

AADL Webinar

Table of Contents

Carnegie Mellon University – Notices.....	4
Architecture Analysis with AADL The Speed Regulation Case-Study	4
What this talk is about?	7
Agenda	8
Agenda	9
Polling Question 1	10
Safety-Critical Systems are Intensively Software-Reliant	11
Errors are introduced early but detected (too) lately	12
Many Errors stems from Architecture or Integration Issues	13
Why Model-Based Engineering Matters?.....	15
Architecture Analysis Design Language	16
AADL Model Example.....	17
Architecture Analysis Design Language	20
Agenda	21
Objectives of this Study	21
Case-Study Description	22
Case-Study Objectives.....	23
Agenda	24
Functional Architecture	25
Functional Architecture, timing perspective	26
Functional Architecture, criticality perspective.....	27

Deployment Alternatives	28
Architecture Alternative 1	29
Reduce Fault Impact Might increase production costs	30
Agenda	31
Modeling Guidelines.....	34
Model Organization – devices	35
Model Organization – devices – textual model	36
Model Organization – Interfaces Specifications	37
Model Organization – platform	38
Model Organization – software (1).....	39
Model Organization – software – textual notation (1).....	40
Model Organization – software – textual notation (2).....	41
Model Organization – safety specification	41
Model Organization – define error flows – error source.....	43
Model Organization – define error flows – error path.....	44
Model Organization – error sink & define component error behavior	45
Model Organization – architecture alternatives	46
Architecture Alternative 1: model instance.....	48
Architecture Alternative 2: model instance.....	49
Agenda	50
Latency Analysis, principles	53
Latency Analysis, results	55
Resources Allocation Analysis, principles	56
Resources Allocation Analysis, results	57

Safety Analyses Overview	59
Safety Analysis, FHA, results	62
Safety Analysis, FTA results.....	65
Safety Analysis, Fault Impact, results	66
Analysis Summary	67
Conclusions	68
Useful Resources.....	70
Questions & Contact.....	71

Carnegie Mellon University

This video and all related information and materials (“materials”) are owned by Carnegie Mellon University. These materials are provided on an “as-is” “as available” basis without any warranties and solely for your personal viewing and use.

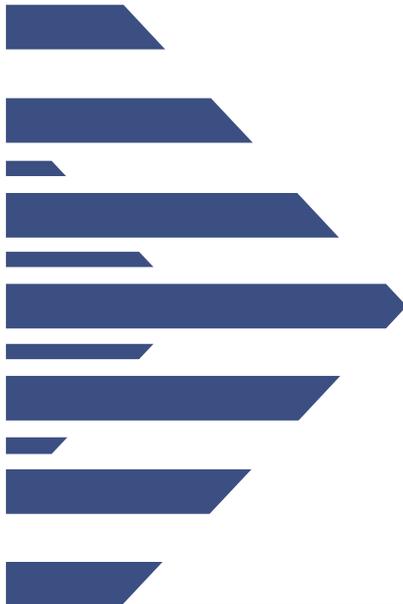
You agree that Carnegie Mellon is not liable with respect to any materials received by you as a result of viewing the video, or using referenced websites, and/or for any consequences or the use by you of such materials.

By viewing, downloading, and/or using this video and related materials, you agree that you have read and agree to our terms of use (www.sei.cmu.edu/legal/).

© 2014 Carnegie Mellon University.



Architecture Analysis with AADL The Speed Regulation Case-Study



Architecture Analysis with AADL The Speed Regulation Case- Study

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Julien Delange



**001 Shane McGraw: Hello, and welcome to the Software Engineering Institute's Webinar Series.

Our presentation today is
Architecture Analysis with AADL.

Depending on your location, we wish
you a good morning, a good
afternoon, a good evening.

My name is Shane McGraw. I'll be
your moderator for the presentation;
and I'd like to thank you for attending.

We want to make today as interactive
as possible. So we will address
questions throughout the
presentation and again at the end of
the presentation. You can submit
questions at any time to our event
staff through the control panel.
You'll see a Questions tab. Simply
type in your question and click Send.
We will also ask a few polling
questions throughout the
presentation. These will appear as a
popup window on your screen.

Another three tabs I'd like to point
out are the Twitter, Survey and
Materials tabs.

The Materials tab has a PDF copy of
the presentation slides now that you
can take today, along with training
information from the SEI on AADL.

For those of you using Twitter, be
sure to use- to follow @saturn_news;
and use the hash tag AADL. Once
again, it's @saturn_news with a hash
tag of SEI AADL.

And lastly the Survey will appear at
the end of the presentation. We

request your feedback as it's always greatly appreciated.

Now I'd like to introduce our presenter, Dr. Julien Delange. He is a member of the Technical Staff at the SEI where his research interests are model-based engineering and improving the development of safety critical systems by early discovery of architecture and design issues.

Before joining the SEI, he worked as a software engineer at the European Space Agency where he led and contributed to several research projects related to software and system architecture for safety critical systems.

Dr. Delange got involved with architecture design and analysis while designing new methods and techniques to improve the safety and critical- safety and security of critical systems.

And now I'd like to turn it over to Julien. Julien, all yours.

Julien Delange: Thank you. Thank you so much for attending this webinar.

What this talk is about?

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



**003 So first of all I would like to point out what this talk is about. And beforehand I would like to address the first polling question to know if you are familiar or not with model-based engineering. Shane?

Shane McGraw: Okay so you can see the polling question on your question now. And we'll give you about 15 or 20 seconds to vote. So Julien, just keep on going.

Julien Delange: Okay. So this talk is about- discusses actual issues for safety-critical system design and why model-based engineering can help you to design safety-critical systems. And after that we'll focus on how AADL technology can detect issues early in the development process.

Agenda

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion



**004 So first of all we'll introduce what is model-based engineering; and also what are the problems actually when you design your safety-critical systems.

We'll present a case study; and we will apply AADL to verify and analyze safety-critical systems. We will present this system, present the AADL model of the system and see how we can analyze the architecture.

So first of all Shane, what are the answers of the polling question?

Shane McGraw: So let me launch the results right now. Take us a moment.

Agenda

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion



Software Engineering Institute

Carnegie Mellon University

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

5

**005 And we got 35 percent that they already work on projects related to model-based engineering; four percent that are real experts; 34 percent have read papers but never investigated; and 27 percent that do not know about model-based engineering.

Julien Delange: All right.

Shane McGraw: Quite a mix.

