Incremental Verification and Validation of System Architecture for Software Reliant Systems Using AADL (Architecture Analysis & Design Language)

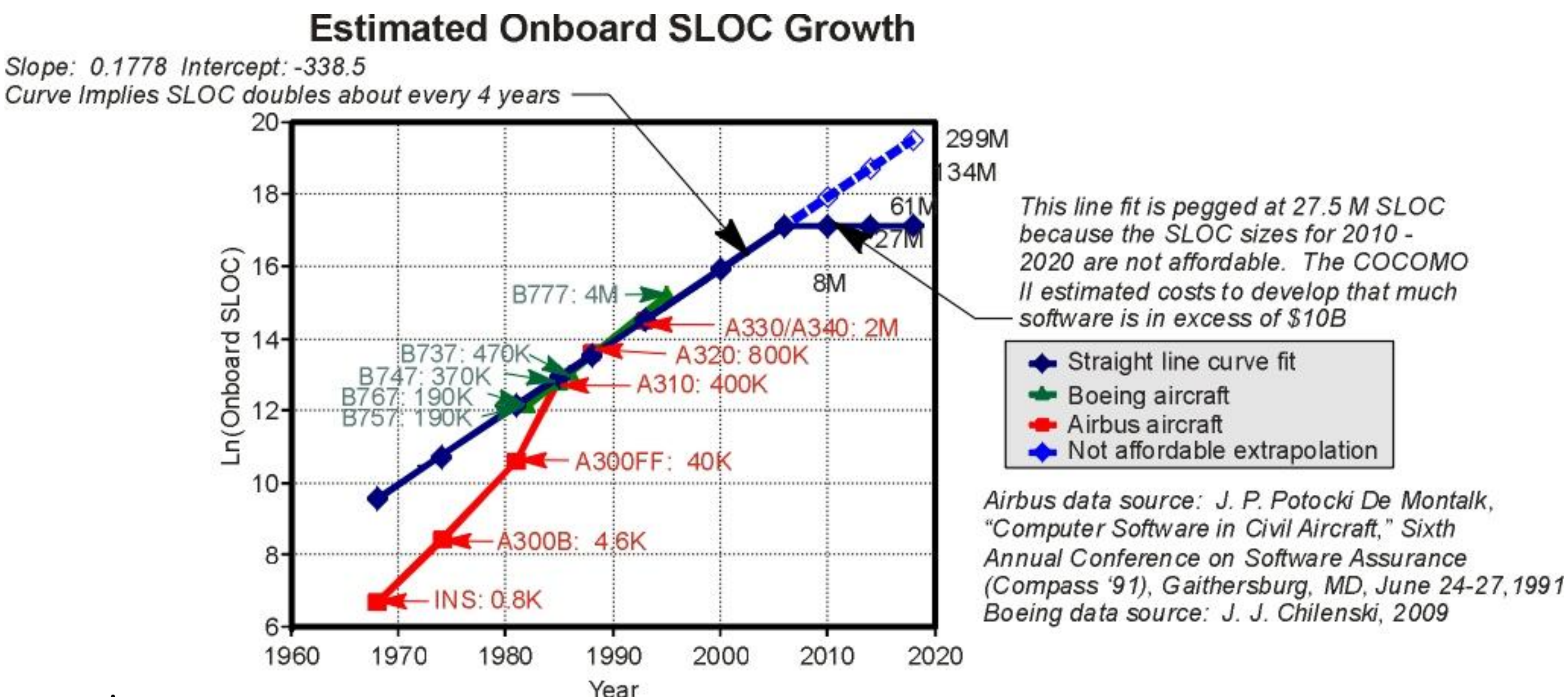
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Bruce Lewis (US Army), Peter H Feiler (SEI)

Layered Assurance Workshop, Dec 6, 2010

Gravity of the Issue Today

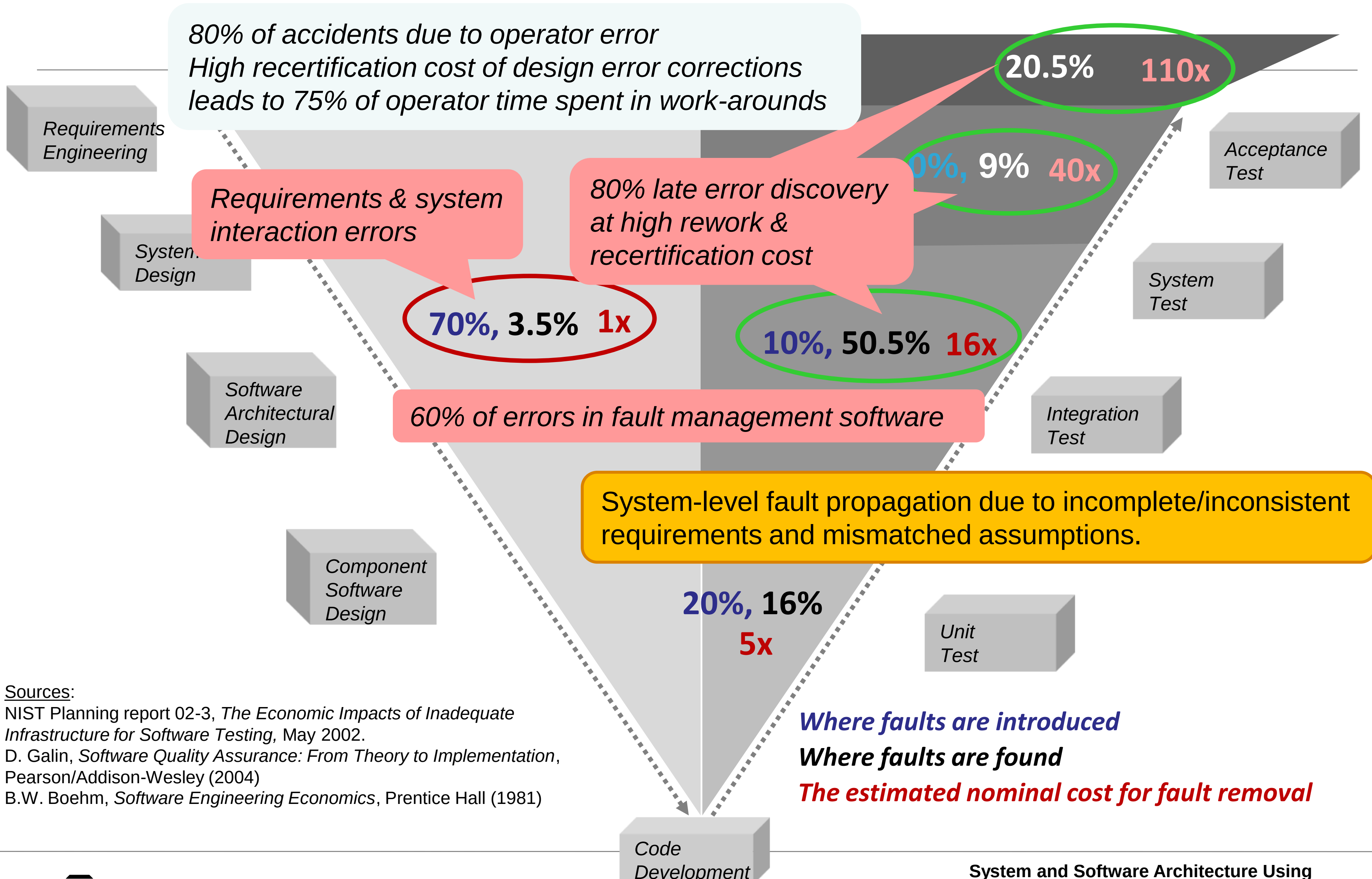
- System Complexity is Growing Rapidly…



Acronyms:

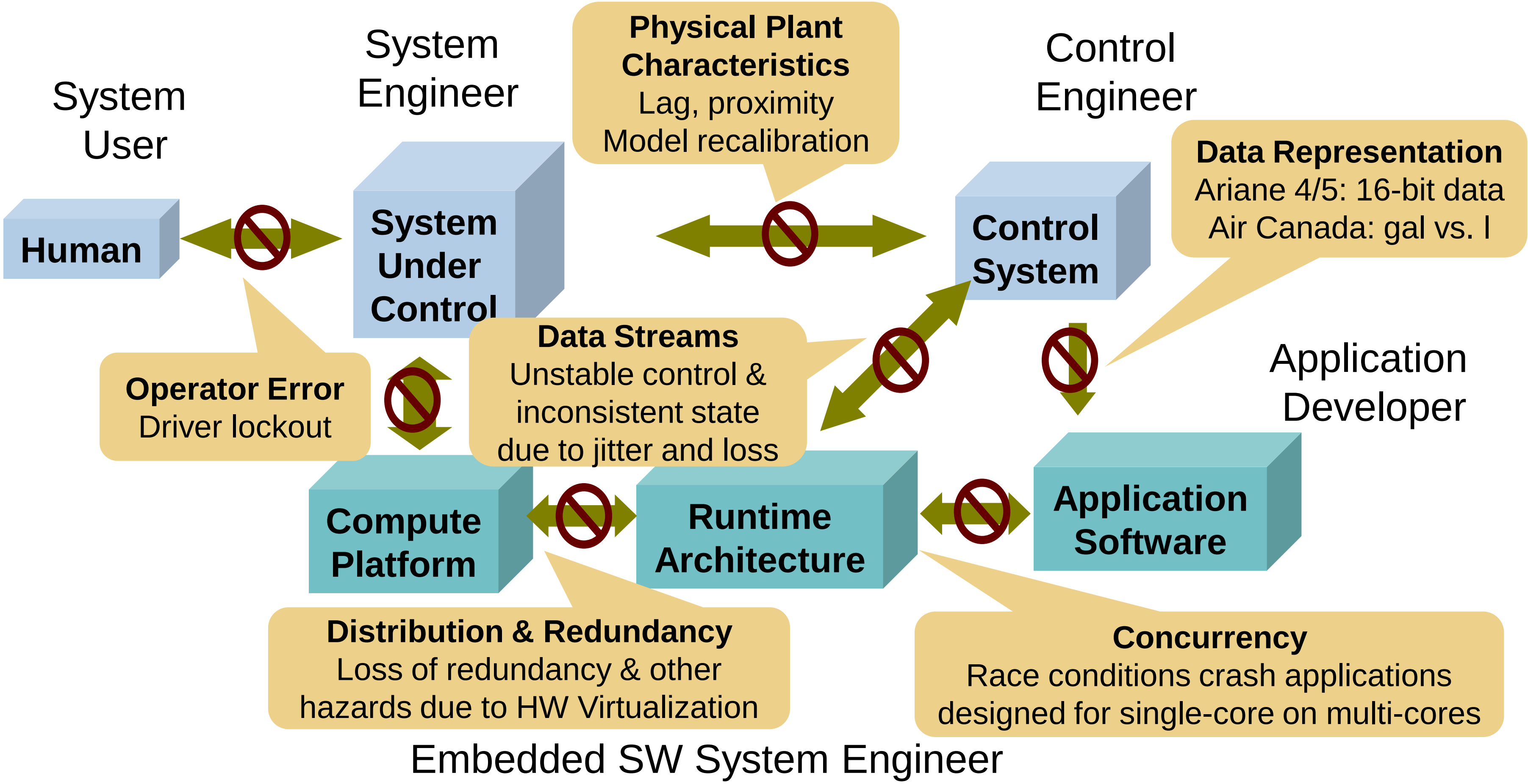
SLOC: source lines of code
COCOMO II: COnstructive COst MOdel *II*

Late Discovery of System-Level Problems



Sources:
NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.
D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)
B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)

New Levels of System Interaction Complexity & Mismatched Assumptions – AADL addresses



Software runtime system impacts safety-critical software & system properties

Fault Root Causes Due to Runtime System Architecture

AADL

Violation of data stream assumptions

- Stream miss rates, Mismatched data representation, Latency jitter & age

Data (stream) consistency
End-to-end latency analysis

Partitions as Isolation Regions

- Space, time, and bandwidth partitioning
- Isolation not guaranteed due to undocumented resource sharing
- fault containment, security levels, safety levels, distribution

Modeling of partitioned architectures

Virtualization of time & resources

- Logical vs. physical redundancy
- Time stamping of data & asynchronous systems

Fault propagation
security analysis
redundancy patterns

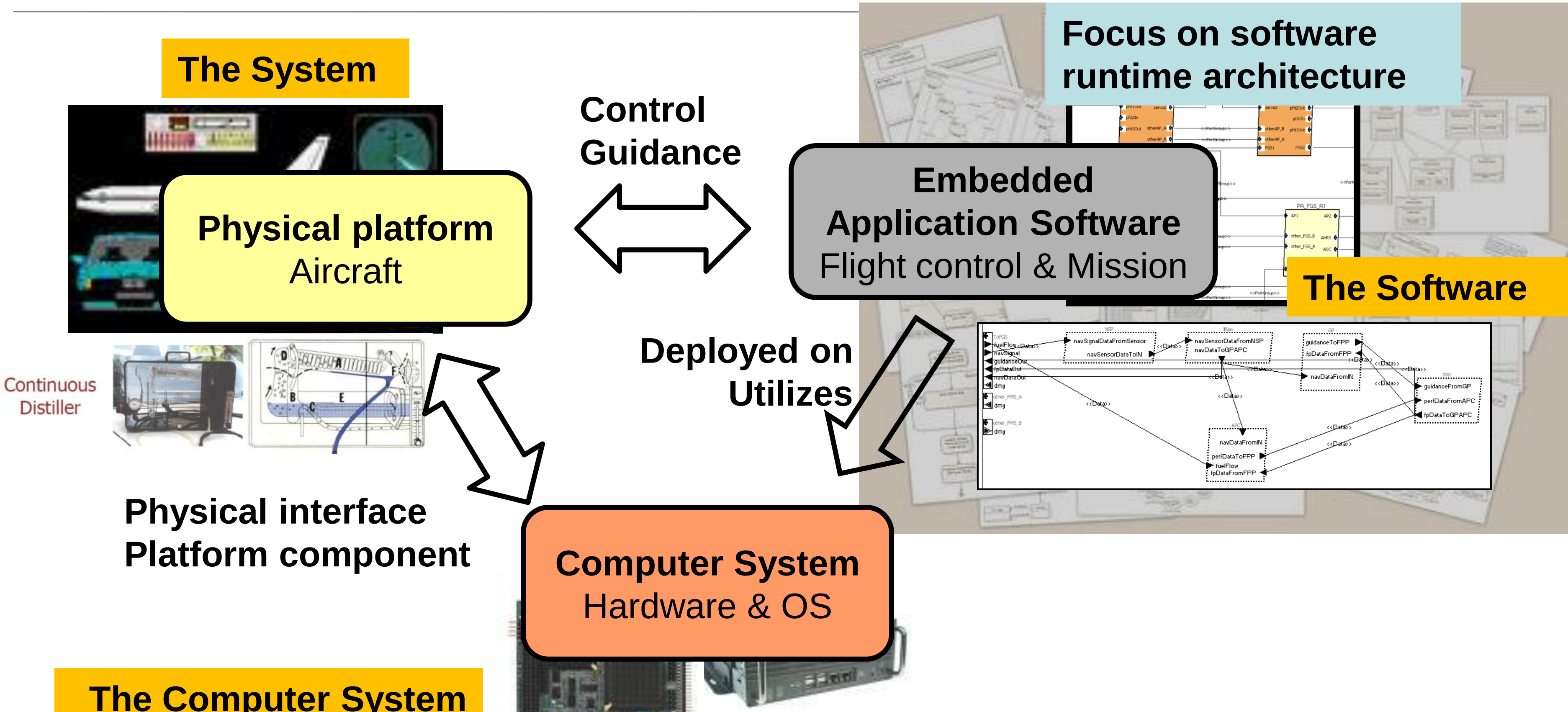
Inconsistent System States & Interactions

- Modal systems with modal components
- Concurrency & redundancy management
- Application level interaction protocols

Validation by model checking & proofs

AADL concepts capture key architecture abstractions to address root causes

SAE Architecture Analysis & Design Language (AADL) for Embedded Systems



AADL focuses on interaction between the three major elements of a software-intensive system based on architectural abstractions of each.

