

Integrating Safety and Security Engineering for Mission-Critical Systems

Sam Procter, Peter Feiler, Lutz Wrage, David Gluch

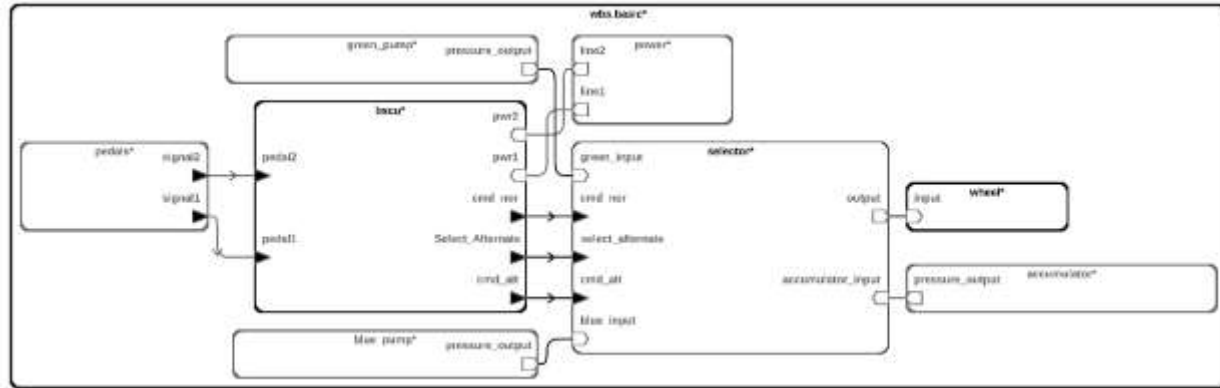
sprocter@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Challenges in Critical-System Safety and Security

Problem: Modern safety-critical systems are created by networking heterogeneous components together

State-of-the-art functionality now often requires exposure of those networks to the outside world, so security has become a concern



How should security analysis and design techniques be integrated with their safety-focused counterparts?

Solution: AADL is an internationally standardized architecture modeling language with a 15-year history of successful use in commercial, industrial, academic, and military applications

Agenda

AADL Foundation

- AADL: What and why
- OSATE

Research Paths

Industry Usage

Architecture Analysis & Design Language (AADL) Standard Targets Embedded Software Systems



SAE
AS 550
Stan
long-te
soluti
mult
model-b



2008 Aerospace industry initiative
use AADL over SysML and other
notations as it specifically addresses
embedded software systems

AADL captures mission and safety critical embedded software system architectures in virtually integrated analyzable models to discover system level problems early and construct implementations from verified models

SAE *International* AADL Standard Suite (AS-5506 series)

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

- Focused on embedded software system modeling, analysis, and generation
- Strongly typed language with well-defined semantics for execution of threads, processes on partitions and processor, sampled/queued communication, modes, end to end flows
- Textual and graphical notation
- Revision V3 in progress: interface composition, system configuration, binding, type system unification

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]

AADL Annexes in Progress

- Network Specification Annex
- Cyber Security Annex
- FACE Annex
- Requirements Definition and Assurance Annex
- Synchronous System Specification Annex

OSATE: Open Source Architecture Tool Environment

Eclipse-based Open Source AADL Tool Environment (OSATE)

- No cost license under EPL license
- Reference implementation of core AADL and annex standards
- Original release in 2004 with publication of AADL standard
- Maintained on GitHub
- OSATE release cycle
 - Quarterly stable release, Nightly builds
- AADL/OSATE community
 - Aadl.info and osate.org websites
 - Discussion forums
- Research prototyping platform
 - European projects, DARPA META and HACMS projects
- Used in pilot projects
 - SEI customer work, Industry and universities around the world

AADL excels at analyzing component-based systems by

- integrating annotated components
- running system-level analyses



Agenda

AADL Foundation

Research Paths

- Previous safety and security projects
- ISSE: Guidance and modeling strategy
- ISSE: Tooling and theory
- Role of research in AADL development

