!)
  - Fail Safe (not Fail Operational…Active Safety??)

- The verification of the non-deterministic timing behavior of the system is performed late in the development process while the architecture decisions are frozen early

- We need early exploration and late binding as opposed to early binding and late verification
Timing Analysis Framework

**Worst/Best Case Analysis**
- Software Task Attributes (T, ET, P, O)
- Message Attributes (T, S, P, O)
- SymTA/S
- Schedulability Analysis
- Worst/Best Case Bounds

**Probabilistic Analysis**
- Commercial Tool
  - Simulation
  - Excel
  - Statistical Analysis
  - Stochastic Analysis
- In-House Tool
- Probability Distributions

**Results**
- CPU/BUS Utilization
- Software Task Response Time (w/ Jitter)
- Message Latency (w/ Jitter)
- End to End Latency (w/ Jitter)
- SymTA/S + Other Tools
- Software Task Output Jitter
- Message Output Jitter
Timing Analysis/Optimization Flow

1. Schedulability Analysis
   - Timing Requirements Satisfied?
     - Yes
     - No

2. Optimization
   - SymTA/S

3. Probabilistic Analysis
   - Other tools

Allocation/Configuration:
- Functional Model
- Architecture Model

SymTA/S:
- #0 Allocation/Configuration
- #1 Schedulability Analysis
- #2 Optimization
- #3 Probabilistic Analysis

Probability Distributions
**Dependability Analysis Framework**

- **FTA** (Fault Tree Analysis)
  - Top Event
  - Undetected Primary Fault
    - Primary Fault
    - Safety Mechanism

- **Failure Model Library**
  - Failure Modes & Assoc. Failure Rates and/or Failure Classes For Typical Components

- **Diagnostic Coverage Library**
  - GM Diagnostics for Failure Modes w/ Level of Coverage Provided

- **PHA** (Potential Hazards Analysis)

- **System Model**

- **Reliability**
  - Mean Time to Failure
  - Failure Frequency
  - Unreliability
  - Mean Time between Failure
  - Mean Downtime
  - Mean Time to Repair
  - Unavailability
  - Total Downtime
  - Total Uptime
  - Mean time to Critical Fault
  - Mean Time between Critical Failures
  - Steady-State Safety
  - Mean Time to Unsafe Failures
  - Quantitative FTA
  - Cut-set Analysis
  - Single Point Fault Metric
  - Latent Fault Metric

**Source:** Joseph D’Ambrosio
Methods and Tools For Dependability Analysis

Mark Mc Kelvin (UCB), Arkadeb Ghosal (GM), Joseph D’Ambrosio, Paolo Giusto (GM)

Slides from SAE 2009 presentation by Mark McKelvin
Fault Tree Analysis (FTA)

- A top-down approach to failure analysis starting with a potential hazard to avoid (top event) and determines event (fault) combinations that may lead to the top event
  - Logic gates define Boolean relationships between events
  - Basic events are initiating, atomic events

- Uses
  - Identify potential low-level causes of critical hazards and determine if adequate hazard controls are applied
  - Design, verification/validation, investigation
  - Existing tools: FaultTree+, Item Toolkit, SAPHIRE, Galileo
FTA Main Steps

• Define system boundary, conditions, and top event
• Construct the fault tree (manual or automatic)
• Analyze the fault tree
  • Qualitative: identifies combinations of events leading to top event
  • Quantitative: frequency and unavailability of top event, event sensitivity
Automotive Steer-by-Wire Application

- GM Sequel experimental vehicle
  - Supports X-by-wire applications
  - Distributed set of host controllers
  - FlexRay and CAN communication network
Application Model

• Steer-by-wire application model
  • Periodic, time triggered control algorithm
  • Static task scheduling
  • Data flow specification

• Fault tolerant requirement
  • Tolerate single point architecture failures under fail silent assumption
Application is captured in a flexible, XML data model.

