Architecture Analysis with AADL
The Speed Regulation Case-Study

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213

Julien Delange
What this talk is about?

1. Actual issues for Safety-Critical systems design

2. Why Model-Based Engineering techniques are helpful

3. How AADL can detect issues early and avoid potential rework
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Polling Question 1

Do you know what Model-Based Engineering is?
Safety-Critical Systems are Intensively Software-Reliant

Source: “Delivering Military Software Affordably” in Defense AT&L
Errors are introduced early but detected (too) lately

High Fault Leakage Drives Major Increase in Rework Cost

Aircraft industry has reached limits of affordability due to exponential growth in SW size and complexity.

70% Requirements & system interaction errors

80% late error discovery at high rework cost

70%, 3.5% \(1x\)

10%, 50.5% \(20x\)

Major cost savings through rework avoidance by early discovery and correction
A $10k architecture phase correction saves $3M

Rework and certification is 70% of SW cost, and SW is 70% of system cost.

Sources:
- D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson/Addison-Wesley (2014)

Where faults are introduced

Where faults are found

The estimated nominal cost for fault removal

Costly certification process leads to high percentage of operational work around.
Many Errors stems from Architecture or Integration

Fact 1: They incur rework costs and postpone product delivery

Root Cause:

Potential issues: inconsistent values

Use of COTS components with poor integration policy, lack of analysis

Fact 2: All these errors could be detected during integration tests

Root Cause:

Potential issues: inconsistent values

Global Variable used among different functions

Fact 3: They should I continue this list?
Why Model-Based Engineering Matters?

Capture system architecture with designers requirements
  Focus on system structure/organization (e.g. shared components)
  Tailor architecture to specific engineering domain (e.g. safety)

Validate the architecture
  Check requirements enforcement (e.g. no global variable)
  Detect Potential issues (e.g. interfaces consistency)

Early Analysis
  Avoid late re-engineering efforts (e.g. less rework after integration)
  Support decisions between different architecture variations
Polling Question 2

Do you already know AADL?
Architecture Analysis Design Language

SAE Standard for Model-Based Engineering
First version in 2003, actual version 2.1

Definition of System and Software Architecture
Specialized components with interfaces (not just “blocks”)
Interaction with the Execution Environment (processor, buses)

Extension mechanisms
User-Defined Properties (integrate your own constraints)
Annexes (existing for safety, behavior, etc.)
AADL Model Example
Architecture Analysis Design Language

Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

Security
- Intrusion
- Integrity
- Confidentiality

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Data Quality
- Data precision/accuracy
- Temporal correctness
- Confidence

Data Quality
- Execution time/Deadline
- Deadlock/starvation
- Latency

Real-time Performance
- Auto-generated analytical models

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Software Engineering Institute
Carnegie Mellon University

Speed Regulation Case-Study
Julien Delange
© 2014 Carnegie Mellon University
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Objectives of this Study

Learn Architecture Modelling with AADL and the OSATE workbench

Model a family of systems with their variability factors

Analyze the Architecture from a performance perspective

Discover Safety Issues using Architecture Models

Support Architecture Alternatives Selection

Illustrate the Process with a relevant case study
Case-Study Description

Self-Driving car speed regulation

Obstacle detection with user warning
  Camera detection
  Infra-red sensor

Automatic Speed and Brake
  Two speed (wheel, laser) sensors
  Redundant GPS
Polling Question 3

On what aspect would you like to focus?
Case-Study Objectives

Help designers to choose the best Architecture
   Best reliability, avoid potential failure/error
   Meet timing and performance requirements

Analyze Architecture according to stakeholders criteria
   Try to analyze what really matters

Quantify architecture quality from different perspectives
   Latency
   Resources and Budgets
   Safety/Reliability
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Functional Architecture, timing perspective

Max end-to-end latency = 900 ms
Functional Architecture, criticality perspective

Redundancy Groups (performs the same function)
Deployment Alternatives

Alternative 1: reduce cost and complexity
   Two processors and one shared bus
   Potential interactions for functions collocated on the same processor

Alternative 2: reduce potential fault impact
   Increase potential production cost (more hardware)
   Three processors inter-connected with two buses
Architecture Alternative 1

- Obstacle Camera
- Obstacle Radar
- Speed Wheel Sensor
- Speed Laser Sensor
- GPS1
- GPS2
- Obstacle Image Acquisition
- Obstacle Distance Evaluation
- Obstacle Detection
- Time to Obstacle Evaluation
- Emergency Detection
- Warning Activation
- Warning Device
- Speed Estimate
- Speed Threshold Computation
- Speed Controller
- Brake
- Acceleration
- Position Voter

Reduce Cost and Complexity
Potential interactions for functions collocated on the same processor