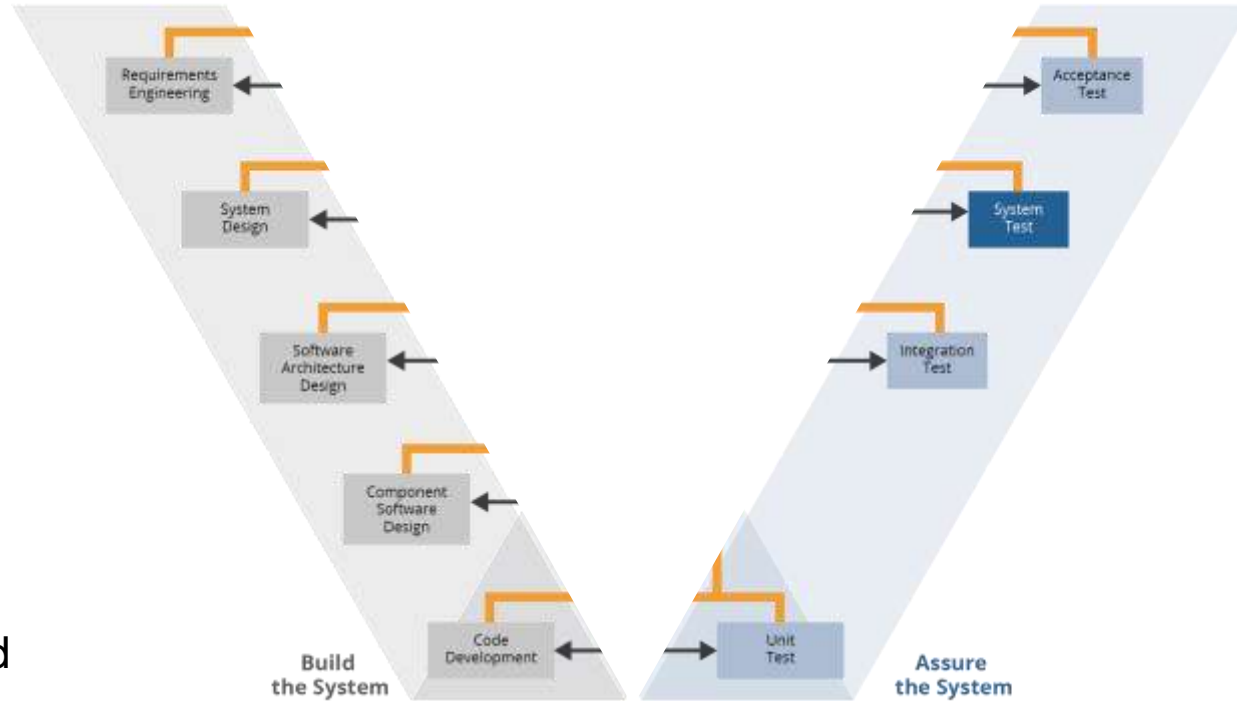
Industry Usage

Safety and Architecture

Problem: Safety problems are created early and caught late

Solution: ALISA is a tool kit and process for addressing system safety at the architecture level

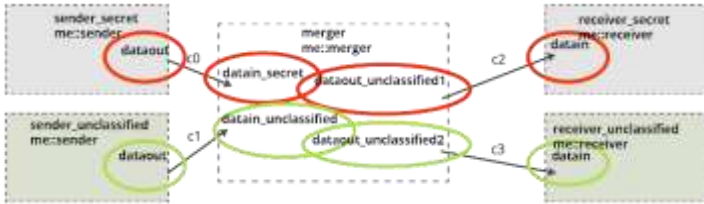
Safety is evaluated in the same way as other quality attributes: components are annotated, and then the integrated system is analyzed