AADL: The Language

Designed for standardized incremental, composable, quantitative analysis and generative system integration

Precise semantics for components & interactions

- Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus, abstract
- Typed properties, properties with units and model reference values

Continuous control & event response processing

- Data and event flow, synchronous call/return, shared access
- End-to-End flow specifications, black box flow specs

Operational modes & fault tolerant configurations

- Modes & mode transition, mode specific properties & configurations

Modeling of large-scale systems

- Component variants, packaging of AADL models, public/private

Accommodation of diverse analysis needs

- Extension mechanism (property set, sublanguage) standardized

AADL Annex Standard Extensions

Behavior Annex (ballot passed 2010)

- Concurrency behavior
- Validation of implementation

ARNIC 653 Annex (ballot passed 2010)

- Define 653 architectural elements in AADL for analysis
- Generation of runtime & configuration file for 653-compliant O/S

Data Modeling Annex (ballot passed 2010)

- Interface with data model in other modeling notation

Code Generation Annex (in review)

- API & code patterns for different programming languages
- Original annex in 2006

Error Model Annex (in revision)

- Error behavior as architecture model annotation
- Original annex in 2006

Architecture Execution Semantics Defined – Components to SoS

Nominal & recovery

Fault handling

Resource locking

Mode switching

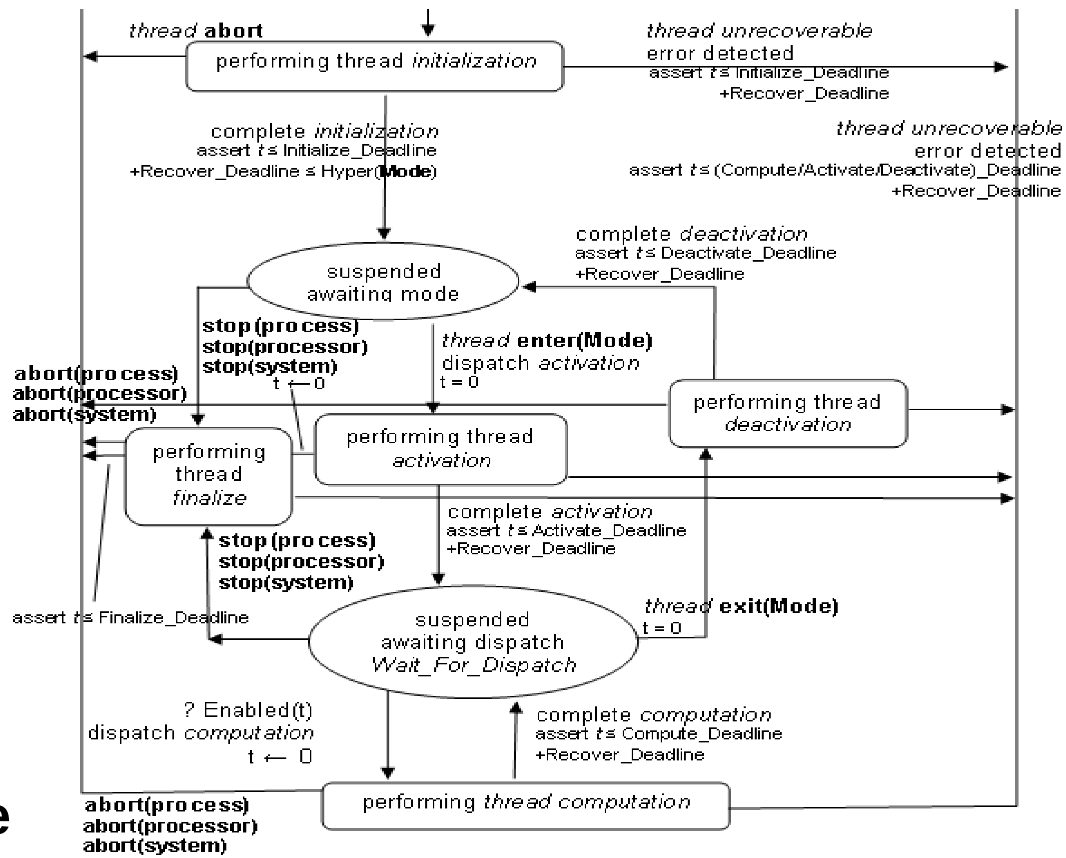
Initialization

Finalization

Temporal Logic

Modes

Behavior Annex



Thread Example Diagram

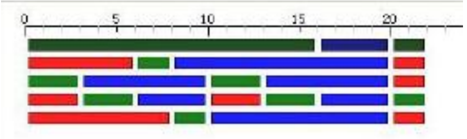
Potential Model-based Engineering Pitfalls

The Issues

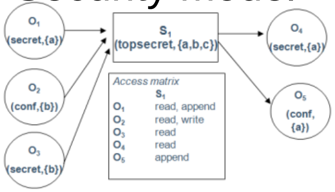
Inconsistency between independently developed analytical models