Bandwidth: 500 kbps
Acquisition time: 10 to 30ms
Transmission time: 1 to 10 us per byte

50 MIPS
ECU1

50 MIPS
ECU2

Software Engineering Institute
Carnegie Mellon University

Speed Regulation Case-Study
Julien Delange
© 2014 Carnegie Mellon University
Architecture Alternative 2

- Obstacle Camera
- Obstacle Radar
- Speed Wheel Sensor
- Speed Laser Sensor
- GPS1
- GPS2
- Obstacle Image Acquisition
- Obstacle Distance Evaluation
- Speed Estimate
- Obstacle Detection
- Time to Obstacle Evaluation
- Emergency Detection
- Warning Activation
- Warning Device

Reduce Fault Impact
Might increase production costs

Bandwidth: 5 kBps
Acquisition time: 50 to 100ms
Transmission time: 10 to 50 us per byte

ECU1: 50 MIPS
ECU2: 50 MIPS
ECU3: 50 MIPS

Brake
Acceleration
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Modeling Guidelines

Separate architecture aspects in different files

Leverage AADL extension and refinement mechanisms
  Capture common characteristics, avoid copy/paste
  Extend generic components

Use properties to quantify quality attributes
  Processed by tools to evaluate architecture quality
  Specify once, use by several analysis tools
  Ensure Analyses Consistency
Model Organization – devices

Generic components

Extension and refinements
Model Organization – devices – textual model

Component Name

Timing constraints
(latency analysis)

Error propagations and flows

Types of faults
(all safety analysis tools)

Documenting the faults
(safety analysis)
Model Organization – Interfaces Specifications

Data types being used to communicate across functions

One property, several analyses ⇒ Ensure Analyses Consistency

Data size properties (resource allocation and latency analysis)
Model Organization – platform

Generic Processor Component (common for all the architecture)

Processor extension, specify bus connections
Share properties of inherited component

Timing information (latency analysis)
Model Organization – software (1)

One software function = 1 AADL process + 1 AADL thread
Model Organization – software – textual notation (1)

Component type

Communication interfaces

Data flow specification
(latency analysis)

Error specification
(safety analyses)

Subcomponents and connections

Component implementation

---

Software Engineering Institute
Carnegie Mellon University

Speed Regulation Case-Study
Julien Delange
© 2014 Carnegie Mellon University

34
Model Organization – software – textual notation (2)

Data flow (latency analysis)

Resource Budgets (resource allocation analysis)

Time information (latency analysis)
Model Organization – safety specification

Error types that could be raised

Error states

Component-specific error transitions
(to be added on a component-basis)

Reusable error state machines
to be attached to components
Model Organization – define error flows – error source

Component camera

Reuse predefined types

Define error types propagated on component interfaces

Define the error sources, what interfaces initiates an error flow

device camera
features
    picture : out data port speed_regulation::icd::picture;
flows
    f0 : flow source picture;
properties
    Period => 200ms;
annex EMV2 {**
    use types speed_regulation::error library;
    error propagations
        picture : out propagation {NoValue};
    flows
        ef0 : error source picture{NoValue};
    end propagations;
**};
end camera;
Model Organization – define error flows – error path

Reuse predefined types and behavior

Define error types propagated on component interfaces

Define the propagations flows

obstacle_distance / NoValue  obstacle_detected / NoValue
obstacle_distance / InvalidValue  obstacle_detected / InvalidValue
Processor / SoftwareError  obstacle_detected / InvalidValue
Processor / HardwareError
Model Organization – error sink & define component error behavior

Use predefined error types and component behavior

Define component-specific error events

Component-specific error transitions

Flowchart:
- Operational
- Failed
- NoValue
- InvalidValue
- Reset

Code snippet:
```plaintext
device warning_device
def features
  warning : in data port speed_regulation::icd::boolean;
def flows
  f0 : flow sink warning;
def properties
    Period => 500ms;
  annex ENV2 {**
    use types speed_regulation::error_library;
    use behavior speed_regulation::error_library::simple;
  }
  error propagations
    warning : in propagation {NoValue,InvalidValue};
  flows
    ef0 : error sink warning{NoValue,InvalidValue};
  end propagations;
  component_error_behavior
    events
      Reset : recover event;
    transitions
      t0 : Operational -[warning{NoValue}] -> Failed;
      t1 : Operational -[warning{InvalidValue}] -> Failed;
      t2 : Failed -[Reset] -> Operational;
  end component;
}
end warning_device;
```
Model Organization – architecture alternatives

Capture common components characteristics

Common type for all architecture alternative

System implementation with all common components

Capture architecture alternatives variability (processors, buses, etc.)
Architecture Alternative 1: model instance
Architecture Alternative 2: model instance

Variability Factors with Alternative 1
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Latency Analysis, principles