Julien Delange: All right, so it's quite a mix.

~~~

## Polling Question 1

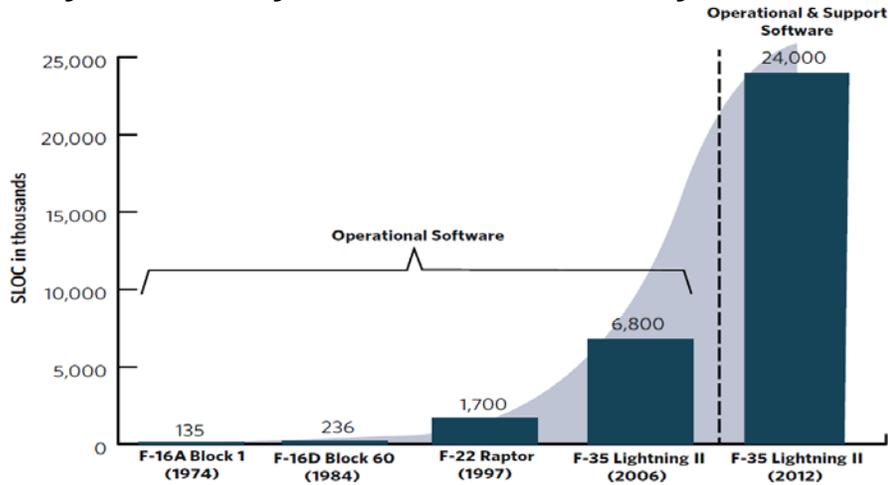
# Polling Question 1

Do you know what Model-Based Engineering is?

\*\*006 So let me introduce what  
model-based engineering is about.

## Safety-Critical Systems are Intensively Software-Reliant

### Safety-Critical Systems are Intensively Software-Reliant



Source: "Delivering Military Software Affordably" in Defense AT&L



Software Engineering Institute | Carnegie Mellon University

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

7

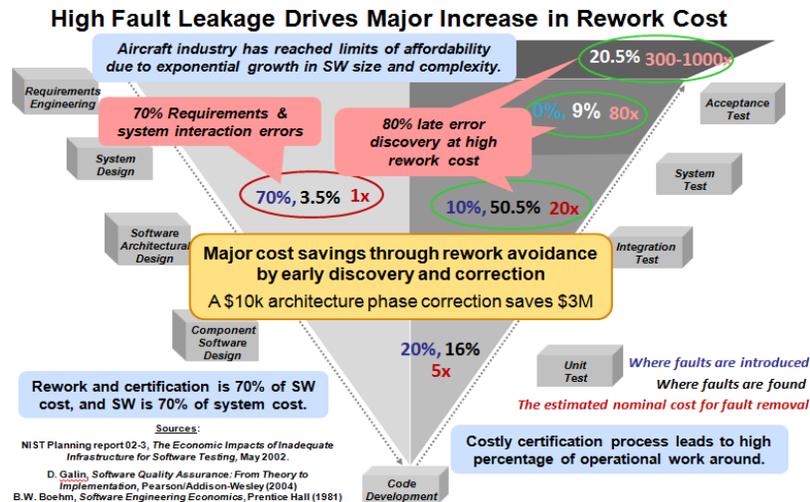
\*\*007 But first of all where we have issues presented in cyber-physical systems.

If you look at the size of the code of avionics architectures, you see that between '74 and 2006 the size-- so in 32 years - the size of the software increased by 5000 percent.

The problem right now is you have more and more components; you have more components in a plane. And all these components are at different criticality levels. So it was a problem when you integrate all these components.

## Errors are introduced early but detected (too) lately

### Errors are introduced early but detected (too) lately



Software Engineering Institute

Carnegie Mellon University

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

8

\*\*008 And this is exactly the problem we find right now. In fact during the system design 70 percent of errors are introduced during the design. But we find only three to five percent of these errors. So we introduce a lot of errors and we don't discover these errors during the design.

On the other hand during integration, during the tests, we introduce 10 percent of the errors; but we find 50 percent of the errors. In other words, we find the errors- the errors really lately in the development process; and we introduce these errors really early.

So we need to address this issue.

## Many Errors stems from Architecture or Integration Issues

### Many Errors stems from Architecture or Integration Issues

Global Variable used among different functions

Potential issues: inconsistent values

Root Cause: Architecture Design

Use of COTS components with

Potential impact

Root Cause

Timing issues

**Fact1: All these errors could be detected at Design-Time**

**Fact2: They are actually detected during integration tests**

**Fact3: They incur rework costs and postpone product delivery**

Should I continue this list?



Software Engineering Institute

Carnegie Mellon University

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

9

\*\*009 And also where-- what kind of error we find-- during the development process? What kind of error are we finding during the integration?

First of all, many global variables and we have many different interactions.

And for example, we have inconsistent values, concurrent accesses and what you call race-condition.

Also we reuse components; and sometimes the reuse quality is not so good and we have some inconsistency when we integrate them

We also have timing issues; for example, the deadlines are not enforced, the worst-case execution time sometimes is wrong and so on.

So we have a lot of issues we discover lately. And I think I should not continue the list because I can have a list that is really, really long.

Fact number one is that all these errors can be detected at design time.

And fact number two: Actually we detect these errors during integration.

And fact number three: They incur a lot of rework costs and engineering work.

So we need to address these issues early in the development process; not during integration.

So we have different technologies; and a model-based technology will help you to discover these errors early. And we have the AADL technology to discover these errors early during the design of the system.

And that's why I would like to ask you if you already know AADL. I would like to know if you are familiar with AADL. Shane?

Shane McGraw: Okay so you'll see a polling question popping up on your screen now, asking if you already know about AADL.

## Why Model-Based Engineering Matters?

# 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



\*\*010 And that's going to help  
Julien-- kind of where we go with the  
presentation.

Julien Delange: Thank you so  
much. So why model-based  
engineering is really important?  
Because with a model-based  
approach you can capture the system  
elements with a high-level language;  
and with that you just focus on what  
really matters on your system: time,  
safety, fault errors and stuff like this.

After that, you use this model to  
validate the architecture. So you  
really focus on what is really  
important. You check that your  
requirements are enforced. For  
example, your end-to-end latency,  
the different number of errors. You  
try to detect potential issues from the  
beginning; and you don't address this  
error later in the development process.

# 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.)



\*\*012 And for that  
have a language called AADL: The  
Architecture Analysis and Design  
Language.

Are you familiar with that? Shane,  
what are the answers?

Shane McGraw: So we got 14  
percent are familiar with it; 45  
percent-- let me-- show end results,  
yes-- 45 percent that are- know the  
principles; and then 41 percent have  
not heard of it.

Julien Delange: Okay so let me  
introduce what is AADL. So the first  
version of the language was in 2003;  
and actually it is version 2.1.

The goal is to define the software  
architecture and the system  
architecture; and not only with blocks  
but with specialized components.

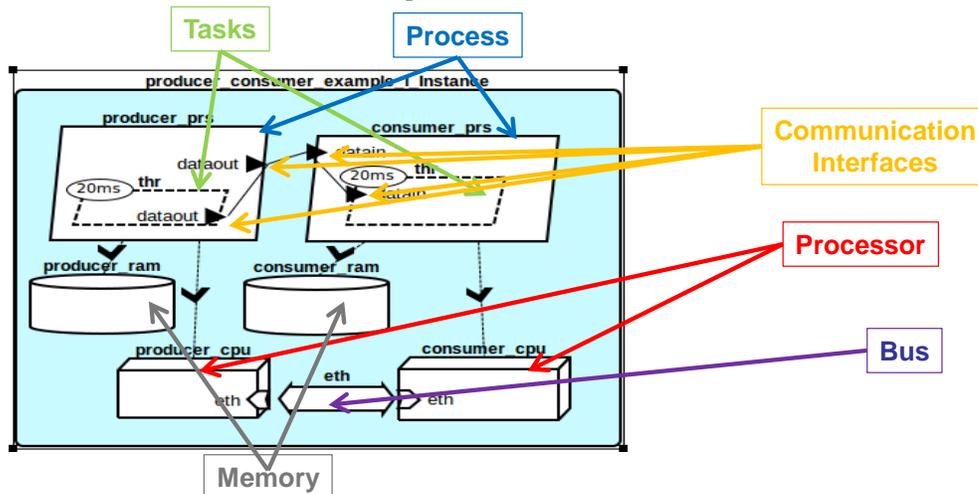
So we define what is the software, what are the tasks, the time requirement and so on. But we also have to shape that to the execution platform: the processor, buses and so on.

And you can also extend the language with different extension mechanisms; what you call the user-defined properties, user can define their own properties, or an annex mechanism.

For example for safety I will show you later on in this presentation how we extend AADL with safety information.

### AADL Model Example

## AADL Model Example



\*\*013 So we have a model; an AADL model can have a graphical notation, as I show, with different component types: tasks,

communication interfaces, process and so on.

What is really important is that

We have a real tool set that supports the AADL language and its notation.

So let me show you what we have today. We have a full tool set that is called OSATE; and it's an Eclipse platform. And you can create an AADL model directly and edit it. So it's a language; but we also have a tool environment.

So let's try to show you how I can create an AADL model. And this is really simple. I create a new AADL package and I will call it Example. All right? And then I can create different component types; as you see on the right of the screen I have different component types. And I will just make a real simple system with a producer and a consumer.

So let me add a process type. Okay? And the process will send the data. So for that I need first of all to make a process that I will call, for example, Sender or Producer; and I will make a new process, another one-- that will be the receiver.

So I have two different processes; and what I need after that is to define the communication interfaces. So I will add what you call a data port; and the goal will be to send the data through the interface. And I

can call it Data Out, for the data I'm sending. All right?

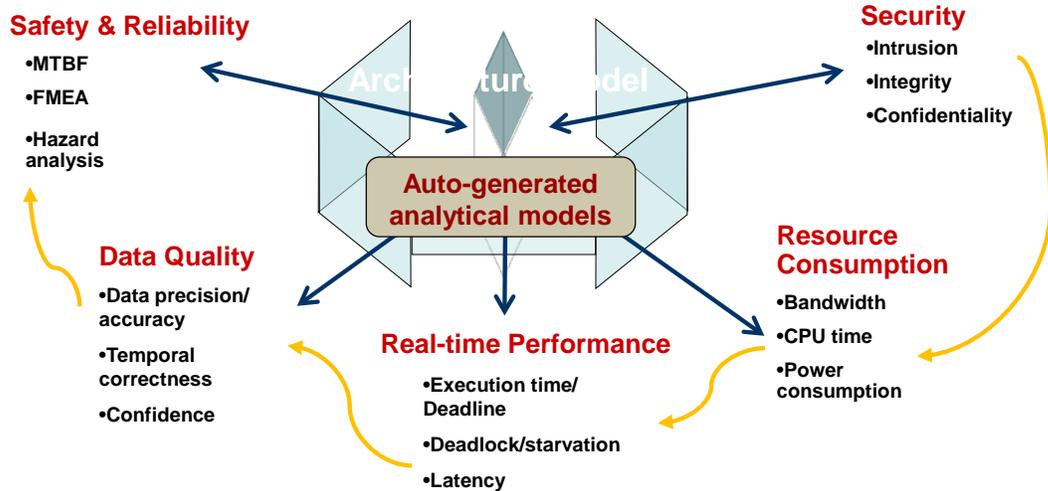
And the receiver, I will have to also define a communication interface here, the out port, and I will call it-- the in port, sorry-- and I will call it Data In; and then it's an in port. All right?

So these two processes are component types. And I can integrate them in what I call a system. So I create a system; and then I will create an implementation. And my system implementation can contain my two components, Producer and Receiver.

So if I go inside my system, I can add the first process; that will be my Sender and the second process that will be my Receiver. So I say this is my producer; this one will be my receiver. And I can connect the interfaces; like this.

So it's really simple. In just two minutes I create a model, the Sender and the Receiver. And after that I can use these models to start to validate my architecture. Okay?

# Architecture Analysis Design Language



\*\*014 So in AADL we have different plugins in these toolsets to validate different aspects of your systems,

Security. If you are-- for example, the producer and the consumer share a different security level; the resource consumption, bandwidth, CPU time. If you associate different tasks to a processor, you will check that you have enough processing time; also the data quality and safety and reliability.

## Agenda

# Agenda

Introduction on Model-Based Engineering

## Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion

\*\*015 All right.

## Objectives of this Study

# 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

\*\*016 So today we will use this

language, AADL, to validate a system. So we'll try to learn what is AADL and the different concepts and model a system with different variations.

## Case-Study Description

# 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



\*\*017 And we'll discover potential issues we can find out in the system.

So the case study is really simple; it's a self-driving car. And in fact you have a different device; and it's an actuator. So we can detect obstacles; and we can also put an alarm for the driver. And also we regulate automatically the speed of the car.

## 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**



\*\*019 So let me also ask you what would you like to investigate in terms of analysis?

For this case study we have different analysis tools we can use: the latency, the resource/budget and the safety/reliability. That's why we're asking you what you'd like to investigate.

So the objective today is to help designers to choose what is the best architecture for their system; and for the self-driving car, we propose a different architecture viability, different architecture candidates.

And then we'll use our tool to analyze the architecture so that you can choose what's the best architecture. And without the implementation we'll be able to have some measure and analyze the architecture to decide

what is the best, according to your requirements.

## Agenda

# Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

## System Overview

AADL model description

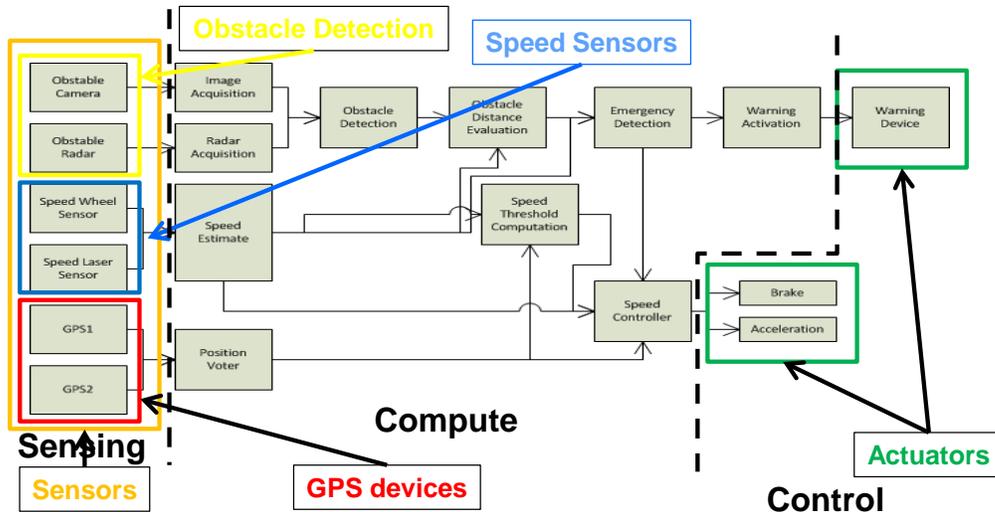
Architecture Analysis

Conclusion

\*\*020 So let me introduce what is the system.

## Functional Architecture

### Functional Architecture



\*\*021 So this is a functional architecture. On the left side you have all the sensors; the camera that will detect an obstacle, radar, speed sensor- two speed sensors and two GPS.

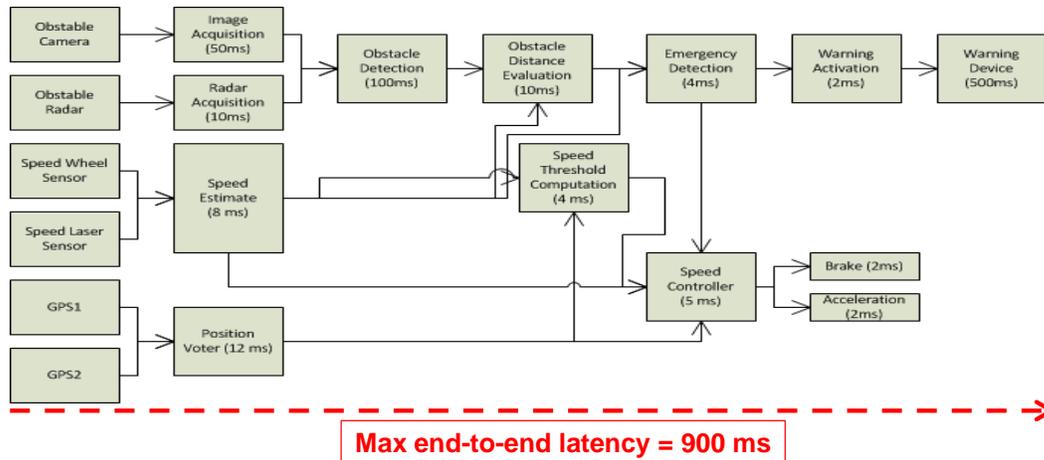
~~~

