HTO Panel
“What evidence is sufficient?”

Jerome Hugues

SSD/ACPS/MBE

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

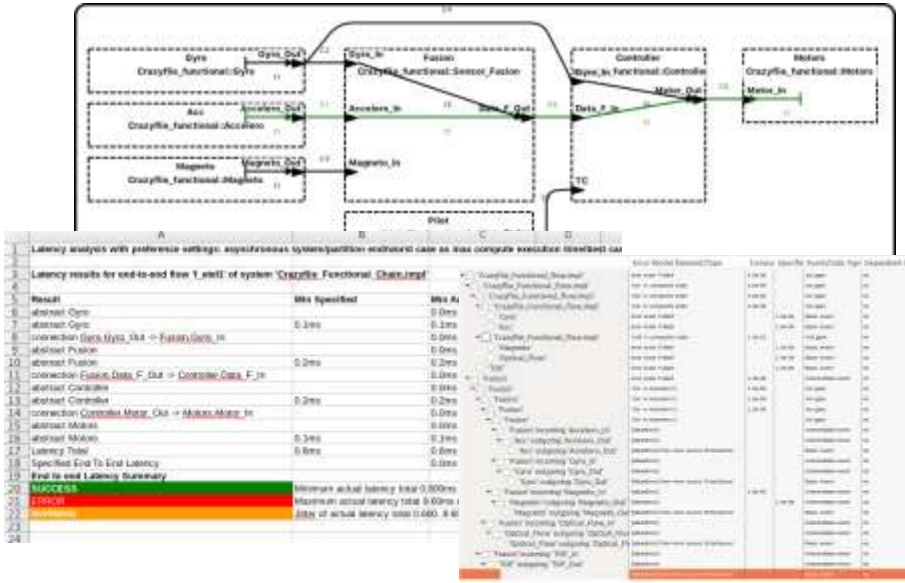
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM22-0338

On evidence for avionics critical systems

Margaret Hamilton, lead software engineer of the Apollo Project, stands next to the code she wrote by hand and that was used to take humanity to the moon. [1969]

to



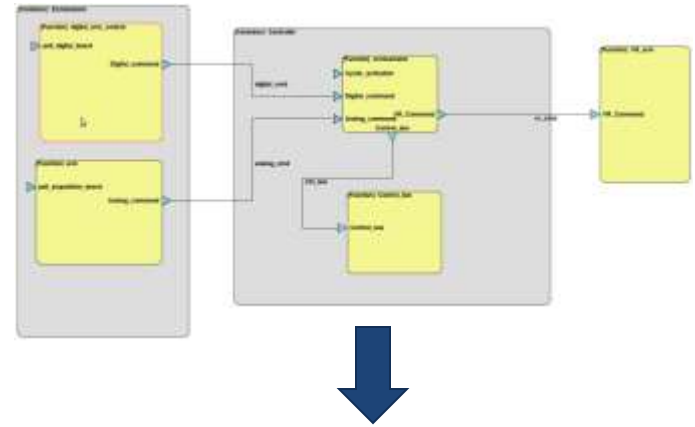
Position

Models are a better way to organize syst. engineering artefacts,

An analysis applied on a model produces evidence

@ESA, the TASTE project: Models as Code

https://taste.tools/



AADL+SDL+Simulink to engineer flight software

Automated toolchain to

- Generate code and test infrastructure
- Gather coverage metrics, WCET,
- Check code patterns, suspicious code
- Prove the correctness of middleware for Ada targets
- Perform run-time verification for C/SDL targets

Automation: easy/time consuming evidence gathering

- Provide ”engineering evidence”, ECSS-E-ST-40C
- Review with customer (regulator) to evaluate sufficiency and adequacy.

From “models producing evidence” to “evidential model-based process”

[1] *”Four Pillars for Improving the Quality of Safety-Critical Software-Reliant Systems”*, by Feiler et al.
[2] *“ROI Analysis of the System Architecture Virtual Integration Initiative“,* by J. Hansson et al.