- Function Definition
  - $f \in F$, set of functional tasks
  - $i \in I$, set of input ports
  - $o \in O$, set of output ports
  - $n = <o,i> \in N$, set of nets
  - $G = (F, N)$

- Architecture Definition
  - $r \in R$, typed resources (i.e. processor, bus)
    - with attributes (i.e. failure rate, cost)

- Mapping
  - $<n,r> \in N_M$
  - $<f,r> \in F_M$
  - $G_M = (F_M, N_M)$
Model to Fault Tree Translation

top event

- Captured model in XML data model
Model to Fault Tree Translation

\[ f, \text{ event} \]

\[ r_1, \text{ event} \]

\[ \text{OR} \]

\[ \text{Ports event} \]

\[ \text{OR} \]

\[ p_1, \text{ event} \]

\[ n_2, \text{ event} \]

\[ \text{AND} \]

\[ n_1, \text{ event} \]

\[ n_3, \text{ event} \]

\[ \text{AND} \]

\[ r_{n2}, \text{ event} \]

\[ f_2, \text{ event} \]

\[ \text{AND} \]

\[ f_1, \text{ event} \]

\[ r_{n3}, \text{ event} \]

\[ f_3, \text{ event} \]

\[ \text{AND} \]

\[ f, r_1 \]

\[ n_2, r_{n2} \]

\[ n_3, r_{n3} \]

\[ n_1, r_{n1} \]

\[ p_2 \]

\[ p_1 \]

\[ p_3 \]

\[ n_5, r_{n5} \]
Model to Fault Tree Translation
Model to Fault Tree Translation

![Fault Tree Diagram]

- $f$, event
- $r_1$, event
- OR
- Ports event
- $n_1$, event
- AND
- $r_{n1}$, event
- $f_1$, event
- $f_2$, event
- $f_3$, event
- OR
- $n_2$, event
- AND
- $r_{n2}$, event
- $n_3$, event
- AND
- $r_{n3}$, event
- AND
- $n_4$, event
- AND
- $n_5$, event
- $r_{n4}$, event
- $n_6$, event
- AND
- $n_{n5}$, event
- AND
- $r_{n6}$, event
- AND
- $f$, event
- $r_1$, event
- OR
- $n_1$, event
- AND
- $r_{n1}$, event
- $p_1$, event
- OR
- $n_2$, event
- AND
- $r_{n2}$, event
- $p_2$, event
- OR
- $n_3$, event
- AND
- $r_{n3}$, event
- $p_3$, event
- function/actuator

2009-01-1377
Model to Fault Tree Translation
Model to Fault Tree Translation

basic event

f, event

OR

r₁, event

sensors

f, r₁

p₁

n₁, rₙ₁
Experimental Results

- **Data sources**
  - Exponential failure distribution
  - Commercial ground vehicle

- **Architectures evaluated**
  - Architecture 1 – one ECU, no functional task redundancy
  - Architecture 2 – two ECUs, same tasks on each
  - Architecture 3 – three ECUs, same tasks on each
  - Architecture 4 (baseline) – four ECUs, same tasks on each

---

**Table 1: Estimated Failure Rates for Architecture Resources in the Case Study**

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Failure Rate, $\lambda$ (failures/hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copper Wire</td>
<td>$1.0 \times 10^{-7}$</td>
</tr>
<tr>
<td>Sensor Type 1</td>
<td>$6.06 \times 10^{-4}$</td>
</tr>
<tr>
<td>Sensor Type 2</td>
<td>$6.06 \times 10^{-5}$</td>
</tr>
<tr>
<td>Processor (ECU)</td>
<td>$6.28 \times 10^{-5}$</td>
</tr>
<tr>
<td>Motor (Actuator)</td>
<td>$7.90 \times 10^{-7}$</td>
</tr>
<tr>
<td>CAN Bus</td>
<td>$2.6 \times 10^{-7}$</td>
</tr>
<tr>
<td>FlexRay Bus</td>
<td>$8.75 \times 10^{-4}$</td>
</tr>
</tbody>
</table>
Validation

- Used minimal cut sets of baseline design
- By inspection, checked that minimal cut steps cause top event
- Does it tolerate single point failures? Yes!
Reliability of various architectures from using initial sensor failure rate estimates (1) and by improving sensor failure rate estimates (2) based on basic event sensitivity results.
Automatic Fault Tree Generation