So the architecture is divided in three parts: sensing, with the sensor; compute, we process the value of the sensors and we control the brakes, the acceleration and the warning device for the driver.

So what is important is we have redundancy. The two GPS are redundant. Also if a speed sensor fails we can use the other one.

Functional Architecture, timing perspective

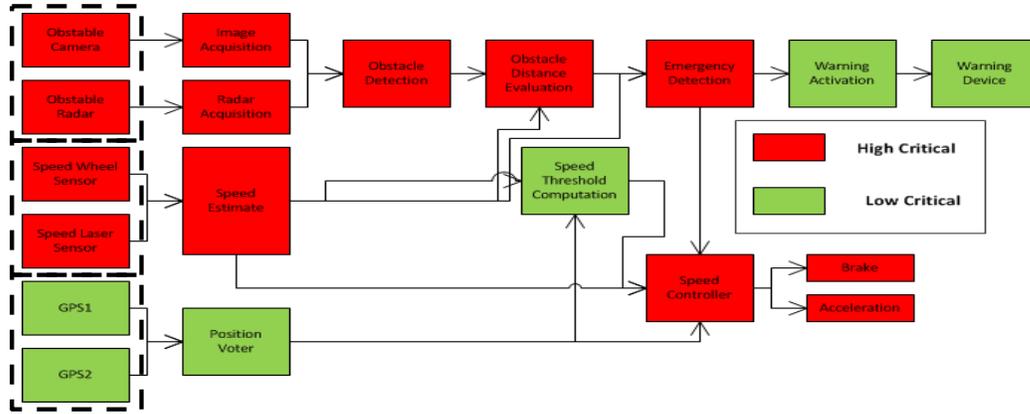
Functional Architecture, timing perspective



**022 The same for the obstacle sensor.

And from a timing perspective we are expecting that the max end-to-end latency, when we produce data to the end, is at max 900 milliseconds. In other words, when data is produced by the obstacle camera or the obstacle radar, I want to make sure that the data is consumed by the warning device before 900 milliseconds.

Functional Architecture, criticality perspective



Redundancy Groups (performs the same function)

**023 From a criticality perspective, let me show you that the GPS is not so important. But the obstacle camera and radar are really important and really critical.

That's why you ask if I have a different criticality level. When I control the car, the GPS it's: Where is the car, where is my position? It's not so critical. But the data about obstacles, and also the speed of the car, really matters and is really highly critical.

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

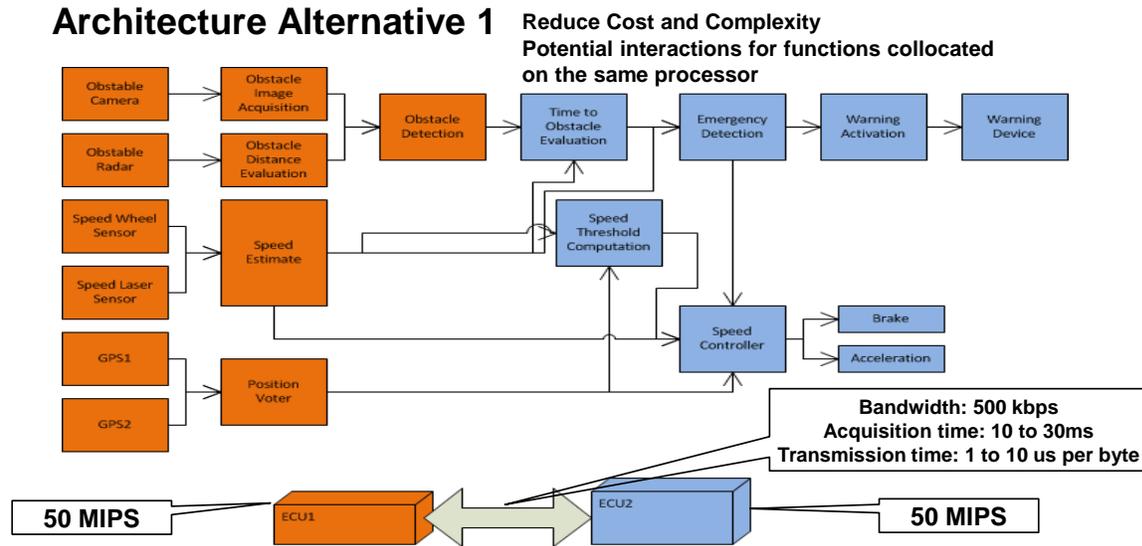


**024 For that we have two different alternatives.

So the first alternative for that data for the system is to reduce the cost and the complexity. I will only use two processors connected on a shared bus. But the problem is in that case I have to co-locate critical and non-critical functions on the same processor. So a non-critical function can impact a critical function.

So alternative number two is to reduce the potential fault impact. So we'll try to co-locate noncritical functions on a single processor and critical functions on other processors.

Architecture Alternative 1



**025 So that a non-critical function cannot impact a high critical function.

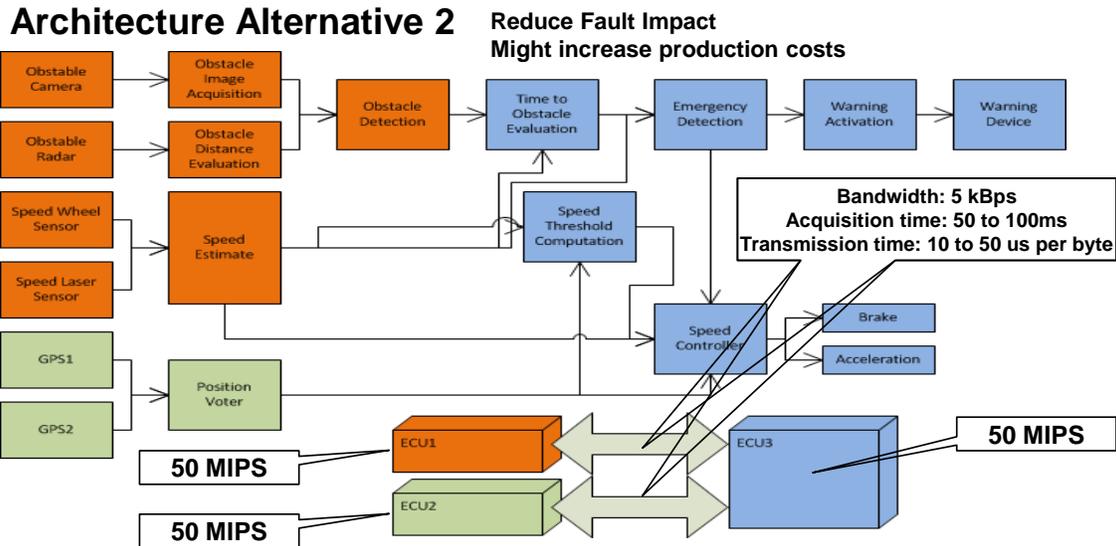
So this is architecture alternative-- sorry-- number one. So in this architecture alternative I reduce the cost and the complexity. So I see that all the sensing functions are collocated on one processor and all the other actuating function and processing function are on another processor.

So I reduce the cost. I don't need three processors, only two; and just one bus.

And I also shared a budget of 50 MIPS on each processor. This is my computing capacity.

And then for my bus I have a bandwidth budget.

Reduce Fault Impact Might increase production costs



**026 For architecture number two I have three processors and two buses. So I reduced the fault impact of non-critical functions; but also I increased the production costs because I have more processors. So it will be more costly to deploy.

And each processor, as the other one, 50 MIPS; and I have a budget for each of those.

Agenda

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion



**027 So let me-- let's have a look at the results.

Shane McGraw: Yes. And could we just pause for a few questions too after we show the results?

Julien Delange: Sure. Do we have any questions?

Shane McGraw: We do. So while I cue up the results.

First question was from Don asking: Does AADL utilize state machines with arcs and nodes along with frequency of innovation and interfacing data types?

Julien Delange: Okay. So in fact we have an annex. I pointed out that early that we have annex extension; and we can associate state machine in the components to

associate the behavior of the system with the state machine.

So we have the capability in the tool. So right now after that we can already define that and make use of this state machine to analyze the system.

Shane McGraw: Okay. And then from Mark asking about the demo: Does the SEI license this tool? And the real important question: Is it free?

Julien Delange: All right. So we- so all what I'm showing right now is available online. We have a model currently on GitHub; github.com. And if you go online at AADL.info/wiki, you have all the description of the model, how to get it. The tool is available under the Eclipse public license and is available for free online.