MBSE := { Reqs + Models } + Analysis

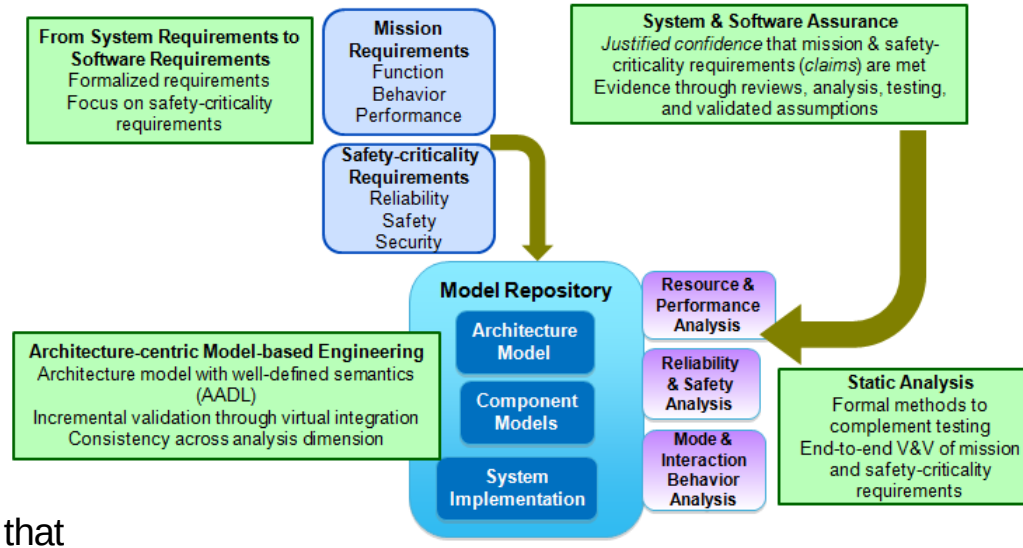
Cost-effective: automation + improved req. [2] because of faster feedback and earlier analysis

Sufficiency of an evidence? [1]

- Confidence in the engineering process
- Conformance to established norms

⇒ Confidence map, i.e., Toulmin-style argument that connects requirements, engineering artifacts, analysis, and results.

Support for evidence acceptability: elicit in a precise way the rationale for this evidence: why? How? …



Backup slides

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

SAE International AADL Standard Suite (AS-5506 series)

Core AADL language standard [V1 2004, V2 2012, V2.2 2017, V2.3 2022]

- Focused on *embedded software system modeling, analysis, and generation*

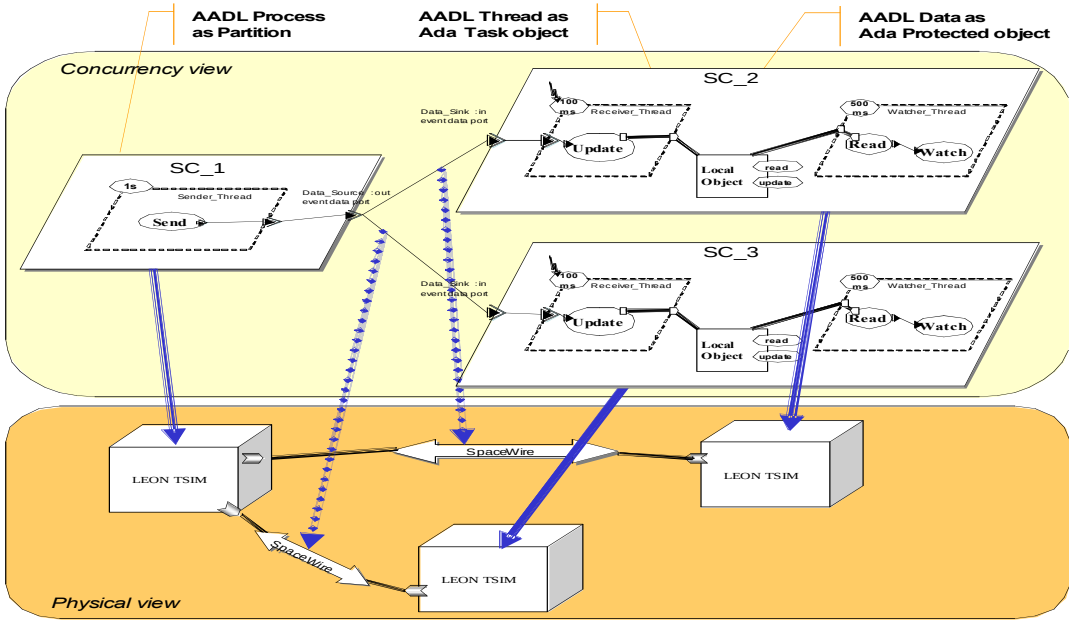
Standardized AADL Annex Extensions
- Error Model language for safety, reliability, security analysis [2006, 2015]
- ARINC653 extension for partitioned architectures [2011, 2015]
- Behavior Specification Language for modes and interaction behavior [2011, 2017]
- Data Modeling extension for interfacing with data models (UML, ASN.1, …) [2011]
- AADL Runtime System & Code Generation [2006, 2015]
- FACE Annex [2019]

- Evidence produced as a result of automated tool-supported analysis
  - Performance analysis: worst-case response time, schedulability of the system
  - Safety analysis: computing fault trees, probability of reaching an unsafe state
  - Automated model review: conformance to modeling guidelines
  - Code generation: generating “correct-by-construction” software

TQL-5, i.e. verification tools. Output :- evidence, part of some argument

TQL-1 tools, i.e. output is part of final system

AADL usages



General AADL tool: OSATE

Requirements verifications: ALISA

Middleware synthesis : Ada, C (POSIX, ARINC653), RTOS: Ocarina

Scheduling: Gateways to Cheddar, MAST

Error modeling, FTA, FMEA: OSATE

Model checking: Petri Nets (Time and CPN), LNT

Security Analysis, CI/CD, […]

AADL goal is to provide “ground truth” for describing safety-critical systems

“What evidence is sufficient?” – Context

evidence /ˈevədəns/

- the available body of facts or information indicating whether a ~~belief or~~ proposition is true ~~or valid~~.
  - Note: as a scientist, I concentrate on formal propositions, not beliefs.