Confidence that model reflects implementation

Timing model

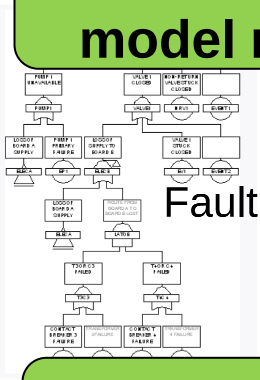


Security model



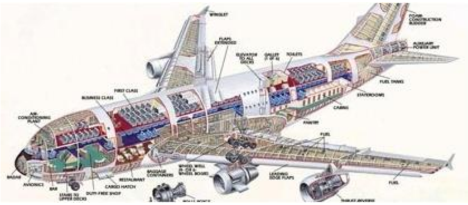
Potential Solution

Architecture-centric model repository



Fault model

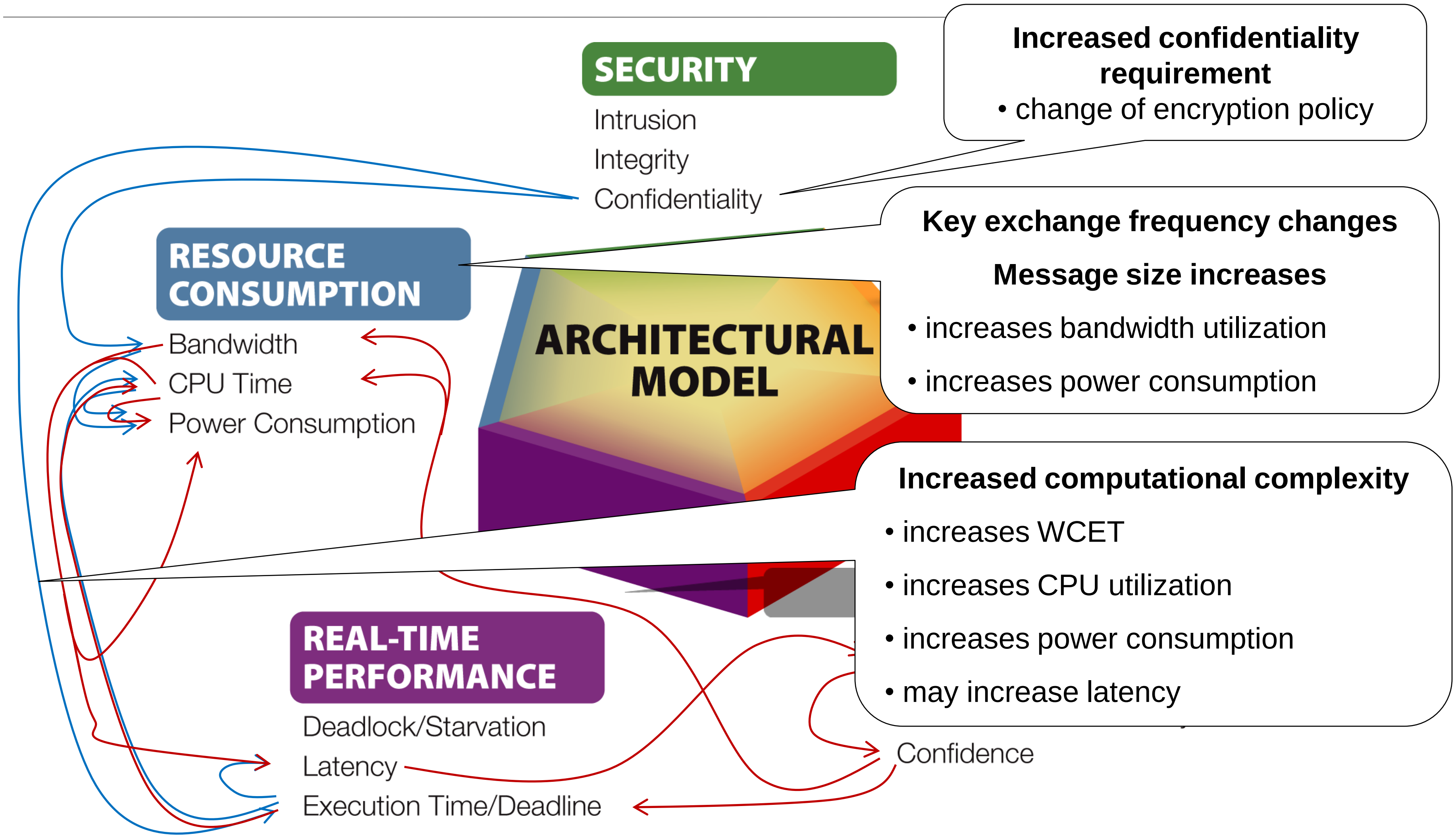
Generation from validated models



System implementation

Single-Truth through consistency across Architectural models => Architecture-Centric

AADL



Formal Methods & AADL (A bridge to formal modeling from an architecture specification).

Concurrency & mode logic: interface with Alloy (deNiz)

Simulink & AADL integration: Emmeskay & Telecom Paristech

Model checking based on Simulink specifications: Rockwell Collins

Behavioral component interaction – AADL & BIP: Verimag

Formal proofs & AADL – BLESS (pace maker): Larson

AADL & Maude Model Checking: Meseguer (UIUC), U Leicester

AADL & Timed Abstract State Machines (TASM): Lundquist

AADL & Timed Automata (Cheddar): Singhoff

AADL & Process Algebra: Sokolsky

AADL & UPPAAL: Sokolsky, Lundquist

AADL & timed Petri nets: Filali (TINA), Kordon

Consistency Across Virtual Integration Models, (Nam, Sha, deNiz)

Summary – AADL Strong Semantics

Integration of CPS effects into the architecture context

Understanding of runtime behavior and communication impact

Single truth modeling and transformation

Model compilation and Model composition

Incremental Verification and Validation

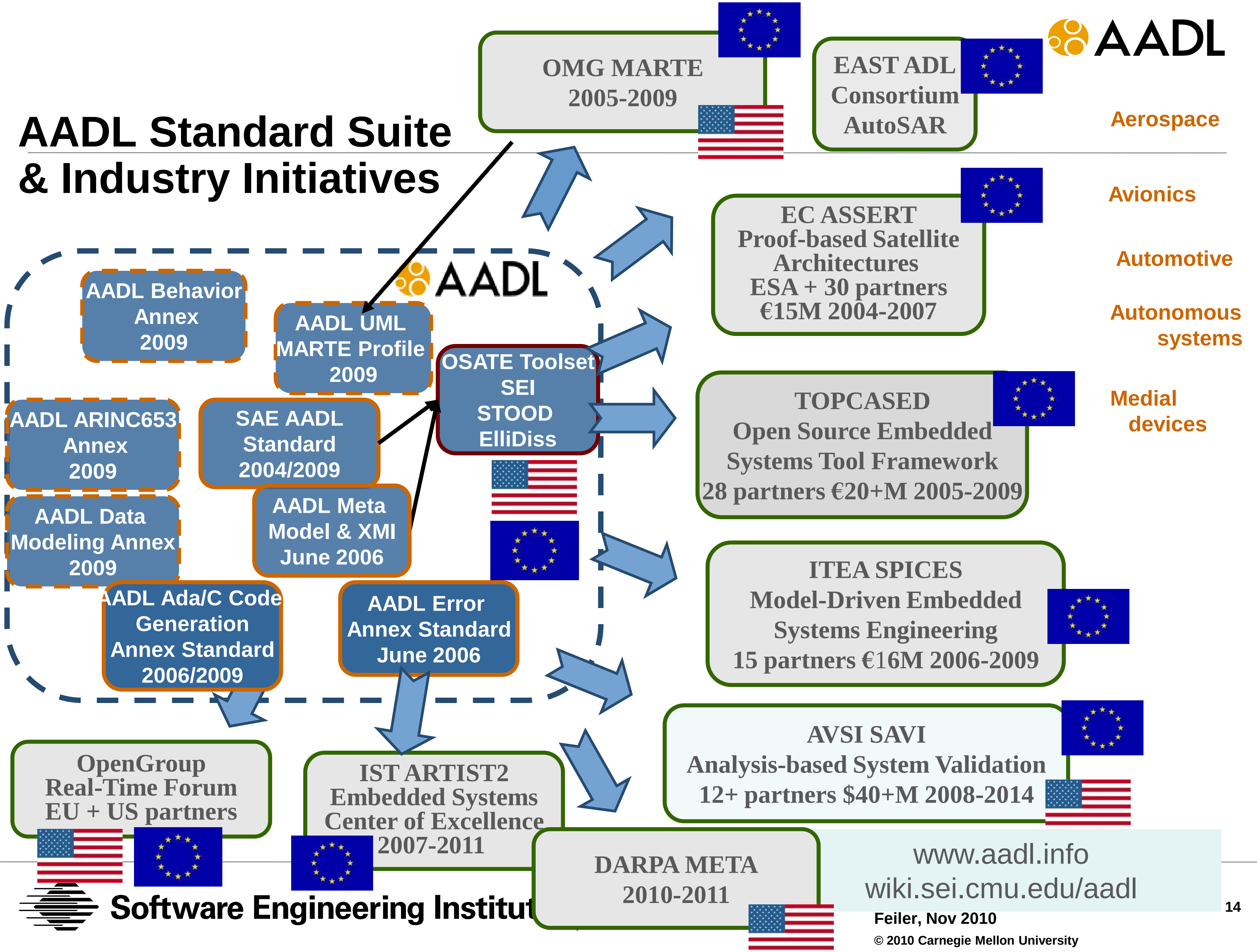
Precise, correct by construction code generation