Also all the demo and video today I have done with OSATE version 2.0.8. So you can reproduce all the demo on your computer.

Shane McGraw: Okay; and just one more quickly from Vishal asking: What are the connectors in a diagram?

Julien Delange: Okay so the connectors-- so we have a different interface in AADL, something that we call Data Ports; even data ports and even ports.

So the difference is data ports is only like sensing ports; will just exchange data. Even data ports will exchange data but also events. So you have the notion, the concept, of an event; it's new data or not. And even ports is only an event without any data.

So we have different interfaces. And in this example I used data ports; like sampling ports. The reason why I'm using that is because it's what is currently used in many safety-critical systems; we just sample the data. All right?

Shane McGraw: Okay. And just to close out the survey: 27 percent requesting resource/budgets allocation; 10 percent on latency; and the majority, 63 percent, at fault/error propagation.

Julien Delange: All right. So we'll make a pass quick on the latency; also I will show how to use the resource analysis; and then focus on the safety tools.

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



**028 All right?

Shane McGraw: Great.

Julien Delange: All right. So some modeling guidelines we are using; because we have different architecture variability. So what we try to do is to factor the model so all the common aspects of the architecture are not duplicated.

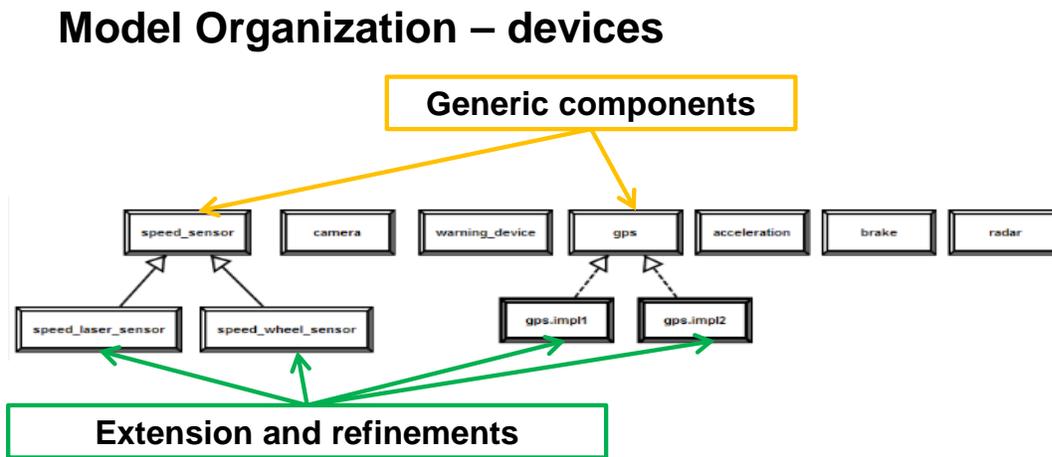
So we have a separate architecture aspects in different files. But what you try to do is to leverage the AADL extension mechanism, like any of them, and try only to capture the different variability factor of each component.

In other words, if for the two different variations what is really important is the number of processors or the number of buses, I can make a generic architecture,

extend it and just add two or three processors.

And we'll use properties, AADL properties, to quantify the quality attributes of the system. The MIPS capacity; so the capacity of the processor and the software; also the different error types that are propagated in the architecture.

Model Organization – devices



**029 And so on.

So let me show you quickly what it looks like. So for all device- for each device-- sorry-- I have a component type. And if you look at it, I've given two different speed sensors. And I take advantage of the extension, like any of them, of AADL, to have a generic speed sensor; and then refine it with a speed laser sensor and a speed wheel sensor.

So I have a common speed sensor that sends data basically. The interface is exactly the same. But after that I will have a different implementation.

I do that with the GPS as well. I have a common GPS and I have two different implementations to distinguish the two different GPS vendor or GPS implementation.

Model Organization – devices – textual model

Model Organization – devices – textual model

```

device radar
features
  distance_estimate : out data port speed_regulation::icd::distance;
flows
  f0 : flow source distance_estimate;
properties
  f0 : flow source distance_estimate;
  Period => 10ms;
annex ENV2 ("")
  use types speed_regulation;
error propagations
  distance_estimate : out propagation {NoValue,InvalidValue};
flows
  ef0 : error source distance_estimate{NoValue,InvalidValue};
end propagations;
properties
  emv2::severity => ARP4761::Major applies to distance_estimate.novalue;
  emv2::likelihood => ARP4761::Probable applies to distance_estimate.novalue;
  emv2::hazards =>
  ([ crossreference => "N/A";
    failure => "NoValue";
    phases => ("all");
    description => "No information from the Radar";
    comment => "Error if both the camera and the radar does not send any value";
  ])
  applies to distance_estimate.novalue;
  emv2::severity => ARP4761::Minor applies to distance_estimate.invalidvalue;
  emv2::likelihood => ARP4761::Probable applies to distance_estimate.invalidvalue;
  emv2::hazards =>
  ([ crossreference => "N/A";
    failure => "InvalidValue";
    phases => ("all");
    description => "Invalid distance sent by the radar";
    comment => "First occurrences of invalid data Should be handled by the distance estimator.";
  ])
  applies to distance_estimate.invalidvalue;
end radar
  
```

**030 So I said before we had a graphical notation of AADL but also a textual notation. This is what the textual notation looks like. I will not show all the details. And please have a look on the wiki description if you want to have a look. Also we have a video where we make a walk through- through the model later.

Model Organization – Interfaces Specifications

Model Organization – Interfaces Specifications

```
data gps_position
properties
  data_size => 50 Bytes;
  data_model::data_representation => Array;
end gps_position;

data speed_command_type
properties
  data_model::data_representation => enum;
  data_model::enumerators => ("brake", "accel");
  size => 2 bits;
  command_type;
end speed_command_type;

data implementation speed_command.i
subcomponents
  kind : data speed_command_type;
  value : data base_types::unsigned_16;
end speed_command.i;

data distance extends base_types::unsigned_32
end distance;
```

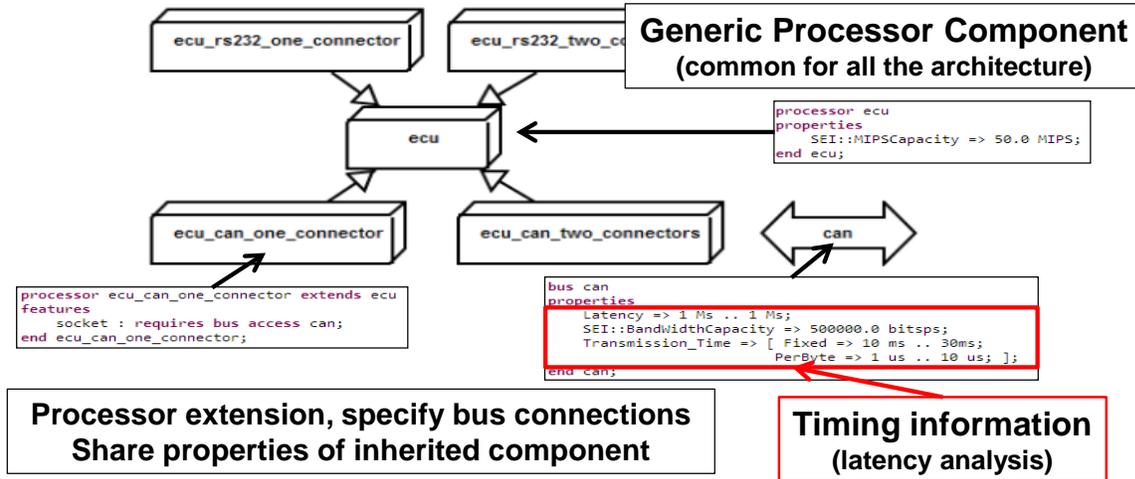


**031 The interface specification.

We define the data size for each data type. So the GPS, for example, we say that the size of the data is 50 bytes; Boolean is 1 bit; and the common type to accelerate or brake is 2 bits, but it is just a flag.

Model Organization – platform

Model Organization – platform

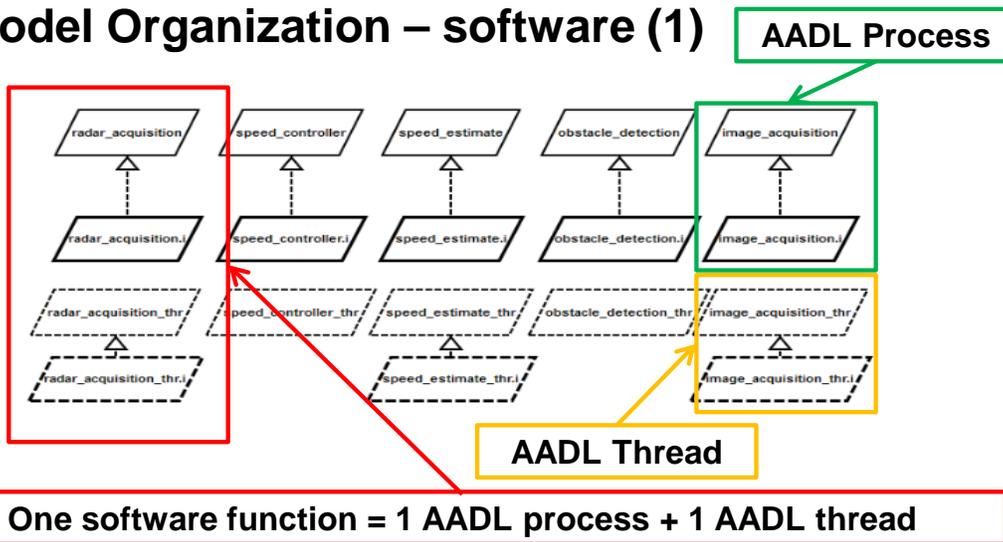


**032 About the organization of the platform, we have a processor; and then we extend the processor ECU with one connector or two connections. For the bus we associate different properties, the latency budget, the bandwidth capacity. We have also a property to say how much time do you take to acquire the bus or to transfer a bite. Okay?

So these kind of information are used when you analyze the system. Basically the transmission time will be used for the latency analysis. Okay? And in the processor I can say what type of network I'm using with this processor. Basically here I say: This ECU uses a CAN network.