“As a consequence of the above, the system will operate as required” is a master evidence, built from many partial evidences, collected from multiple artifacts (standards, engineering, tests, …).

“What evidence is sufficient?”

- Why do we need this evidence? – “who” is asking?
- How is the evidence produced? – by “who”? Which artifact? tool-supported vs. human expertise
- Who is reviewing this evidence? – a human? Inputs to another tool/process?
- What does sufficient mean? Minimal amount of work to produce it? To accept it? Do we have well-defined criteria for acceptability?
- What is the relationship between an evidence and an argument?

AADL Model-Based Engineering for Cyber-Physical Systems@SEI

Create the best design that holds up over time as the system evolves. + Test the design without having to write any code. = Build a single model to assess hardware and embedded software before the system is built.

At the SEI, my team works on analysis techniques for safety-critical models

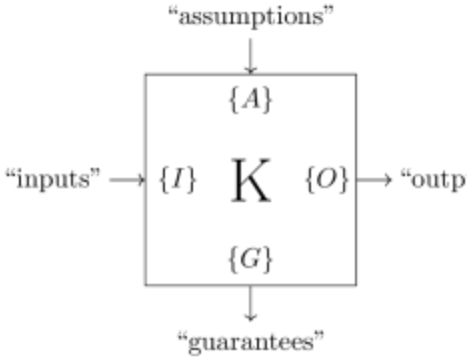
- Analysis operates either on the system, or a model of the system
- Models have different levels of fidelity:
  - Either ***an abstraction of an existing*** system
  - Or ***a blueprint of a future*** system
- Our main line of research is on AADL, an architecture description language
  - Scheduling or latency analysis, generation of fault trees, code generation, …
  - Each result from analysis is some form of an evidence

From analysis to Assume/Guarantee contracts

Brau, Hugues, Navet, https://doi.org/10.1016/j.scico.2017.12.007

Definition (Contract). A contract related to an element is a tuple K=(I,O,A,G):

- I are **inputs**: the data required by the element,
- O are **outputs**: the data provided by the element,
- A are **assumptions**: the properties required by the element,
- G are **guarantees**: the properties provided by the element.

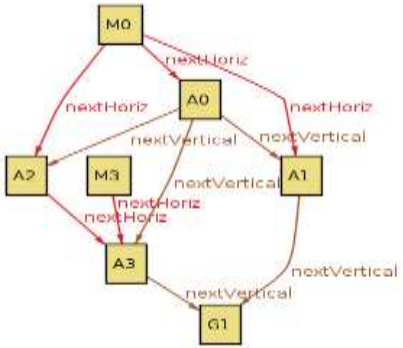


A/G are logic predicates, G is a fragment of an evidence built using formal methods

Contracts are amenable to composition A/G -> A/G' + A'/G'' + A''/G

These are have "raw evidences", insufficient because

- Confidence in the predicates? analysis methods ? debatable
- Connection with system-level safety process? To be established



Time Sensitive Engine Control Problem

From CMU/SEI-2015-TR-006 by Feiler et al.

Stepper motor (SM) controls a valve

- Commanded to achieve a specified valve position
  - Fixed position range mapped into units of SM steps
- New target positions can arrive at any time
  - SM immediately responds to the new desired position

Safety hazard due to software design

- Execution time variation results in missed steps
- Leads to misaligned stepper motor position and control system states
- Sensor feedback not granular enough to detect individual step misses

Software modeled and verified in SCADE
Full reliance on SCADE state machines
Problems with missing steps not detected

Software tests did not discover the issue
Time sensitive systems are hard to test.

Two Proposed Solutions
1. Sending of data with 12ms offset
2. Buffering of commands

No analytical confidence that the problem will be addressed

Analysis Results and Solution

Architecture Fault Model Analysis

- Fault impact
  missed steps
  - Early arrival
  - Step increm
  - Transient message corruption or loss
- Understandin
  - When is ea
  - Guaranteed
    for step incr
  - Built a confidence map, i.e., Toulmin-style argument that connects requirements, engineering artifacts, analysis predicates, and results

Archetype of a ”sufficient” evidence supported by model, analysis results, and a confidence map.

Open questions: Is this affordable? An overkill?