Potential impact on latency

<table>
<thead>
<tr>
<th>Bus characteristics</th>
<th>Alternative1</th>
<th>Alternative2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisition Time</td>
<td>10 to 30 ms</td>
<td>200 to 500 ms</td>
</tr>
<tr>
<td>Transmission Time (/B)</td>
<td>1 to 10us</td>
<td>2 to 5 ms</td>
</tr>
</tbody>
</table>
Latency Analysis, results

<table>
<thead>
<tr>
<th>flow</th>
<th>model element</th>
<th>name</th>
<th>deadline or conn delay</th>
<th>total</th>
<th>expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>f0</td>
<td>End to End Latency report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>device</td>
<td>obstacle_camera:f0</td>
<td>200.0 ms</td>
<td>200.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>obstacle_camera:picture</td>
<td>0.0 us</td>
<td>200.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>thread</td>
<td>image_acquisition:thr:0</td>
<td>50.0 ms</td>
<td>250.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>image_acquisition:thr:0</td>
<td>0.0 us</td>
<td>250.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>thread</td>
<td>obstacle_detection:thr:100.0 ms</td>
<td>100.0 ms</td>
<td>350.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>obstacle_detection:thr:30.00125 ms</td>
<td>300.00125 ms</td>
<td>390.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>thread</td>
<td>obstacle_distance:eval:10.0 ms</td>
<td>10.0 ms</td>
<td>390.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>obstacle_distance:eval:0.0 us</td>
<td>0.0 us</td>
<td>390.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>thread</td>
<td>emergency_detection:thr:14.0 ms</td>
<td>14.0 ms</td>
<td>394.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>emergency_detection:thr:14.0 ms</td>
<td>14.0 ms</td>
<td>394.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>thread</td>
<td>warning_activation:thr:2.0 ms</td>
<td>2.0 ms</td>
<td>396.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Connection</td>
<td>warning_activation:thr:0.0 us</td>
<td>0.0 us</td>
<td>396.00125 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>device</td>
<td>warning_alert:f0</td>
<td>500.0 ms</td>
<td>500.0 ms</td>
<td>900.0 ms</td>
</tr>
<tr>
<td>f0 (Synchronous)</td>
<td>Total</td>
<td></td>
<td>0.0 us</td>
<td>500.0 ms</td>
<td>900.0 ms</td>
</tr>
</tbody>
</table>

**Architecture Alternative 1**

- f0: End-to-end flow f0 calculated latency (Synchronous) 896.00125 ms is less than expected latency 900.0 ms

**Architecture Alternative 2**

- ERROR: f0: End-to-end flow f0 calculated latency (Synchronous) 956.00625 ms exceeds expected latency 900.0 ms
Resources Allocation Analysis, principles
Resources Allocation Analysis, results

Architecture Alternative 1

Architecture Alternative 2
Safety Analyses Overview

Functional Hazard Analysis (FHA)
- Failures inventory with description, classification, etc.

Fault-Tree Analysis (FTA)
- Dependencies between errors event and failure modes

Fault-Impact Analysis
- Error propagations from an error source to impacted component

Need to combine analyses
- Connect results to see impact on critical components
Safety Analysis, FHA, results

Architecture Alternative 1: 15 errors contributors ×

Architecture Alternative 2: 17 errors contributors ✓

Difference stems from additional platform components (ecu)
Have to consider criticality of fault impacts
Safety Analysis, FTA results

Architecture Alternative 1: 15 errors contributors ✗

Architecture Alternative 2: 17 errors contributors ✓

Difference stems from additional platform components (ecu)

Have to consider criticality of fault impacts
Safety Analysis, Fault Impact, results

Architecture Alternative 1 & 2: 443 error paths

Use the same paths

  The additional ECU in alternative 2 covers path from ecu2 in Alternative 1

Impact on components criticality

  Defect on the additional bus in Architecture 2 impact low-critical functions

  Isolate defect from low-critical functions to affect high-critical
### Analysis Summary

<table>
<thead>
<tr>
<th></th>
<th>Architecture 1</th>
<th>Architecture 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Resources Budgets</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Safety</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Cost</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>

**What is the “best” architecture?**
Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion
Conclusions

Safety-Critical Systems Development issues is not a fatality
  Late detection of errors is no longer possible
  Need for new methods and tools

AADL supports Architecture Study and Reasoning
  Evaluate quality among several architectures
  Ease decision making between different architecture variations
  Analysis of Architectural change on the whole system

User-friendly and open-source workbench
  Graphical Notation
  Interface with other Open-Source Tools
Useful Resources

AADL wiki – http://www.aadl.info/wiki

*Model-Based Engineering with AADL* book

SEI blog post series http://blog.sei.cmu.edu

Mailing-List

see. https://wiki.sei.cmu.edu/aadl/index.php/Mailing_List