Model Organization – software (1)

Model Organization – software (1)

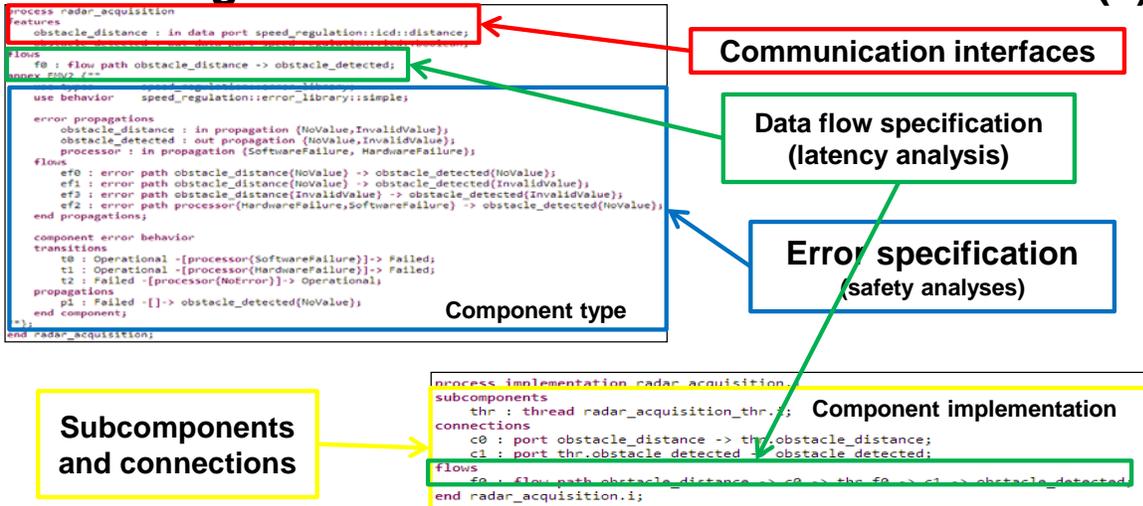


**033 For the software, for each function I have a process. So one software function is one AADL process with one task; what you call the AADL thread.

Also you can see that the AADL process just a line; and for the thread it's a dotted line.

Model Organization – software – textual notation (1)

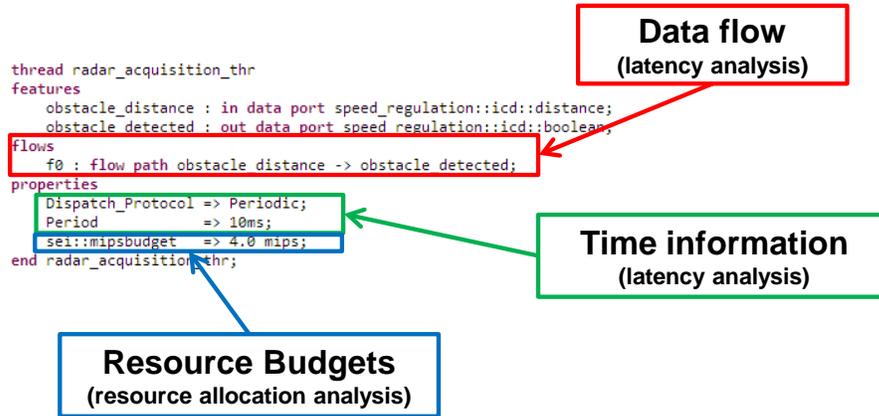
Model Organization – software – textual notation (1)



**034 So we also define the communication interfaces. You see in the textual language. We have the features-- okay?-- and we also define what are the flows. So the features are the communication interface; and the flow says that if you have new data coming on this interface; or if you are just transmitting existing data, if it's just a data flow.

Model Organization – software – textual notation (2)

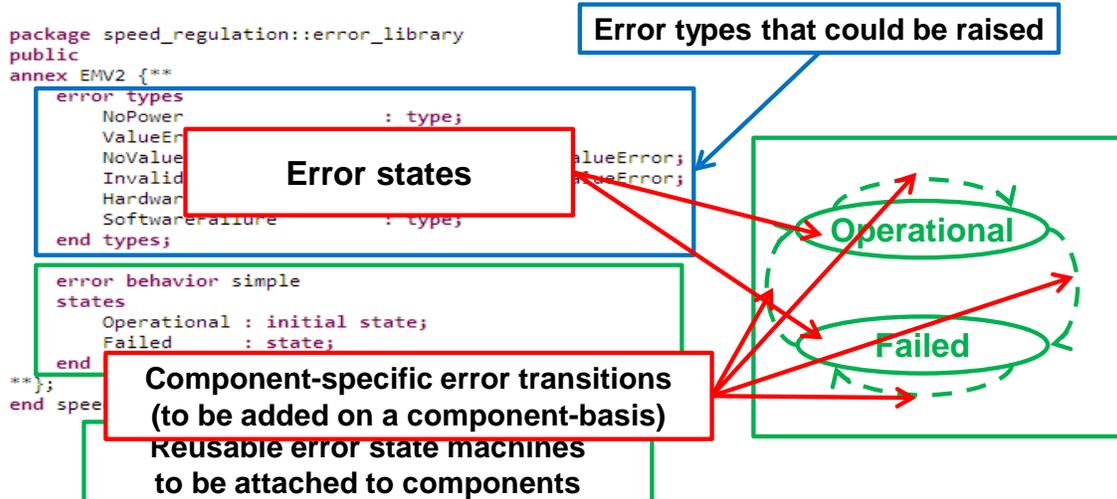
Model Organization – software – textual notation (2)



**035 All right.

Model Organization – safety specification

Model Organization – safety specification



**036 So this is the same thing.

And for the safety specification we can distinguish different error types in the system.

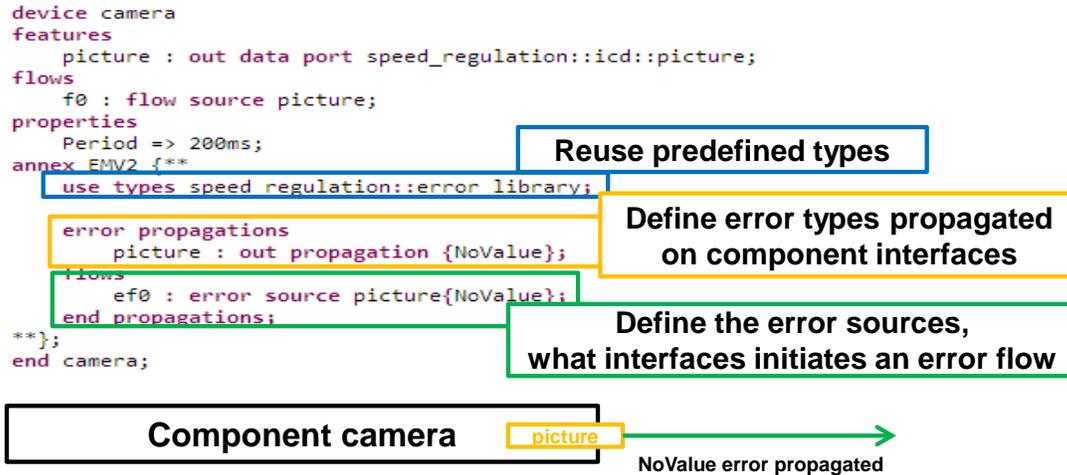
So, for example, if I have no power, or if I have a value error like out of bound error and stuff like this, if I have no value, for example if my component die and is no longer working, then I will have no value; or if I have an invalid value, if my component is failing and for example I have a static value, if I have a hardware failure on the processor, also have to have failure, for example, of the operating system.

And I can also specify an error behavior with different states; if my component is operational, if the component is failing or not. And I can reuse that.

And it's kind of a state machine, if you think about it. This was a question before. I can-- the initial state is operational; and then I can switch to the failed state later on. And this is error state; and I can add transitions.

Model Organization – define error flows – error source

Model Organization – define error flows – error source



**037 We can also define the error flows. So in the components I can define where the error can originate and also propagate in the architecture.

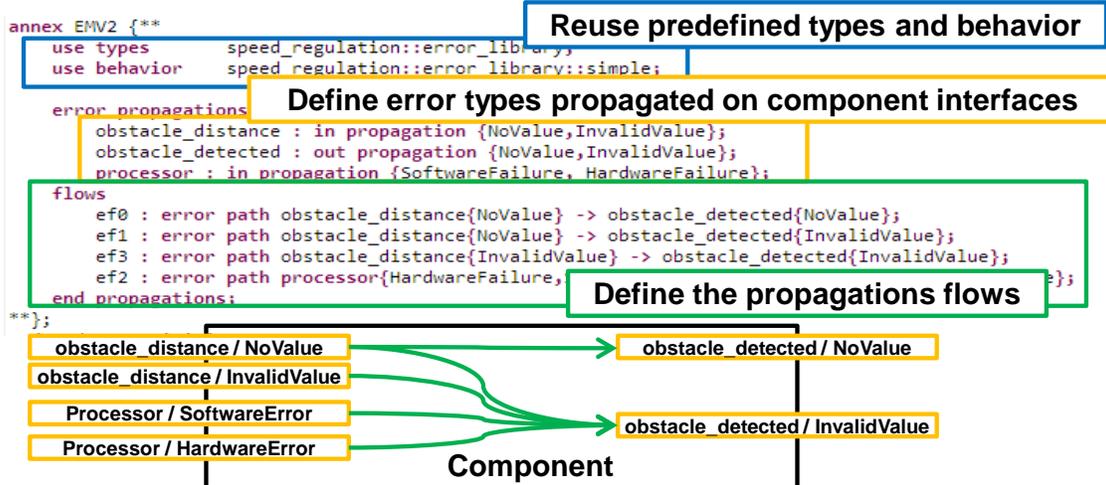
So for the camera, for example, if I look at the device camera, I can say that I use the different error types I showed you before; and the picture can have an error propagation that's no value.

In other words, when my camera is failing, I will have no value on the interface. So in other words this means that I will propagate the no value error the other components; and these components we have to take care of it or it will also impact these components. So if I have no value on the camera, maybe I cannot be able to activate the warning device or to detect a new obstacle. So this

is an error in my system; and I would be able to make use of it to see the different errors in my system.

