Safety Analysis and Fault Detection Isolation and Recovery (SAFIR) Synthesis for Time-Sensitive Cyber-Physical Systems

NOVEMBER 15, 2022
Jerome Hugues and Keaton Hanna
The AI Fresh Breeze – Coming from an Iceberg

There is general enthusiasm for Increasingly-Autonomous CPS [1] to improve system efficiency (decrease # of operators), system capability (automate high-level tasks), and faster than human action.

Increasingly-Autonomous Systems embed advanced “intelligent” capabilities, from basic control to advanced AI.

But the fast pace of action and poorly-defined safety mechanisms make it impossible for a human to mitigate issues.

⇒ Distrust in system, longer V&V, or capability not deployed
⇒ May jeopardize capabilities of future DoD projects

The AI Fresh Breeze – Coming from an Iceberg

There is general enthusiasm for Increasingly-Autonomous CPS [1] to improve system efficiency (decrease # of operators), system capability (automate high-level tasks), and faster than human action.

Increasingly-Autonomous Systems embed advanced “intelligent” capabilities, from basic control to advanced AI.

But the fast pace of action and poorly-defined safety mechanisms make it impossible for a human to mitigate issues.

⇒ Distrust in system, longer V&V, or capability not deployed
⇒ May jeopardize capabilities of future DoD projects

SAFIR aims to
• Deliver advanced Safety Analysis techniques to
  - Implement Fault Detection, Isolation, and Recovery policies for time-sensitive IA-CPS

Case Study – UAV Patrolling

Let us consider a patrolling mission with the following:

- Need to find and detect intruders, recognize threats
- Partially known place: a factory, with slow-moving parts
- Both closed and open areas; wind, lighting conditions
- Tight maneuvers to enter/exit buildings, safety margins to avoid damages
- Autonomy in decision making and man-machine teaming

Research questions:

- How to guarantee that the system is safe to operate and will operate safely?
LSI SAFIR “Big Picture”

This autonomous CPS context is safe because...

- It does ___reqs; it is implemented by ___arch+code
- V&V ___activities demonstrate strict conformance
- It is operated safely, and hazards or threats are monitored and mitigated by ___FDIR.

LSI SAFIR is building a comprehensive approach to support Model-Based Systems Engineering and Safety Assessment through

- Architectural patterns – static and dynamic
- Tool-support analysis capabilities
- Argumentation to articulate all artifacts
LSI SAFIR Core Contributions

SAFIR focuses on the engineering of safety-critical IA-CPS at the architectural level; i.e., it

**Assumes** operational hazards, sensor/actuators faults or vulnerabilities, timing anomalies, and AI functions misbehaviors are known

**Guarantees** the architecture properly mitigates faults down to the implementation through

- Fault taxonomy, guidelines for selecting fault detectors
- Mechanized semantics of architectural description
- Representation of safety argumentation for review by certification authorities
LSI SAFIR Contributions

1. **Fault Detection, Isolation, and Recovery (FDIR)** provides the foundation to safety
   - fault/attack detection mechanisms based on reinforcement learning
   - FDIR patterns in the scope of autonomous systems
   - reference architecture for FDIR-capable systems

2. Mechanized semantics of architectural description

3. Generating arguments about system safety
Integrating AI Components – An Architecture Perspective

Dual challenge:

• Architecture -> AI/ML: Can the architecture mitigate erroneous inputs feeding LECs?
  - Fault detection + design guarantees

• AI/ML -> Architecture: How can we contain misbehaviors triggered by LECs?
  - Run-time assurance through fault detection, isolation, and recovery
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
- Evidence produced as a result of automated tool-supported analysis
  - Performance analysis: worst-case response time, schedulability
  - Safety analysis: eliciting unsafe scenarios, computing fault trees, probability of reaching an unsafe state
  - Automated model review: conformance to modeling guidelines
  - Code generation: generating “correct-by-construction” software
  - Assurance process with ALISA

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]
- FACE Annex [2019]
Fault Taxonomy Classification of Misbehaviors in IA-CPS

Notionally, the AI function receives data from the environment, takes a decision or acts on it.

Pragmatically, this data is processed by the CPS platform, inducing the following risks:

- Data tampering (no reference point) due to sensor faults, attacks, etc.
- Timing (blurring references) due to unavoidable latency in the system, timing violations or faults
- Pre-existing faults in non-AI CPS

Can be characterized using AADL EMV2 fault taxonomy

---

**Fault Propagation Taxonomy**

Part of SAE AADL Standard Suite
**Contribution #1: Fault Detection**

https://doi.org/10.2514/6.2022-0969, with GeorgiaTech

**Research question:** How to select a fault detector?

**Solution:** Define a decision procedure using the EMV2 fault taxonomy as pivot

Scenario :- set of errors it produces, error_set {scenario}
Detection :- set of errors it mitigates, error_set {detector}

A detector is efficient if  

\[
\text{error_set} \{\text{detector}\} \supseteq \text{error_set} \{\text{scenario}\}
\]

⇒ **Contribution:** Survey of fault detectors, mapping of fault/attack scenario to detection mechanisms
Contribution #2: Impact of Clock Offsets on RL Components

Research question: A CPS gathers rely on multiple sensors, interconnected through buses and CPUs. The architecture may induce delays and jitters that are clock offsets. Could these impact Reinforcement Learning (RL) based controllers?