Architecture Description → Resources → Mapping → Fault Tree Generation

Behavior

Function Description

Data Model

Currently available

Galileo

Free academic tool

Under progress

FaultTree+

Robust Commercial Tool
Methods and Tools For Cost Analysis

Arkadeb Ghosal, Sri Kanajan, Randall Urbance (GM), Alberto Sangiovanni-Vincentelli (UCB)

Slides from SAE 2008 presentation by Arkadeb Ghosal
Motivation

- Rigorous cost models for initial design phase
- Effect of design decisions on cost of design life-cycle
- Use of a standard model to evaluate monetary cost
- Lack of understanding for cost of product line alternatives
- Evaluation of reuse vs. modularity
- A cost model that captures the impact of design decisions at the system level for a ECS product line architecture
Related Work

Technical Cost Modeling

- Evaluate cost of early product designs and new processing options
- Illustrate how cost drivers change when considering alternative part designs, materials, processes and product architectures
- Not exact price model but is an unbiased way to compare architectures
- Fixed cost / variable cost

Architecture Trade-off Studies

- Main purpose of the model is to allow system engineers to compare different solution alternatives with respect to cost, in order to perform an early optimization
- Total cost includes product cost, i.e. the cost of hardware components, hardware development, and software development

Targeted to manufacturing, not Electronic and Control Systems

Targeted to general embedded systems, not automotive systems
What are the specifications, functions and requirements?

What is the cost incurred on the design due to the specifications passed from the concept?

How many modules? Wire-length? Weight? placement?

How many sub-components? Special packaging?

What is the cost incurred on the assembly due to the topological specifications passed from the design?

What is the cost incurred on the part fabrication due to change in IO mapping?
Cost breakdown

- In-Service Cost
- Integration/ Assembly
- Part Fabrication
- Design & Development (Hardware)
- Design & Development (Software)

- Internal Fixed
- External Fixed
- Variable Internal
- Variable External
Product Line Architecture

To Reuse or Not To Reuse
Architecture Instance

- Software modules
  - Function points
  - Kilo lines of code per function point
  - Complexity
    - Memory constraint
    - Timing constraint
    - Virtual machine volatility
    - Turnaround time
  - Other COCOMO factors
    - Product
    - Project
    - Personnel
  - Newness
    - Off-the-shelf
    - Developed from scratch

- Hardware modules
  - Number of instances
  - Specification effort
  - Validation effort
  - Set of components
    - Component
    - Number of instances
  - Packaging complexity
  - Newness

- Number of Cut-leads

- Flash Time
Cost Model

■ Software development cost
  ● Cost of development effort
  ● Cost of part maintenance

■ Hardware development cost
  ● Specification cost
  ● Validation cost
  ● Package design cost
  ● Part maintenance cost

■ Part Fabrication cost
  ● Cost of module
    – Component cost
  ● Interconnection cost
    – Cut-lead cost

■ Integration Assembly Cost
  ● Flash Cost
Active Safety Sub-system

Passive Safety
- Reduce the effects of an accident
- Airbags, seat belts and strong body structures

Active Safety
- Automatic reaction to threat and ensures safe conditions
- Adaptive cruise control, lane keeping and automatic crash preparation

The case study focuses on studying the cost of alternative design decisions for network architecture
Functionality and Architecture

- Left sense
  - Detect tracked objects
  - Camera
    - Detect tracked objects
  - Right sense
    - Detect tracked objects

Object fusion

Active safety/ features

- Brake
- Throttle
- Steer

- Short range radar left
- Camera sensor (CAM)
- Short range radar right

CAN bus

Object Detection Fusion, Active Safety Feature (Component Control Module – CCM1)

Object Detection Fusion, Active Safety Feature (Component Control Module – CCM2)

CAN bus

Brake actuator
Throttle actuator
Steer actuator
# Product Line and Alternatives

<table>
<thead>
<tr>
<th>packages</th>
<th>features</th>
<th>vol</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>f1, f2, f3</td>
<td>300</td>
</tr>
<tr>
<td>p2</td>
<td>f1, f2, f3, f4, f5, f6</td>
<td>450</td>
</tr>
<tr>
<td>p3</td>
<td>f5, f7</td>
<td>210</td>
</tr>
<tr>
<td>p4</td>
<td>f5, f7, f8, f9</td>
<td>150</td>
</tr>
<tr>
<td>p5</td>
<td>f5, f8, f9</td>
<td>210</td>
</tr>
<tr>
<td>p6</td>
<td>f1, f2, f3, f4, f5, f6, f7</td>
<td>90</td>
</tr>
<tr>
<td>p7</td>
<td>f5, f7, f8, f9</td>
<td>150</td>
</tr>
</tbody>
</table>