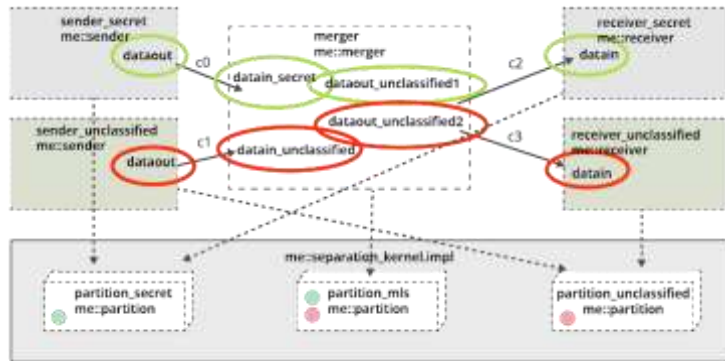
Source: "Architecture-Led Safety Process," CMU/SEI-2016-TR-012

Security and Architecture

Security policy vulnerabilities: Analyze information flows



Security enforcement vulnerabilities: Analyze deployment mechanisms



Previous security architecture research, such as the Multiple Independent Levels of Security (MILS), focused on separating security policy and enforcement

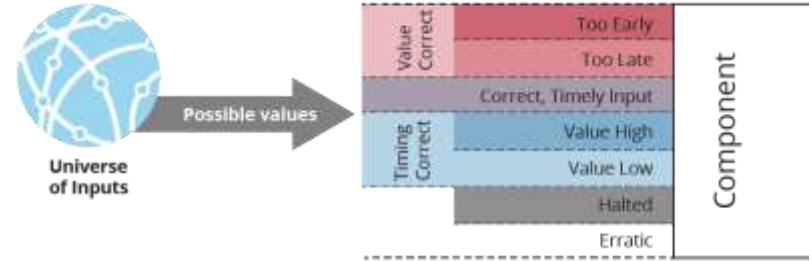
We added support for MILS within AADL/OSATE as well as code generation from models with security policies:

1. Security policy specification
2. Security policy enforcement
3. Generation and deployment of compliant systems

Modeling Security Requirements in the Context of Safety

Approach: Use effects-focused analysis and tooling

- When are various techniques appropriate?
 - Biba model (integrity)
 - Bell–LaPadula (confidentiality)
- What “building blocks” should be used?
 - examples: encryption, partitioning, checksums
- How should requirements be verified?



Measurement: Proposed user study (in FY 20) to measure qualities of design and analysis guidance

- Objective qualities
 - Number of issues found / avoided
 - Time required
- Subjective qualities
 - Quality of issues found / avoided
 - Complexity

ALISA used for Security

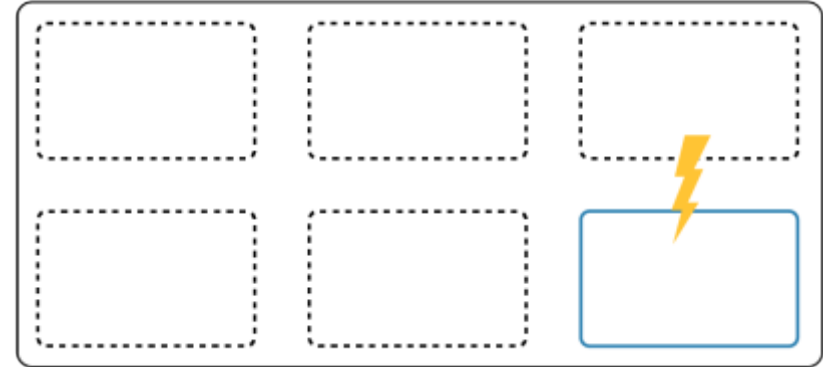
1. Requirement
2. AADL Model
3. Verification Plan
4. Verification
 1. Method Registry
 2. Method

```
system aircraftControl
  features
  method VerifyEncryption (component): "verify that a component has encryption
  A Resolute claim that the encryption property is true."
  has_encryption_form (element: aadl) : bool =
  has_property (element, SecurityEnforcementProperties::Encryption_Form)
    description "this confirms encryption."
  ]
  securityProps::level => critical;
end aircraftControl;
```