Analysis tools per domain built to common architectural semantics

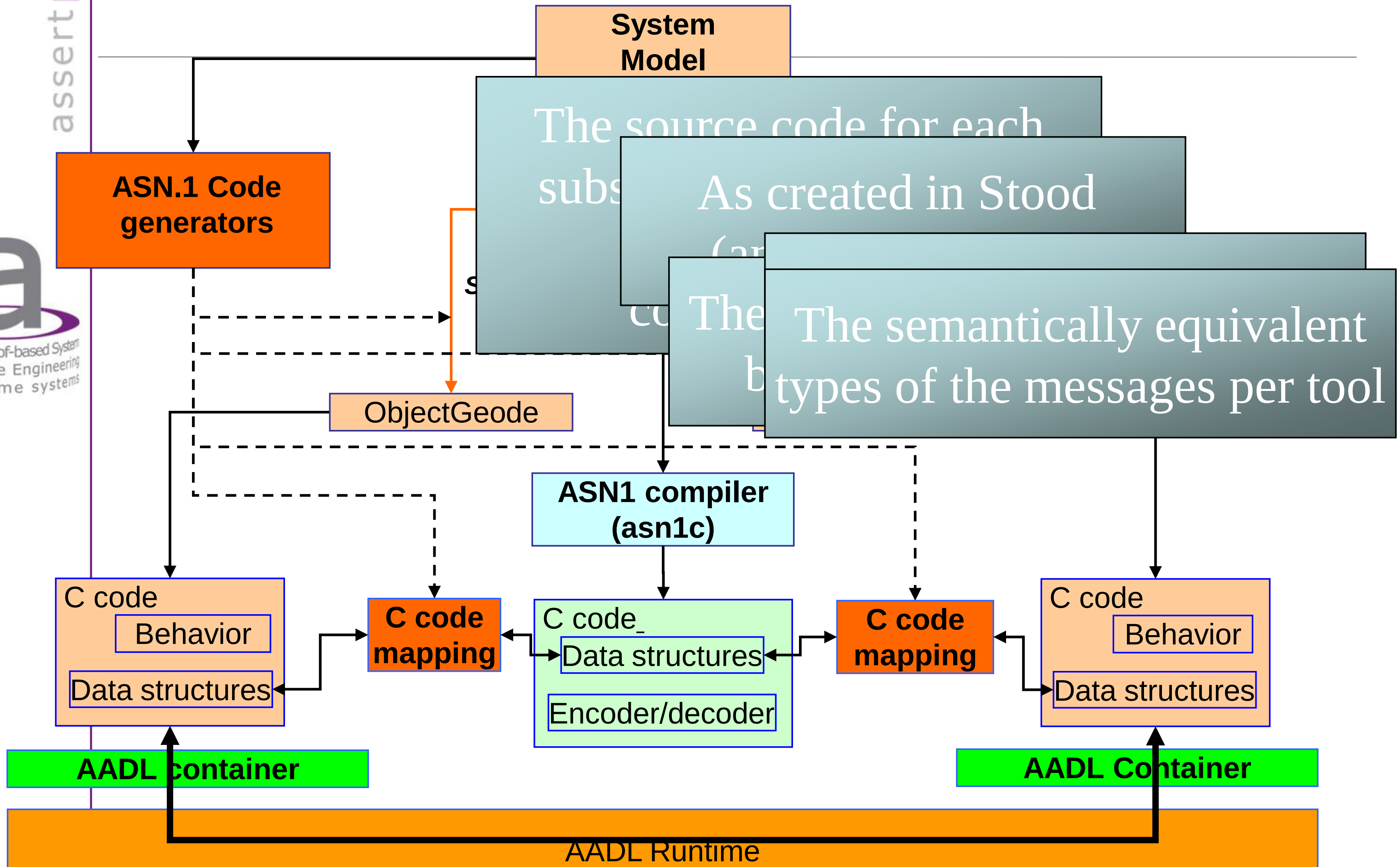
Bridge to formal analysis

Cyber-Physical adds several more dimensions of complexity – to integrate the effects into an understanding of system behavior we need a similar standardized approach, perhaps AADL annex.

AADL Standard Suite & Industry Initiatives



The ASSERT process – "Applying Model-Driven Engineering Concepts to build High-Integrity systems in the IST-ASSERT process" by Jerome Hugues



Aerospace Vehicle Systems Institute

AVSI is a global cooperative of aerospace companies, government organizations, and academic institutions

Past AVSI projects have covered the breadth of aerospace systems and current research includes projects in the areas of reliability, certification, and virtual integration.

The System Architecture Virtual Integration program is an AVSI *program* addressing virtual integration of systems.

SEI was selected as the contractor to help work the proof-of-concept Effort

How Are We to Address This Issue?

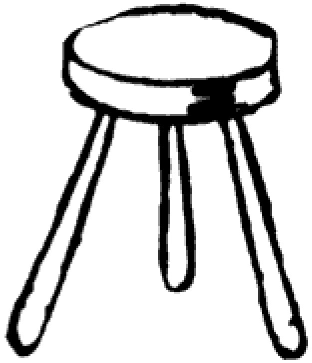
Analysis & Validation through Virtual Integration!

But what exactly does that mean?

SAVI Approach: Integrate, Then Build

SAVI is

- A changed acquisition paradigm to facilitate systems integration
- A research effort to define the standards and technologies needed to effect virtual integration
- Built on the three-legged stool of
  - ✓ *Model-based,*
  - ✓ *Proof-Based, and*
  - ✓ *Component-Based engineering*
- Structured/transformable data interfaces
- A global collaboration