- Processor component: new vs. off-the-shelf
- Number of CCM ECUs: multiple vs. integrated
- CAM sensor: standalone vs. integrated with CCM ECU
Architecture Alternatives

Baseline
Alternative 2
Alternative 1
Alternative 3

CCM1 (off-the-shelf components)
CCM2 (off-the-shelf components)
CCM1 (new processor components)
CCM2 (new processor components)

Integrated CCM (new processor components)
CCM1 (new components)
CAM + CCM2 (new components)
A single CCM ECU with a new processor component and an independent ECU for CAM.
Cost Comparison

- Key cost factors considered are development cost (software and hardware modules), and parts cost.
- Piece cost for an ECU is computed from the type of CAN connections, number of CAN transceivers, PCB size, memory type and size, CPU type and connector type.
- Cost figures are not absolute - differences in architectural elements have been accounted assuming linear cost model.

<table>
<thead>
<tr>
<th>Cost</th>
<th>Baseline</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>128.3</td>
<td>79.8</td>
<td>123.4</td>
<td>79.7</td>
</tr>
<tr>
<td>Dev.</td>
<td>3.4</td>
<td>4.3</td>
<td>3.7</td>
<td>7.0</td>
</tr>
<tr>
<td>Total</td>
<td>131.7</td>
<td>84.1</td>
<td>127.1</td>
<td>86.7</td>
</tr>
</tbody>
</table>
## Analyzing the cost

<table>
<thead>
<tr>
<th>Cost</th>
<th>Baseline</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>128.3</td>
<td>79.8</td>
<td>123.4</td>
<td>79.7</td>
</tr>
<tr>
<td>Dev.</td>
<td>3.4</td>
<td>4.3</td>
<td>3.7</td>
<td>7.0</td>
</tr>
<tr>
<td>Total</td>
<td>131.7</td>
<td>84.1</td>
<td>127.1</td>
<td>86.7</td>
</tr>
</tbody>
</table>

COTS modules used by the baseline are more expensive than the custom made ECU used in Alternative 1.
Analyzing the cost

<table>
<thead>
<tr>
<th>Cost</th>
<th>Baseline</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>128.3</td>
<td>79.8</td>
<td>123.4</td>
<td>79.7</td>
</tr>
<tr>
<td>Dev.</td>
<td>3.4</td>
<td>4.3</td>
<td>3.7</td>
<td>7.0</td>
</tr>
<tr>
<td>Total</td>
<td>131.7</td>
<td>84.1</td>
<td>127.1</td>
<td>86.7</td>
</tr>
</tbody>
</table>

A modular architecture where only one lower capacity (and cheaper) ECU is required for the lower end packages, contributes to a overall lower cost in comparison to integrated ECU.
**Analyzing the cost**

<table>
<thead>
<tr>
<th>Cost</th>
<th>Baseline</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>128.3</td>
<td>79.8</td>
<td>123.4</td>
<td>79.7</td>
</tr>
<tr>
<td>Dev.</td>
<td>3.4</td>
<td>4.3</td>
<td>3.7</td>
<td>7.0</td>
</tr>
<tr>
<td>Total</td>
<td>131.7</td>
<td>84.1</td>
<td>127.1</td>
<td>86.7</td>
</tr>
</tbody>
</table>

Alternative 3 is very close in terms of parts cost, but has a larger design and development cost due to the complexity in integrating the CAM ECU with the CCM ECU.
Effect of Package 1 volume

Alternative 1 is the winner in lower volumes; difference with Alternative 3 vanishes as the volume is increased.
**Effect of Package 3 volume**

Alternative 1 is the winner in lower volumes; difference with Alternative 3 vanishes as the volume is increased.
Effect of discount rate of CCM