Using Theory to Guide Tool Development

Approach: Use fault-injection tooling

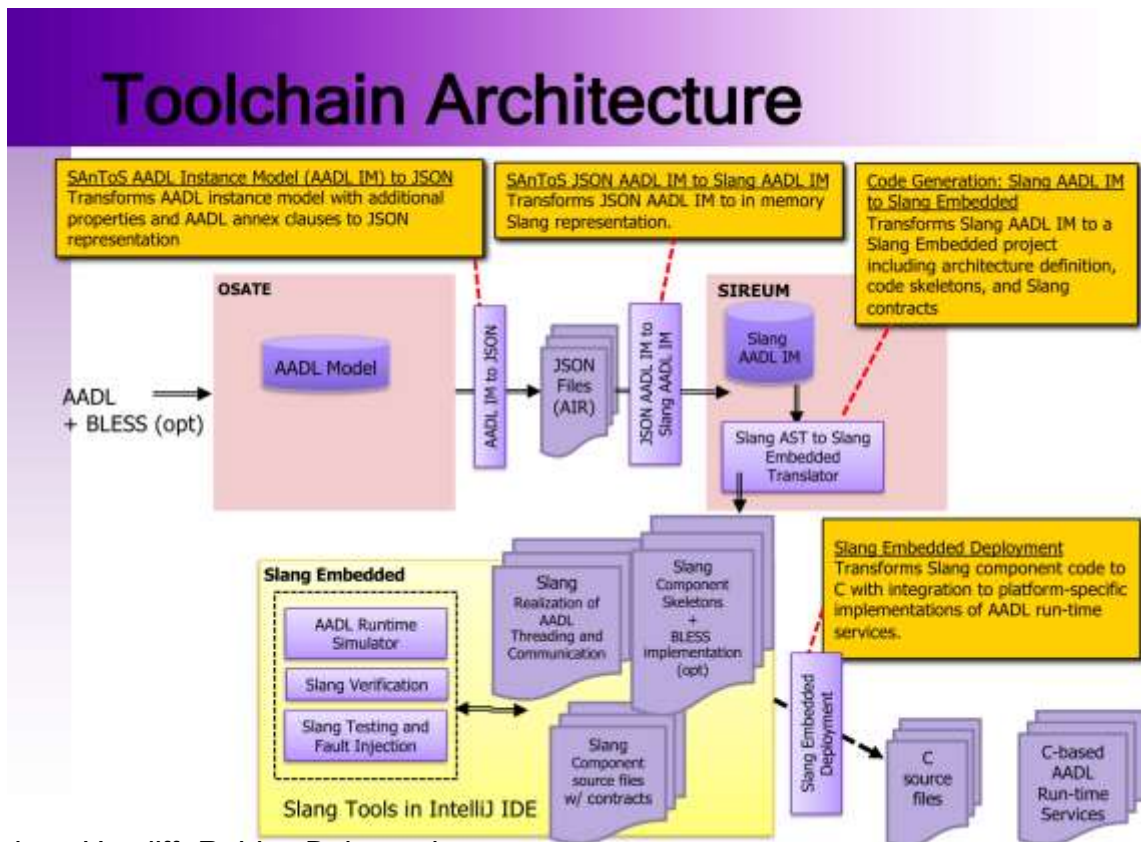
- Fault-injection pairs naturally with an effects focus
- Collaborators are building a large simulation and verification environment to enable this testing



Measurement:

- Current AADL can describe component behavior in the presence of errors
- This project will let us verify those descriptions