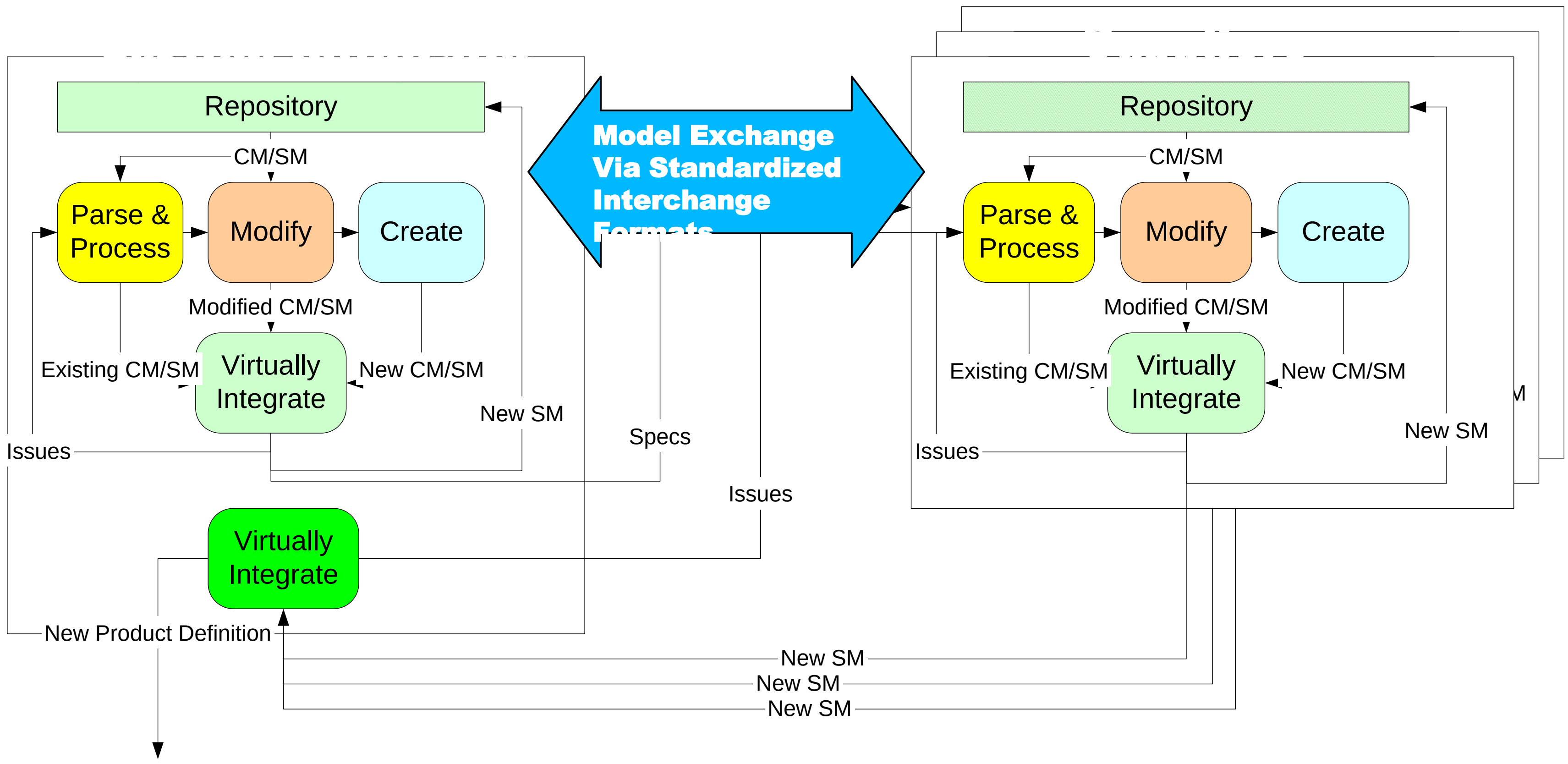


SAVI is not

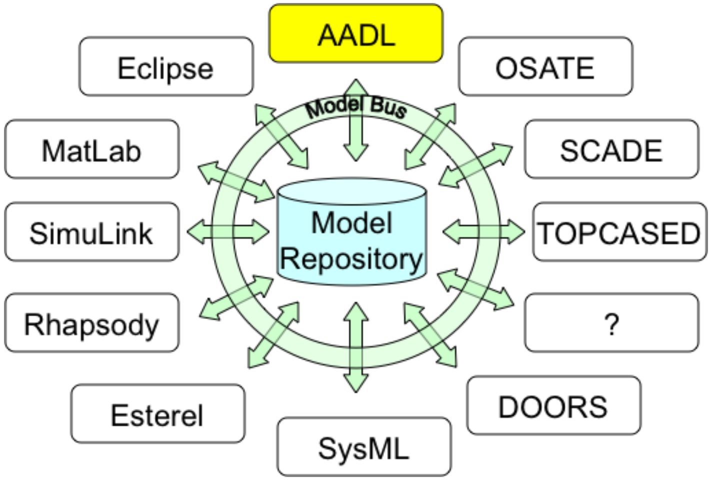
- A software tool or a design tool
- A continuation of current system development practices

Modified Business Model

System Integrator defines a new product using internal repository of virtual "parts"
Specifications for virtual subystems sent to suppliers
Proposed and developed subsystem models incrementally provided to integrator

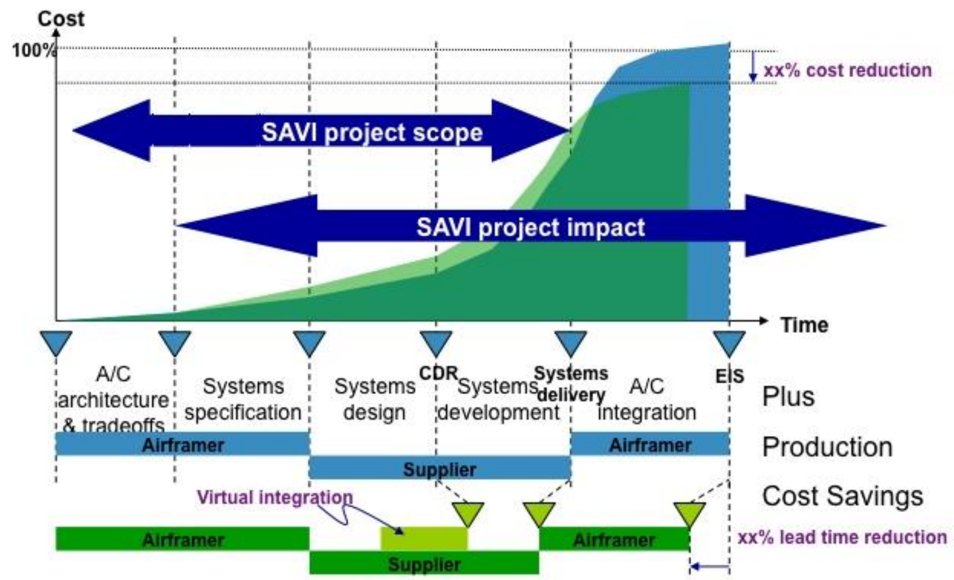


Virtual Systems Integration Uncovers Errors Earlier in Development



Standarized architecture language with strong semantics, the Model Bus and Model Repository concepts in SAVI enable…

… early validation of system and embedded software system behavior to reduce integration errors.



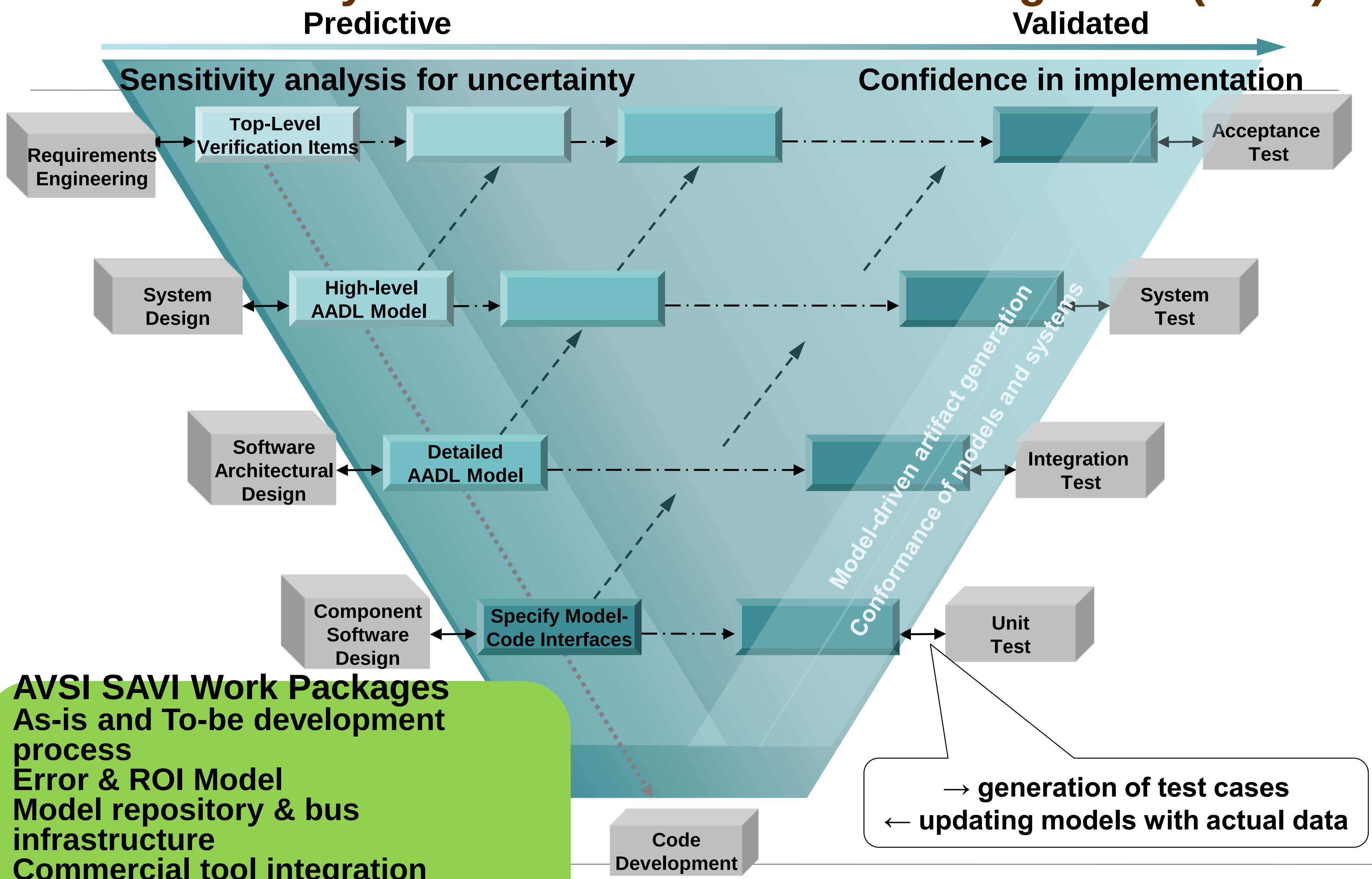
Architecture Design Language Requirements for SAVI
Supporting Embedded Software System (ESS) Analyses