Cost of Alternative 1 reduces faster than other alternatives
Variation of package 1 volume and discount rate of CCM ECU

Discount Rate vs. Package 1 Volume vs Total Product Line Cost

Alternative 1 wins at lower discount and lower volume;
Alternative 3 is wins at higher discount and higher volumes
Variation of package 3 volume and cost of CCM ECU

Alternative 3 wins at lower discount and lower volumes;
Alternative 1 wins at higher discount and higher volumes
Winning Choice – Alternative 1

- Lowest total product-line piece cost
- Favorable sensitivity to changes in package volume and piece cost
- Most modular architecture among all the alternatives
  - Alternative 2 (integrated solution, less modularity) had significant give-away cost that made it more costly for low end packages
  - Baseline architecture (equivalent in modularity to Alternative 1) used components over designed relative to the requirements.

- Robust to changes in CCM ECU cost
  - Lowest cost for discount $< 1.0$ which is practical as discount $> 1.0$ means that the piece cost increases as volume increases.
Challenges and Future Work

- Lack of data
- Lack of openness
- Lack of records
- Lack of process model

- In-service cost
- Technology Evolution
- Architecture Evolution
- Information gap
Example

Paolo Giusto, Arkadeb Ghosal, Haibo Zeng (GM NA), Swarup Mohalik (GM India), Mohammed A Yousuf, James K Thomas, (GMNA Software and Controls)
Active Safety Module

- Fail Safe Fault Tolerant Strategy

- Dual Core Processor Architecture

2 Paths
- Primary Path from Image & Radar Processors (via CAN) generates messages to BCM (via CAN)
- Secondary Path provides confirmation command or warning to driver
Multi rate Modeling

Tasks activated periodically. Data propagated using SymTA/S Registers.

End to End Latency

- “Max Age” Semantics
- “First Through” Semantics

Source: Kai Richter - SymTAVision

3rd SymTA/S Conference on Industrial Timing Analysis
Topological vs. Scheduling Cycles

Simulation: [Source2,[Task1,Task3]^6, Task2, Sink1]
Schedulability Analysis: [Source2,Task1,Task2, Sink1]
Primary Path (for illustration purposes)

3rd SymTA/S Conference on Industrial Timing Analysis
Analysis/Optimization

Objectives

- To compute the end to end latency of the primary and secondary path
- To minimize the two latencies (<100ms)
- To minimize the difference between the two latencies (<10%)
- By changing Task Offsets and Priorities
Assumptions

- Multi rate Synchronized Execution Model:
  - No Task Activated by Predecessor
  - Periodic Tasks w/ Priorities, known Execution Times, and Offsets
  - CAN Synchronized Message w/ Priority and Payload
  - CAN Un-Synchronized Messages w/ Priorities and Payloads

- Task Execution Times:
  - Uniform distributions with range [MAX/2, MAX]

- SPI bus
  - 2 pairs of periodic TX/RX tasks

- Scheduling
  - Static priority preemptable tasks, Static priority CAN messages (no-preemption, blocking considered)

- No shared variables between tasks
  - Critical regions blocking delays not modeled
Calculation of End to End Latencies (Worst & Best Case)

Worst/Best Case Task to Task Latency (+ Jitter)

Worst/Best Case Message Response Time (+ Jitter)

- TaskCAN_TX_1 and msg320 are synchronized!
- No Sampling Delay between Task_CAN_TX_1 and msg320
- Msg320 response time computed assuming un-synchronized senders
Calculation of End to End Latencies (Probabilistic Case)

PDF of Task to Task Latency (via simulation using random task execution times)

Convolution is performed on pdf's. CDFs are shown for illustration purposes.

\[ pdf = pdf_{e2etasks} \oplus pdf_{resp\_time\_msg} \]

\[ CDF = P(X <= x) = \int_{o}^{x} pdf(t)dt \]
Optimization Results

- Two step-process
  - Applied analysis/optimization flow to original design, then
  - Changed design and reapplied the flow
- Message response time is invariant in original and new design
  - Not explored optimizations at the bus level