Model Organization – define error flows – error path

Model Organization – define error flows – error path



**038 So in other words it means my camera, the value will not be propagated.

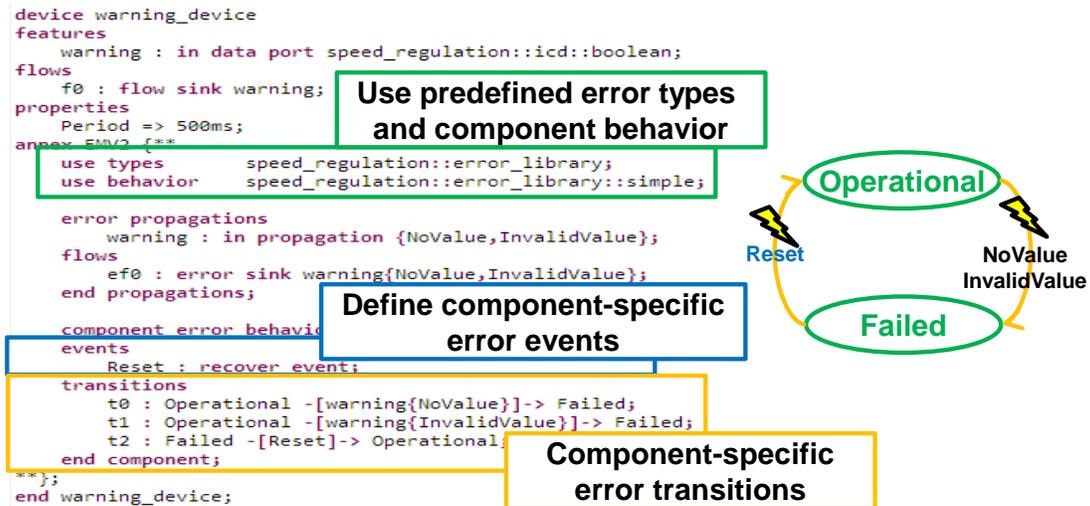
And then I can also describe the in and out propagation; and describe the error path.

So for example the obstacle, if I have no value as the obstacle distance it means that I don't have the value between my car and the obstacle, it would be propagated; and I have no value for the obstacle detected. In other words if I'm not able to evaluate the distance for the obstacle, I cannot say if I have an obstacle or not.

These are exactly these kind of things that will help me to detect the impact of the faults in the architecture.

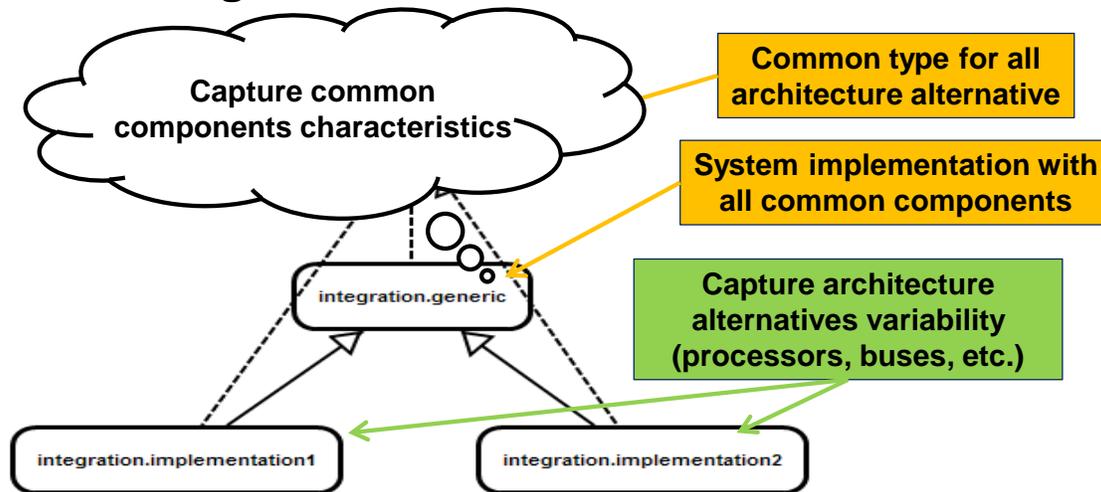
Model Organization – error sink & define component error behavior

Model Organization – error sink & define component error behavior



**039 Because I will be able to analyze the flows of the architecture.

Model Organization – architecture alternatives



**040 Let me show- let me show
you the model in a video

So if you import the model on your computer with the tool, you'll be able to have these different files; and you will see that we have the complete AADL textual model with a camera, and we add the property to describe what is the period of the camera, the compute execution time and all what we call the quality attributes of your system.

We also defined the error propagation through the interface; for example, the picture I can have no value.

And I added too some properties about the error. So no value means the description is no picture from the camera.

So this is for the device. For the errors, you'll see that we have different error types: No power, value error, no value error, invalid value; and so on, as we saw before.

In the ICD, this is the interface. I have all the different data types and the size of the data; and I will make use of this data on the interface to characterize what are the different data types; exchanged between the components.

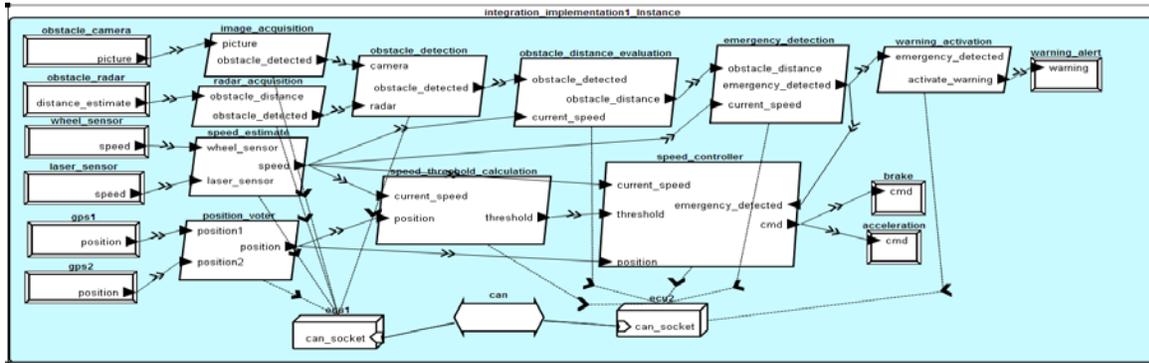
On integration I have a generic component that integrates all the common components of the architecture; and then I will capture the variability of the system.

So I have the generic component; and later on I define two different variations, one called implementation 1, as we see in this video; or implementation 2. And the difference, as you see, is the number of ECU components, over there and over here, and the number of buses.

Finally we have two other files: platform and software. Platform will be for the processors, the buses; and we'll also add some error description in the softwares about all the different processes, tasks with the error propagation added.

Architecture Alternative 1: model instance

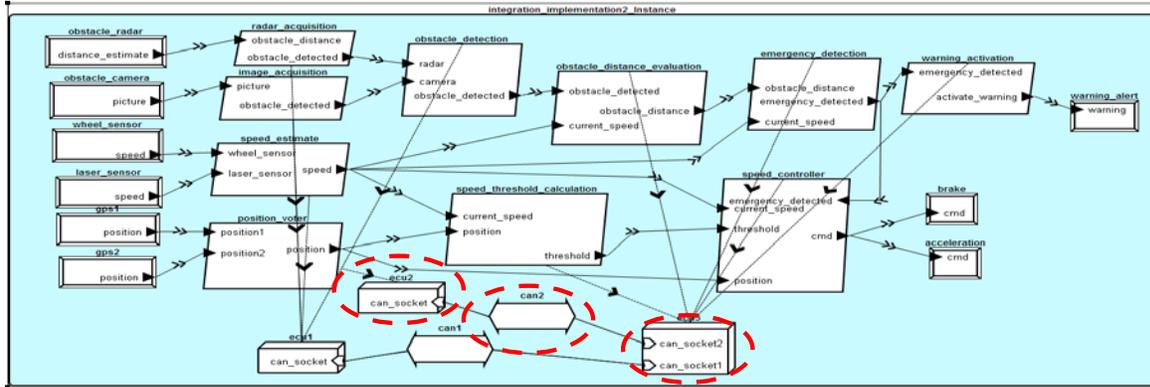
Architecture Alternative 1: model instance



**041 So when you look at the model, finally it looks like this with the AADL notation. So the graphical view of the model is the following for the architecture alternative number one: two ECU, two processors connected through a bus; and all the different software functions bind to either ECU1 or ECU2.

Architecture Alternative 2: model instance

Architecture Alternative 2: model instance



Variability Factors with Alternative 1

**042 Alternative number 2, what we have between the two is we add another processor and another bus. This is the only difference we have.

Agenda

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion



**043 So right now what we're going to do is try to analyze the architecture. But do we have any questions?

Shane McGraw: We do have a number of questions coming in. Johan wants to know: Is the OSATE tool qualified for being used in the context of ISO-2626.2, Functional Safety?

Julien Delange: All right. So it's interesting because the example is a car system. So ISO 2626.2 is a standard for automated system; and as of today the open-source AADL toolset can be used to analyze and validate automated architecture.

On the other hand we didn't have any qualification status. But I will be glad to discuss that if there is any requests to do so and to address that concern.

Shane McGraw: Okay. Han would like to know: Is there a translator from AADL to UML or from UML to AADL?

Julien Delange: So there is-- in OSATE there is no capability like this; for many reasons. Software for UML is mostly graphical language; and we can't make-- if we make an import or export function, it should be really tool dependent.

So we are aware of different initiatives of people that are translating AADL into another notation or importing a notation into AADL; for example, importing a SCADE model or a Simulink model. I also see SysML models with tools like enterprise architects.

The thing is it's possible. Something else that's really important is there is a profile for UML called MARTE; and MARTE, we have a good contribution with the UML community to interface UML and the concept within AADL with UML.

Shane McGraw: Great. Let's get one more from Lee-Anise asking: Do you have security specifications to prevent bugs inducing into the system?

Julien Delange: To prevent bugs in OSATE?

Shane McGraw: Yes.

Julien Delange: Okay. So right now we have a lot of code review on

OSATE to make sure that the code is- does not have any security issue; especially because the code is being deployed in critical environments. So we have this kind of concern right now.

Shane McGraw: Okay. Move on.

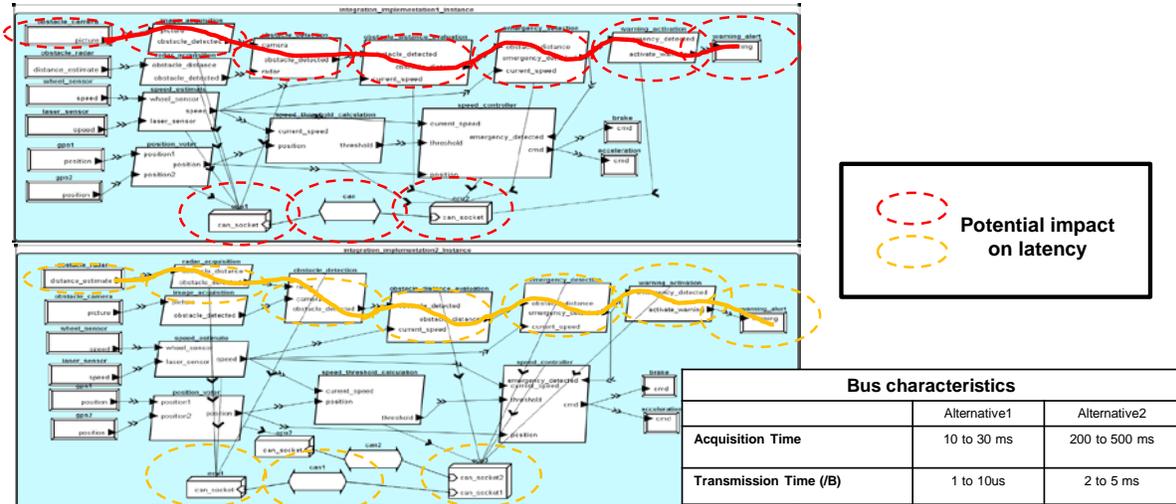
Julien Delange: All right, please continue to ask any question online. Shane will address them through the presentation.