Contributions:

• If clock offsets are bounded, RL can still converge, with theoretical proof without quantification, simulation results show limits in offsets that ensure convergence
• Quantization errors also impact RL convergence

To be published at the 2022 American Control Conference, with GeorgiaTech
LSI SAFIR Contributions

1. Fault Detection Isolation and Recovery provides the foundation to safety

2. Mechanized semantics of architectural description
   ⇒ Coq mechanization of AADL semantics: static, behavior, time, error

3. Generating arguments about system safety
AADL Layers

Static architecture: structural and configuration parameters

Component model

Property sets

Component model

Consistency rules

Component state machine

Port semantics

Communication semantics

AADL Standard

Dynamic architecture: state-based behavior

“Model-Based paradox”: claim that model-based approaches are better than document-based. But the semantics of a model is described as a … document

⇒ Prone to interpretation errors

Subset of the static structure of the AADL model. Analysis specific rules

Behavioral model

Different behavioral models, depending on semantics (timed, untimed, stochastic, etc.)
AADL Mechanization in Coq

Research question: Provide unambiguous formal semantics for AADL
• Reference for other tools
• Improved standard by eliminating corner cases

Solution: Mechanize the semantics of AADL using the Coq Interactive Theorem Prover (ITP)
• Static and dynamic semantics, property sets

Oqarina released as software artefact: github.com/Oqarina under the BSD (SEI) license.

SAFIR delivers formal semantics of AADL as Coq types, theorems, and operational semantics.
Oqarina

Import model from AADL toolsets
AADL meta-model as Coq types
Well-formedness rules
Operational semantics

AADL JSON Parser ➔ AADL Instance Coq data type ➔ Well-formedness rules ➔ Operational semantics

Coq data types ➔ AADL meta-model, typing rules, support for building a model
Well-formedness rules ➔ AADL legality/consistency rules (i.e., model validity)
Operational semantics ➔ how to "execute" a model (e.g., proof, model checking, simulation)

Features:
• Simulation of an AADL model by mapping to the DEVS formalism
• Mono-core scheduling analysis using the PROSA library
• To come: fault propagation and analysis
LSI SAFIR Contributions

1. Fault Detection Isolation and Recovery provides the foundation to safety
2. Mechanized semantics of architectural description

3. Generating arguments about system safety
   ⇒ Extend ALISA to generate Goal Structuring Notation reports
Argumentation

Safety argumentation requires a clear and unambiguous representation, beyond plain English.

Main Objective:
• Increase readability and understanding of safety arguments

Gives developers freedom to use:
• AADL modeling language
• Supported OSATE analyses
• ALISA’s rigorous assurance verification
• Generated clear and concise GSN arguments

Design time

Run time

This autonomous CPS, context, is safe because:
• It does [ ]
• It is implemented by [ ]
• V&V [ ] demonstrate strict conformance

It is operated safely, and hazards or threats are monitored and mitigated by [ ]
Goal Structuring Notation

Manages stakeholder and system requirements
**Verify**

claim EA_HS_1 [  
    rationale "Heat source ability to turn on and off is valid"  
    activities  
        isOnOffHC: ResoluteIsolette.IsOnOff()  
]

- Verification of requirement EA_HS_1
- **Rationale** keyword is used as a description of verification activities
- **Activities** keyword calls verification methods
  - Verification methods are call analyses built into OSATE to verify requirements against the AADL model
ALISA to GSN Mapping

**ReqSpec:** System requirements

- **Description:** A textual description of the req.
- **Category:** List of category references
- **Refines:** References to other reqs. that this req. refines
- **Rationale:** The rationale for the requirement

**Verify:** Claims to show reqs. are met

- **Rationale:** The rationale for the claim

**GSN:** Assurance case

- **Goal:** Claims about the system
- **Strategy:** Clarification of the argument strategy
- **Subgoal:** Refinements of higher-level goals
- **Context:** The context of the claim
- **Solution:** Direct evidence supporting a claim
Workflow

1. AADL Model
2a. Category File
2b. ReqSpec File
2c. Verify File
3. Generated Argument File
4. Import GSN Argument into AdvoCATE
SAFIR contributions

SAFIR extends virtual integration capabilities of MBSE (a.k.a shifting to the left) for AI CPS

#1: FDIR with and for AI function [Reports]

#2: Semantics of CPS Architecture improved V&V, simulation
   [Software] Oqarina, mechanization of AADL in Coq

#3: Reports generation part of ALISA assurance process
   [Software] OSATE output arguments in the GSN format
Our Team

Jerome Hugues
Principal Investigator,
Sr. Architecture Researcher

Aaron Greenhouse
Sr. Architecture Researcher

Keaton Hanna
Assistant Software Engineer

Sam Procter
Sr. Architecture Researcher

Joseph Seibel
Member of the Technical Staff

Lutz Wrage
Senior Member of the Technical Staff

Contact Us

To work with us, contact info@sei.cmu.edu
Document Markings

Copyright 2022 Carnegie Mellon University.

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM22-0882