<table>
<thead>
<tr>
<th></th>
<th>Msg320</th>
<th></th>
<th>Msg321</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Case</td>
<td>.168</td>
<td>Worst Case</td>
<td>6.314</td>
</tr>
<tr>
<td>Worst Case</td>
<td>6.524</td>
<td>Best Case</td>
<td>.168</td>
</tr>
</tbody>
</table>
Optimization Results (cont’d)

- **Original Design**
  - W/B Case Latencies
  - Statistics

- **Optimized Original Design**
  - W/B Case Latencies
  - Constraint on Latency (<100ms)

<table>
<thead>
<tr>
<th></th>
<th>Secondary</th>
<th>Primary</th>
<th>Secondary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Case</td>
<td>80.388</td>
<td>110.398</td>
<td>81.544</td>
</tr>
<tr>
<td>Worst Case</td>
<td>132.884</td>
<td>262.884</td>
<td>79.334</td>
</tr>
<tr>
<td>Mean Latency</td>
<td>94.33</td>
<td>142.39</td>
<td>48</td>
</tr>
</tbody>
</table>
Optimization Results (cont’d)

- **New Design**
  - W/B Case Latencies

- **Optimized New Design**
  - W/B Case Latencies
  - Constraint on latencies (<50ms)
  - Statistics

<table>
<thead>
<tr>
<th></th>
<th>Secondary</th>
<th></th>
<th>Primary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best Case</strong></td>
<td>6.539</td>
<td>15.544</td>
<td>28.889</td>
<td>0.40</td>
</tr>
<tr>
<td><strong>Worst Case</strong></td>
<td>163.371</td>
<td>55.544</td>
<td>28.929</td>
<td>0.40</td>
</tr>
</tbody>
</table>

- **Mean Latency**
  - Secondary: 28.889
  - Primary: 28.929

**GREAT SUCCESS!**
### Automatic Task Offset/Priority Assignment

<table>
<thead>
<tr>
<th>TASK_NAME</th>
<th>CPU</th>
<th>PRIORITY</th>
<th>PERIOD</th>
<th>WCTIME</th>
<th>OFFSET</th>
<th>DEADLINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task100ms_E_1</td>
<td>F-1</td>
<td>15</td>
<td>100</td>
<td>0.013</td>
<td>26</td>
<td>100</td>
</tr>
<tr>
<td>Task100ms_M_1</td>
<td>F-1</td>
<td>2</td>
<td>100</td>
<td>0.025</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Task100ms_P_2</td>
<td>F-2</td>
<td>8</td>
<td>100</td>
<td>0.017</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>Task10ms_G_2</td>
<td>F-2</td>
<td>6</td>
<td>50</td>
<td>0.11</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Task10ms_Inp_A_1</td>
<td>F-1</td>
<td>2</td>
<td>10</td>
<td>0.115</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Task10ms_Inp_A_2</td>
<td>F-2</td>
<td>12</td>
<td>10</td>
<td>0.115</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Task10ms_B_1</td>
<td>F-1</td>
<td>1</td>
<td>10</td>
<td>0.258</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Task10ms_B_2</td>
<td>F-2</td>
<td>7</td>
<td>10</td>
<td>0.377</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Task50ms_G_2</td>
<td>F-2</td>
<td>11</td>
<td>50</td>
<td>0.11</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_F_1</td>
<td>F-2</td>
<td>0</td>
<td>50</td>
<td>0.167</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Task50ms_L_2</td>
<td>F-2</td>
<td>8</td>
<td>50</td>
<td>0.167</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_N_1</td>
<td>F-1</td>
<td>2</td>
<td>50</td>
<td>0.025</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_C_1</td>
<td>F-1</td>
<td>15</td>
<td>50</td>
<td>3.644</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_C_2</td>
<td>F-2</td>
<td>14</td>
<td>50</td>
<td>0.167</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_E_1</td>
<td>F-2</td>
<td>12</td>
<td>50</td>
<td>2.54</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Task50ms_N_2</td>
<td>F-2</td>
<td>11</td>
<td>50</td>
<td>0.148</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>TaskCAN_RX_1</td>
<td>F-1</td>
<td>14</td>
<td>10</td>
<td>0.01</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>TaskCAN_RX_2</td>
<td>F-2</td>
<td>7</td>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>TaskCAN_TX_1</td>
<td>F-1</td>
<td>9</td>
<td>10</td>
<td>0.02</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>TaskH_1</td>
<td>F-1</td>
<td>8</td>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>TaskH_2</td>
<td>F-2</td>
<td>10</td>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>TaskIPC_RX_1</td>
<td>F-1</td>
<td>13</td>
<td>10</td>
<td>0.01</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>TaskIPC_RX_2</td>
<td>F-2</td>
<td>0</td>
<td>10</td>
<td>0.01</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>TaskIPC_TX_1</td>
<td>F-1</td>
<td>6</td>
<td>10</td>
<td>0.01</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>TaskIPC_TX_2</td>
<td>F-2</td>
<td>4</td>
<td>10</td>
<td>0.01</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
Conclusions