So let me right now talk about the architectural analysis. So we have these architectures again.

The example is available online. Please try it; download OSATE and try it on your own computer. And contact the AADL community if you have any question. The community is really active; and I will give you some pointers at the end of the presentation to get support; and also more examples.

Latency Analysis, principles

Latency Analysis, principles



**044 Let me show you how the system analysis works.

So I know that a few of you were really concerned and interested by the latency analysis. Let me show you the principle.

As we said in the beginning of this presentation, we want the end-to-end latency to be less than 900 milliseconds. Okay?

So let's have a look at the flow between- in these architectures. So we have a flow for the software architecture and the similar architecture; and we can see the different flow contributors.

So this flow goes from the camera, the obstacle camera, to the warning alert. And in fact if you have a look, what is really important is the execution time of the different tasks.

But they're the same in both architectures.

What is really different is buses. And you see that the transmission time is not the same on both architectures.

So the thing is, if we have a look at the analysis tool, and if we have a look at how the tool works, we see quickly that what matters is the execution time of the device and also the bus latency associated with the different bus.

So if we check the flow latency, the tool will create automatically a report in OSATE; and the report is an Excel report. You can automatically open it using Excel.

So for the first architecture what is interesting is if you look at the end-to-end latency-- remember, the end-to-end latency, the maximum was 900; and you see at the bottom that the requirement is captured in the model at 900; and for these architecture candidates the max latency was 886. So these requirements is okay for these architecture candidates.

But if you think about this, the difference between the two architectures was the bus. So let's have a look at the impacts of the bus in the second architecture.

And you will see that between the bus takes more time to be acquired. Then my latency is one second and 350 milliseconds. So my requirement is not met. And I find that in

the model; and just by changing a single component my requirements is no longer a max.

Latency Analysis, results

Latency Analysis, results

Architecture Alternative 1



flow	model element	name	deadline or conn delay	total	expected
f0: End to End Latency report					
f0 (Synchronous)	device	obstacle_camera.f0	200.0 ms	200.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_camera.picture	0.0 us	200.0 ms	900.0 ms
f0 (Synchronous)	thread	image_acquisition.thr.f	50.0 ms	250.0 ms	900.0 ms
f0 (Synchronous)	Connection	image_acquisition.thr.c	0.0 us	250.0 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_detection.thr	100.0 ms	350.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_detection.thr	30.00125 ms	380.00125 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_distance_eval	10.0 ms	390.00125 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_distance_eval	0.0 us	390.00125 ms	900.0 ms
f0 (Synchronous)	thread	emergency_detection	14.0 ms	394.00125 ms	900.0 ms
f0 (Synchronous)	Connection	emergency_detection	0.0 us	394.00125 ms	900.0 ms
f0 (Synchronous)	thread	warning_activation.thr	2.0 ms	396.00125 ms	900.0 ms
f0 (Synchronous)	Connection	warning_activation.thr	0.0 us	396.00125 ms	900.0 ms
f0 (Synchronous)	device	warning_alert.f0	500.0 ms	896.00125 ms	900.0 ms
f0 (Synchronous)	Total		0.0 us	896.00125 ms	900.0 ms

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

Architecture Alternative 2



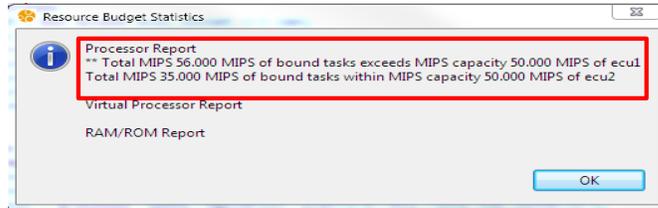
flow	model element	name	deadline or conn delay	total	expected
f0: End to End Latency report					
f0 (Synchronous)	device	obstacle_camera.f0	200.0 ms	200.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_camera.picture	0.0 us	200.0 ms	900.0 ms
f0 (Synchronous)	thread	image_acquisition.thr.f	50.0 ms	250.0 ms	900.0 ms
f0 (Synchronous)	Connection	image_acquisition.thr.c	0.0 us	250.0 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_detection.thr	100.0 ms	350.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_detection.thr	100.00625 ms	450.00625 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_distance_eval	10.0 ms	460.00625 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_distance_eval	0.0 us	460.00625 ms	900.0 ms
f0 (Synchronous)	thread	emergency_detection	4.0 ms	464.00625 ms	900.0 ms
f0 (Synchronous)	Connection	emergency_detection	0.0 us	464.00625 ms	900.0 ms
f0 (Synchronous)	thread	warning_activation.thr	2.0 ms	466.00625 ms	900.0 ms
f0 (Synchronous)	Connection	warning_activation.thr	0.0 us	466.00625 ms	900.0 ms
f0 (Synchronous)	device	warning_alert.f0	500.0 ms	966.00625 ms	900.0 ms
f0 (Synchronous)	Total		0.0 us	966.00625 ms	900.0 ms

ERROR: f0: End-to-end flow f0 calculated latency (Synchronous) 966.00625 ms exceeds expected latency 900.0 ms

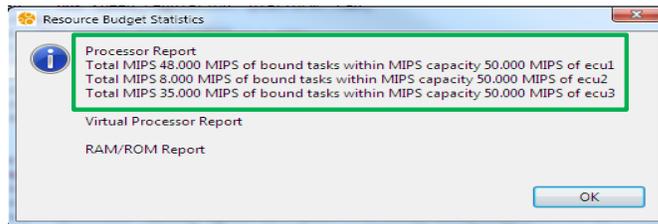
**045 So you see easily with these architectural descriptions that it's a perfect picture of the end-to-end latency is enforced. For the second architecture I have an issue, just by changing a component.

Resources Allocation Analysis, results

Architecture
Alternative 1 



Architecture
Alternative 2 



**047 f I look at the resources analysis, I will analyze the budget, the processing budget on the processor and also in the task. In other words, my processor has to provide enough processing capacity for my task.

So if I have a look at my processor, I define what I call the MIPS capacity that define the processing capacity I can provide to my task. And in my task what I'm going to define is the MIPS budget: How many MIPS I need to be executed.

And with the information in mind, I can start to make a resource budget analysis. And in my architecture I can see that-- for example, in my first architecture the total MIPS for the software size is 54 MIPS, and my processor can provide only 50. So I can't execute this architecture with only one processor.

So I have to make a choice. Either I use I optimize my software or I use a more powerful processor. The second architecture candidate is okay. With the three processors there is one function that is really consuming in terms of processing capacity and it's executed on the second processor.

So with three processors I'm fine; and all the processors are connected to all the tasks. They have no problem on that.

So if you have a look at the different architectures, the first one--

I have an issue- has an issue; and the second one doesn't have any issue at all.

For the first one I have a budget error in terms of processing capacity; and for the second one I have no problem at all.

So let's have a look right now at the safety analysis.

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



**048 With a different safety analysis tool.

The first one is what you call the Functional Hazard Analysis. And this is a list of all the errors in my system.

The second one is the Fault-Tree Analysis. I will show you the top level faults and see all the contributors of this faults.

And after that I will have the Fault-Impact Analysis. So I will show how an error propagates into the system.

I will show you first how these tools work.

So let's have a look first at the Fault-Hazard and Impact Analysis.

So when I do the safety analysis, what I have to provide is what type of fault I propagate into my system.

For example, for the camera, these components, I will have to define the out propagation and the out source.

So I will say: My camera can propagate a no value error, I can have no value in my camera, no picture-- and this will be in here also-- and it will propagate all of the other systems.

But also to generate safety documentation, I need to have some comments, some documents, where these are already defined: Where in the specification; what is the impact; do we have any comments; what is the reference with the other models and so on.

So we add this information. And then when I'm supposed to receive an error, I have error sink.

So here for the warning device I have an in propagation for no value and an error sink. I can say: Hey on my warning it quits; I have no value; I don't know if I have to issue a warning or not.

And finally I have the error source and the error sink; but I also to define error path. So how the error propagates into the architecture.

What I can also do is I can say well I have no value from the camera, maybe it will translate it to an invalid value because maybe I will use the previous value or something like this. So I have to define the different mappings between the error types when I propagate the faults.

So in fact in the architecture I define the error flows between the different components; and I'm able with the analysis tool in OSATE to process this information and use the tool to generate what you call the Functional OSATE Assessment Report.

It will be a full report available as an Excel document. And I will list all the errors in my architecture.

The camera I can just propagate the no value at all on the picture. The radar, no distance; anything like this. Okay?

So I have all the errors. In the example we have only 20 errors. But the fact is when you have a real system, it's thousands of errors; and most of the time these documents are made manually. Here it's automatically generated. You don't need to do that manually. So you can update the documents as your architecture is evolving.

Something else we are doing is to have the fault impact. It's also an Excel spreadsheet; but it's not- it's not only the list of the errors, it's also how they propagate in the system.

So the obstacle camera, I can have no value in the picture. But after that it will propagate in the image acquisition components; and then it will also propagate later on to the obstacle detection and so on.

And finally eventually it will reach the warning alerts. So even if there was

no value in the camera can impact the warning alerts, the alert mechanism, to alert the driver.

So with this simple architecture and these 20 faults, you can see that finally with only 20 faults I can have 447- and 45 error paths in my architecture. So one error will have different impacts. And it's really critical because when you are- when are doing this manually it takes a lot of effort and a lot of time.

Safety Analysis, FHA, results

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

**049 So for the FHA what we see is for architecture number 1 I have only 15 errors; and for architecture number 2 17 errors.

The thing is the difference comes from the number of processors. Okay? In my second architecture I have more processors. So for sure I

have more faults. But the fact is I have more processors to isolate in terms of criticality and safety the different functions that are really critical.