Desired quality	Reason	AADL
ESS architecture concepts with precise semantics	Standardized analysis quantitative assessment Use of formal methods	ESS abstractions as language primitives Semantics well-documented for each component & interaction category
Checkable consistency of architecture formation	Incremental change impact detectable Impact analysis across quality attributes	Compilable strongly typed language with standard legality & consistency rules EMF-based meta model drives XMI standard Design & operational quality attributes
Component-based fidelity multi-dimensional modeling	Consistency & quantitative analysis early & throughout development life cycle	Hierarchical composable SW/HW/physical components with interaction behavior & timing Explicit support for templates, patterns, incomplete models Standard extensions via property sets & annex sublanguages to core
Model scalability, variability & management	Large scale system modeling & subcontractor management	Spec/instance separation Type/implementation variation Dynamic re-configurability Packages to manage model space

PoC Prioritized Requirements

#	Requirement	Category
1	Establish Model Bus infrastructure	Process
2	Establish Model Repository Infrastructure	Process
3	Inform RoI estimates through POC performance & results	Process
4	Analyses be conducted across the system	Analysis
5	Two or more analyses must be conducted	Analysis
6	Analyses be conducted at multiple levels of abstraction	Analysis
7	Analyses must validate system model consistency at multiple levels of abstraction	Analysis
8	Analyses must be conducted at the highest system level abstraction	Analysis
9	Model infrastructure must contain multiple model representations	Model
10	Model infrastructure must contain multiple communicating components	Model

Benefits of System Architecture Virtual Integration (SAVI)

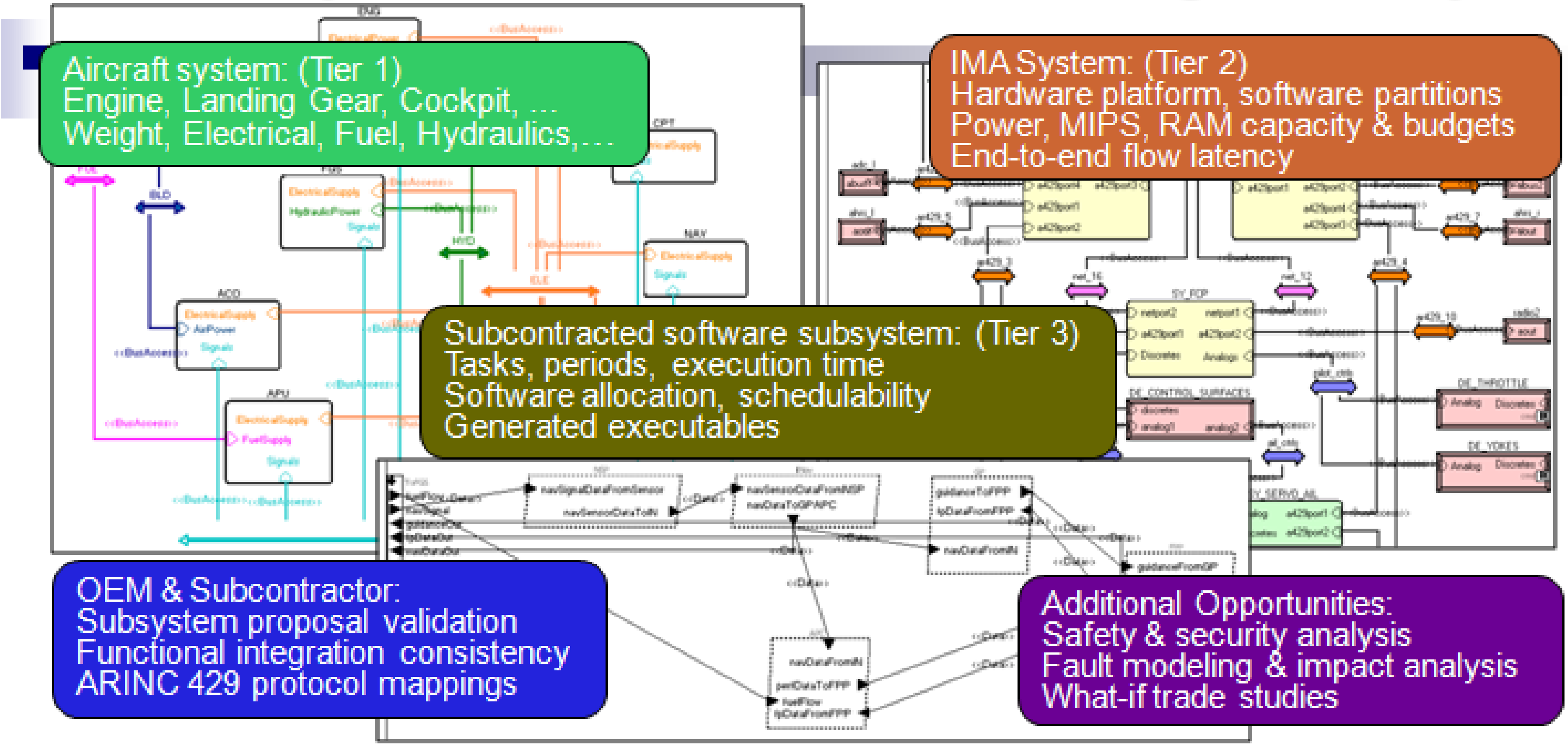


AVSI SAVI Work Packages

- As-is and To-be development process
- Error & ROI Model
- Model repository & bus infrastructure
- Commercial tool integration
- Standards based process
- Acquisition
- Certification

SAVI Proof Of Concept Demo

Incremental Multi-Fidelity Multi-dimensional Multi-Layered Architecture Modeling & Analysis



- System & software system
- Integrator & subcontractor virtual integration

Proof-of-Concept Demonstration - (4/4)

- Did the results from this PoC Demonstration indicate that the System Architecture Virtual Integration (SAVI) methodology is technically feasible to pursue?

UNANIMOUS -- YES!