- Automotive Architecture Design is an increasingly complex task & unmanageable with current practices
  - Early binding and late verification are no longer sufficient
  - *We need early exploration and late binding*
- Timing analysis/optimization methods and tools are one of the key components of this new design paradigm
Thank you for the attention!

paolo.giusto@gm.com
arkadeb.ghosal@gm.com
mckelvin@eecs.berkeley.edu
Back-up
Software Development Cost Model

Diagram:

- Hardware
- Design and Development
- ECS Cost
- Part Fabrication
- Integration - Assembly

Software

- Software size
- COCOMO model
- Project & Organization Factor

Software effort

- Determined in function points (number of inputs, outputs, inquiries, file structures etc)
- Platform (e.g. storage and processor capacity)
- Product (e.g. complexity and reliability)
- Personnel (e.g. capability and experience)
- Project (e.g. tools and time constraints)
- Communication overhead

Software man-months × labor cost
The design/redesign cost for a component is decided by the change or rework required; amount of rework required may be 100% implying that a component has to be designed from scratch. For standard off-the-shelf components, design cost is assumed negligible.

The complexity of functionalities, number of modules, software-hardware interfaces, new features, network topology.

Component Design/redesign

Packaging Design/redesign

Module Design

Module Validation

System Validation

Specification

Hardware

Software

Design and Development

ECS Cost

Part Fabrication

Integration - Assembly

Protocol type, number of interfaces, number of interrupt routines, software design methodology, concurrency, type of testing, number of IOs, number of sub-components, network controller.

External (Variable)

Internal (Variable)
Parts Fabrication Cost Model

Network controller
Timers
Power supply circuits
IO circuits
Interface controllers

Peripherals

Analog / Sensor
actuator

Processing
Unit

Secondary
Storage

Component

Connection

PCB

Housing

Packaging

Harness

BEC

Transceiver, Mode choke
Crystal, Protocol Type

type
packaging
accuracy
sensitivity

program size
data size

memory

processor

flash cost

boot sw
cal-set size
flash tech.

type
capacity

Component size

Wiring Size

PCB size
cost

X

Component

size

X

Housing

size
cost

X

Environment

factor

Peripherals

Hardware

Software

Design and
Development

ECS Cost

Tooling Investment

X

Production Volume

Part Fabrication

External

Interconnection

# IO pins

Fixed

Module

Piece

# cut-leads, # connectors
# specialized components

# relays, # fuses, # circuit breakers

Processing Unit

# cut-leads, # connectors
# specialized components

# IO pins

Component

Connection

PCB

Housing

Packaging

Harness

BEC

Environment

factor

Effort, speed, type

Memory

Processor

Flash cost

Type

Capacity

Type

Packaging

Accuracy

Sensitivity

Memory

Processor

Flash cost

Type

Capacity

Type

Packaging

Accuracy

Sensitivity

Effort, speed, type

Boot sw
Cal-set size
Flash tech.

Component size

Wiring Size

PCB size
cost

X

Component

size

X

Housing

size
cost

X

Environment

factor

Dotted lines represent the relationships between different components and their respective cost factors.
Assembly-Integration Cost Model

Hardware
Design and Development
Software

Part Fabrication

ECS Cost

Integration - Assembly

Part Maintenance
Harnessing
Place\ntment
EOL Verification
Flash

Cost of maintaining each new hardware module
+ Cost of maintaining each new calibration set

Flash time required by all the control modules
× flashing cost per unit time (often provided as a step function)