Slang – Simulation / Verification Environment



Kansas State University – Hatcliff, Robby, Belt, et al

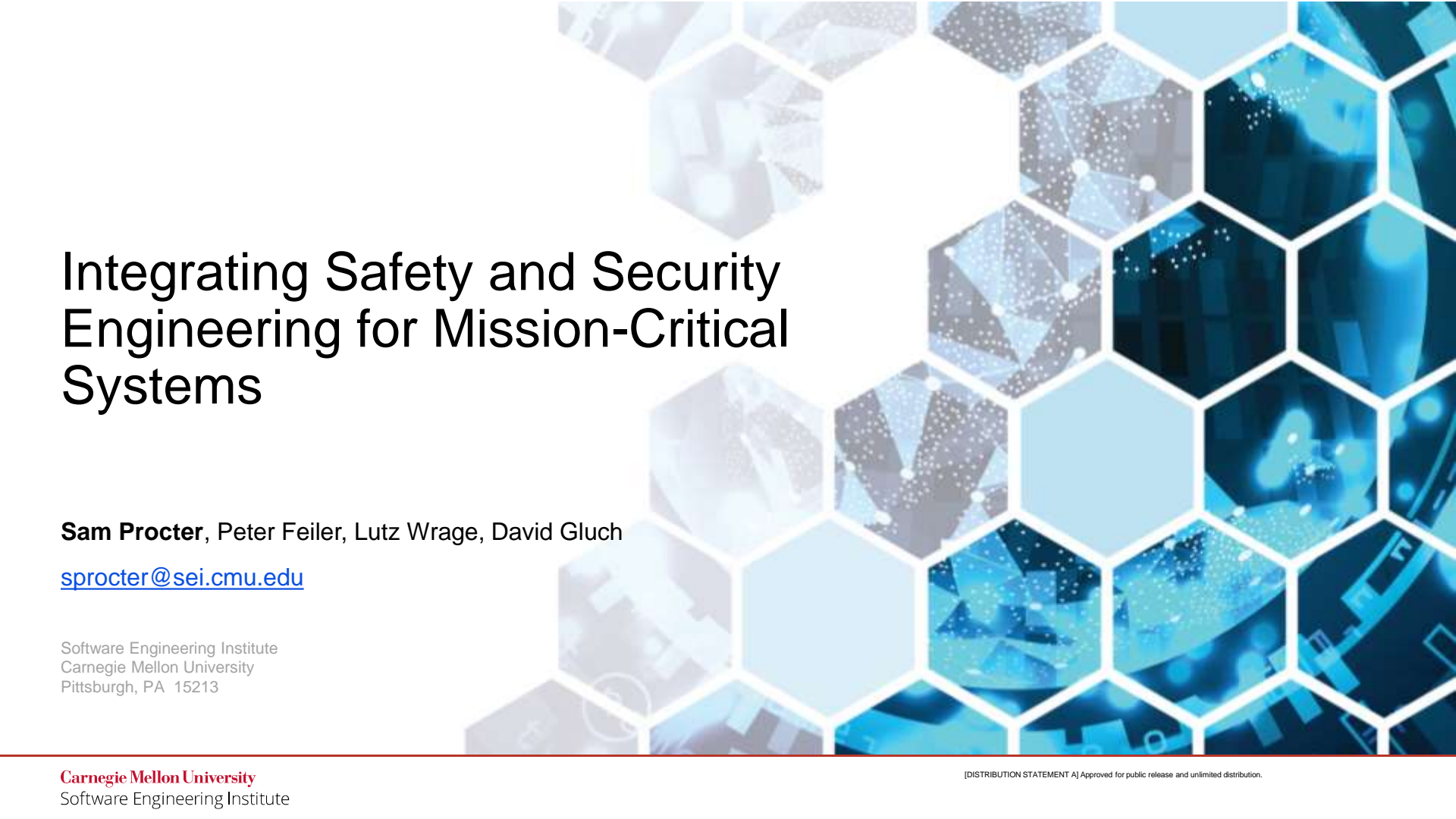
Demonstrations of Effectiveness in use of ACVIP with AADL

Finding Problems Early (AMRDEC/SEI)

- Summary: 6 Week Virtual Integration on CH47 using AADL
- Result: Identified 20 major integration issues early
- Benefit: Avoided 12-month delay on 24-month program



CH47 Chinook



Integrating Safety and Security Engineering for Mission-Critical Systems

Sam Procter, Peter Feiler, Lutz Wrage, David Gluch

sprocter@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213