Right now let's have a look at what we call the FTAs, the Fault Tree Analysis.

So the Fault Tree Analysis starts with the top level faults. My warning system or my self-driving car is not working. And then it will show all the error contributors.

So I defined my systems as my system is failing - in the failed state, if I have no brakes and if I have no acceleration. So I say my braking system is failing and my acceleration is failing. And then in my brake system I can define also the reason why the brake failed. So the brake can fail because I have no value on the command. Okay? I do not have break. I have an invalid value as well; can be an error.

I do exactly the same with the acceleration. In the acceleration I can explain what are the different errors and why I can be failing.

So with this description I have the top level error, I'm failing; and I can refine and define what are the different contributors and the reason why in the architecture I have this error that is coming. And I can use the AADL model to produce this fault tree with all the dependencies between the different faults.

So we have this built-in capability in OSATE; and it generates automatically the fault tree in different formats; one for a tool called Open FTA. That is an open-source tool to use Fault Tree Analysis and to visualize Fault Tree Analysis.

Also we have an interface with a commercial tool like CAFTA to visualize this fault tree.

So I will show you with OpenFTA; because it's an open-source tool and you can use it at home and on your computer and reproduce the example.

When you open the generic fault tree, you will see that-- finally you think software for that there is nothing. But if you zoom out, you see that in fact there are so many errors that can contribute to this top level error.

So let's zoom into some faults. So what this Fault Tree Analysis tool looks like-- I can see that okay, for example, I have an error from the voter; and the voter can say because I have no value from the first GPS or no value from the other GPS. So if my tool was on the GPS failed, then the position, what I would be failing.

And I can also associate some probability. And after that with some Fault-Tree Analysis tool I can make analysis about the probabilities that my system is failing or not.

So I have all the different errors I introduced in my system; an invalid value from the wheel sensor, from the laser sensor, from the camera and so on.

Safety Analysis, FTA results

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

**050 So if I have a look on the-

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



**051 On the fault impact, for
architecture 1 and 2 I have exactly
the same number of paths- of error
paths.

The thing, and what we have to
consider, is that I have the same
number of paths; that we have to
distinguish the paths according to the
criticality of the components.

Analysis Summary

Analysis Summary

	Architecture 1	Architecture 2
Latency		
Resources Budgets		
Safety		
Cost		

What is the “best” architecture?



**052 So in terms of safety the thing is architecture number 2 will be better because we isolate the different criticality levels.

For the resource budgets the first one is not- is not- is the best one because the processor one is overloaded. And for architecture number two, in that case I can enforce all the resources.

What we have to keep in mind also is-- something else we didn't capture in the model-- is the cost. In Architecture number 1 you need two processors and one bus. For Architecture number 2 you need three processors and two buses.

This is really important because when you have a car the cost of the processor, the cables, the bus and so on really matters; because you have

to produce thousands and thousands of units. In some other domains, maybe it doesn't matter all that much because you only produce a couple of items, a couple of products.

So the thing is-- the question in the beginning was: What is the best architecture? And the best architecture will depend on your requirements.

If you are cost focused or if you really care about safety, resource budget and so on, in that case if you are really focused on the costs, maybe the first architecture will be better; and you just have to have a more powerful processor just to optimize your software and then the resource budget will not be an issue.

Conclusions

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

**054 So let me conclude. First, I hope that this demo gave you a good

introduction to model-based system design and analysis.

So we can see that just with the model, without implementing the system, we can already find many issues that you will find in the integration of the system.

So this is a new method a new tool. Okay? And this is evolving. But this is already ready to use on real projects; and we have several people that are already using this kind of tool to evaluate their systems and their specific architecture.

The AADL with the specialized components really helps you to reason about the architecture. What kind of- what kind of components are you using? You can have your own products. You can extend the language; and so on.

And we have a good workbench to support this language. So we have the graphical notation; the textual one notation. And all these tools are open-source and available online.

Useful Resources

Useful Resources

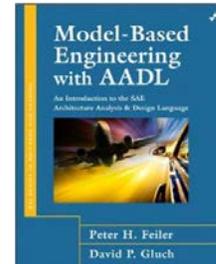
AADL wiki – <http://www.aadl.info/wiki>

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



**055 Useful Resources. Software for the AADL wiki. You have plenty of information according to- according many examples. Case study and so on you can access for free online.

A book that was written by Peter Feiler at the SEI and David Gluch also at the SEI, about AADL and the use of the different language features.

We have also a lot of blog post series about AADL; and we update the blog posts with new research we're having at the SEI.

We are also really active on the AADL mailing list. The community provides good support and help you if you have any questions.

Something else is next- this month-- sorry-- in Valencia there is a Models Conference; and we have an AADL workshop. It's called Architecture-Centric Virtual Integration. We have a lot of good submissions and good papers. The proceedings are already online; and if you want to check- to check them out please do so.

Also if you want to attend the Models Conference, or if you are planning to attend, we would be really happy to see you at the workshop.

Questions & Contact

Questions & Contact

Dr. Julien Delange

Member of the Technical Staff
Architecture Practice
Telephone: +1 412-268-9652
Email: info@sei.cmu.edu

Web

www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

U.S. Mail

Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Customer Relations

Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257



Software Engineering Institute

Carnegie Mellon University

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

56

**056 Thanks. And if you have any questions please let me know. Shane, I hope that we have many questions.

Shane McGraw: Sure we have lots of questions coming in for Julien. And just before we get to that, just a

reminder folks, before we close out in about eight minutes, to fill out your survey as your feedback's always greatly appreciated.

Julien mentioned the book. If you go to your Materials tab on your console, you'll see a discount for that book available; and also an upcoming training course on AADL from the SEI. So we hope you'll look into that as well.

So let's get into the questions. About seven minutes left with Julien.

Derek wants to know: Can AADL be used along with ArchiMate? If you're familiar with that.

Julien Delange: I'm not familiar with ArchiMate. So if Derek can send me some pointers, I will be really happy to answer.

Shane McGraw: Okay. Let's move on to Anon. He wants to know: Will models add metadata to systems?

Julien Delange: What? Sorry Shane.

Shane McGraw: One more time from Anon: Will models add metadata to systems?

Julien Delange: Okay. So in other words, if I understand correctly, it's what are the metadata associated to the model? So in that case this is what we call- what we have with the different AADL properties; and we can capture different metadata in the

different components in the system.
Or with the different extensions; we
can use them, like the AADL Annex
and so on.

Shane McGraw: Okay. Johan
asked: For the latency calculation do
you support randomization? Can you
then model check the results?

Julien Delange: All right. So that
is- this is really interesting because
we have different work for the
latency.

First of all there is something with
the model checker. In that case you
have to export the AADL model in the
formal notation; like Petri nets or another
formal language

The thing is right now we are not
doing that. We are really checking at
the high level. But there is many
tools that export the AADL notation
into this formal language and do so.

The thing is we are-- the tool that is
working on OSATE is a tool that is
using the AADL properties. So it's a
really high language verification. And
then you can refine it with model-
appropriate model checker. And
some- we have some users in some
companies that they are working
with.

Shane McGraw: So just a
reminder. So Julien talked about the
OSATE tool earlier. We'll send out a
follow-up email tomorrow of where to
get the archive and the recording;
and we'll include that information

where they can go and get that, the tool, for free there.

Okay next question from Jong-Wai asking: Can the end-to-end latency analysis be integrated with scheduling analysis in OSATE?

Julien Delange: So again this is something that can be integrated for sure. This is not something that is provided right now in the toolset and available in the open-source version. But this is something that can be done; and some tool like AADL Inspector from a vendor in France provides also latency analysis, scheduling analysis and so on. In OSATE right now we have the budget analysis, the latency analysis, that are available.

Shane McGraw: Just a follow-up in case it's relevant here. And this is also from Jong-Wai asking: Does OSATE scheduling analysis support hierarchical scheduling mechanisms such as ARINC 653?

Julien Delange: All right. This is a really good question. So right now ARINC 653 can be used for the resource budgets. For the scheduling analysis, we don't have scheduling simulation. We have validation; like the tool I showed.

But we have also some export to a scheduling validation tool like AADL Inspector or Cheddar or MAST.

So this is something that already exists. Unfortunately it's not

available in the OSATE toolset. But the semantics of AADL can provide everything that you need to do so.

Shane McGraw: Okay. From Johan asking: is there an FMEA module planned as well?

Julien Delange: Yes. In fact if you look at the safety analysis tool, what is done by the fault impact is really similar to the FMEA; and customizing the fault impact reports will provide the ability to generate the FMEA report as well.

Shane McGraw: Okay. Rob would like to know: For a system- for a system with thousands of signals and hundreds of tasks and multiple processors, each under a safety critical scheduler, one, how much effort and time is required to set up such a model without being overcome by events; and two, how hard is it to verify a complex model is correct?

Julien Delange: I love this question. I love it. Thank you for asking.

Shane McGraw: And that was-- yes Rob; yes so yes.

Julien Delange: All right. Thanks again for asking.

Scalability is a big issue for many model-based tools. And in fact we have many users that are asking several questions: How scalable is your tool?; and second, what is the learning curve?

And I understand. Because when you- when you have to incorporate a new technology, a new language, you are looking at the cost of acquiring- getting the new technology; how much it costs you and what are the benefits?

We have a blog post on that topic that shows that within a couple of days or weeks a team of students, Master's students, were able to acquire the main concepts of AADL, to also learn the safety annex of AADL, access information and generate all the documentation for a real generic system.

So we have that on the SEI blog post. You can check out at blog.sei.cmu.edu.

As for the scalability of the system, some years ago we had a request from a customer; and the customer had thousands and thousands of components. And OSATE had a hard time to process that because of many internal notation details; for example, the Java heap and so on.

And we make a lot of improvements into the OSATE toolset. And right now we are scalable to analyze a system with more than- with a couple of thousand components.

Shane McGraw: Okay got a minute left. We're going to do one more question for Julien from Anon asking: Are there supporting design patterns in AADL to support analysis and provide proven ways?

Julien Delange: Yes. So we are working also on this. And if you look at the AADL wiki we have a special section dedicated to modeling patterns.

So we have different patterns; and with these patterns we provide different properties. And it's already filled. So you just have to customize the different properties, okay, with your own requirements; and also the different safety aspects and so on. So please check aadl.info/wiki. Thank you.

Shane McGraw: Julien, great presentation. Thank you very much for your time today.

Folks, we appreciate you joining the SEI Webinar Series. Look for an email tomorrow with some follow-up information on this topic and how to retrieve the archives. Have a great day.

Julien Delange: Thank you.