- Core concepts were demonstrated on three different models, BUT…
  - **Scalability was not fully explored**
  - **Open issues with AADL (ADL used in PoC) are to be explored**
    - ***Meets needs of all Use Cases?***
    - ***Full compatibility with DoDAF version 2?***

Cost Reduction through Rework Avoidance

Based on research investment not cost to apply

$$ROI = \frac{NPV\ (\text{Cost avoidance with SAVI discounted at } 10\%)}{NPV\ (\text{Cost to develop SAVI discounted at } 10\%) * \text{Years}}$$

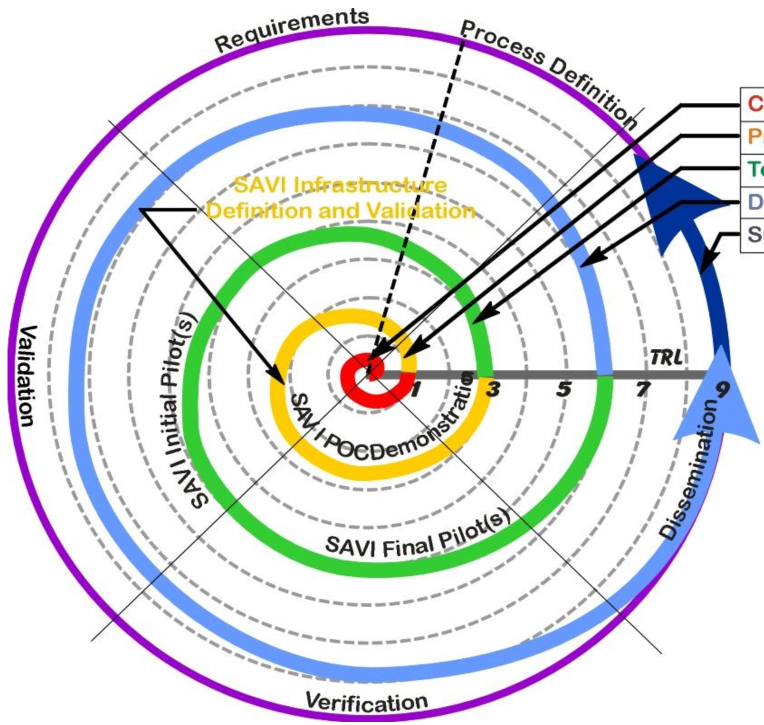
Cost reduction ranges from $717M (7.8%) to $2,391M (26.1%) on a $9,176M new airplane project (2014-2018)

Every increase of 1% in defect removal efficiency results in a conservative cost reduction of $22M
Estimates based on conservative assumptions

- Based on industry data from SAVI participants
- Model assumes development of a single large aircraft in the 2014-2018 timeframe
- Savings largely driven by reduction of rework via discovery of requirements related problems earlier in the development lifecycle
- ROI does not include savings in maintenance & field upgrades, schedule overrun, loss of life & equipment, mission delay
- Conservative – used research investment of $108M, 2.5x expected, 2010-2014.

Spiral Development Planned

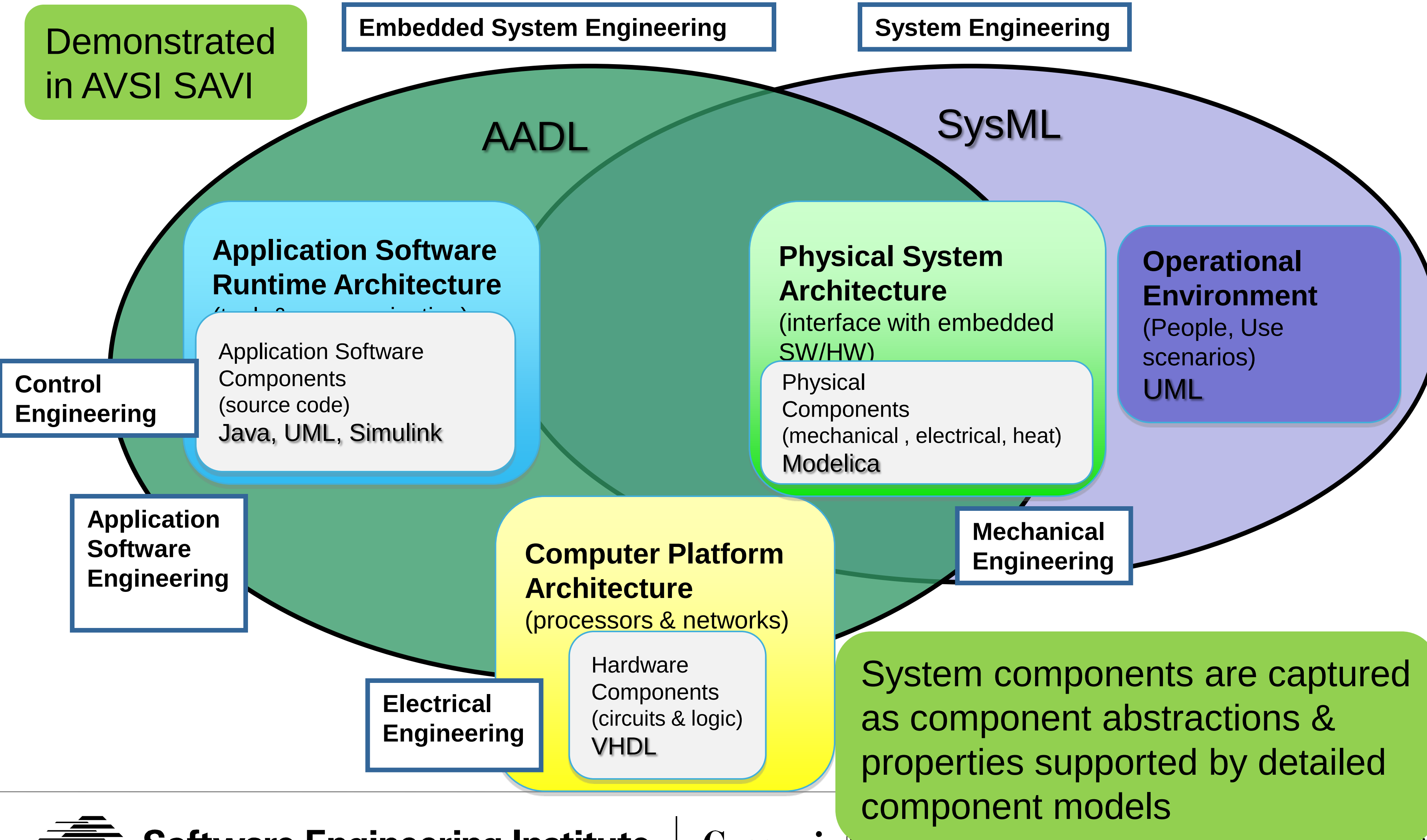
- ❑ Three Iterations to Reach TRL 9



	SAVI
Concept Loop	
Proof of Concept Loop	AFE#58
Technology Demonstration Loop	AFEs#60-67
Development "Productionization" Loop	
Sustainment Loop(s)	

- ❑ Schedule Roadmap Next

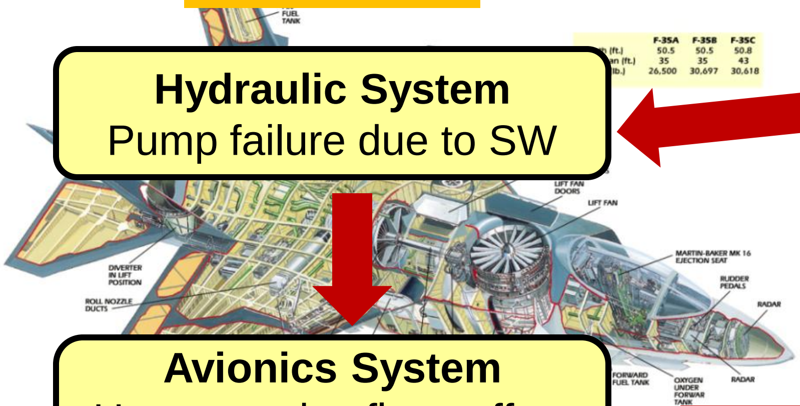
Cooperative Engineering of Systems: A Multi-notation Single Source Repository Approach



A Fault Propagation Use Case
System & Embedded Software Loop

AADL

The System

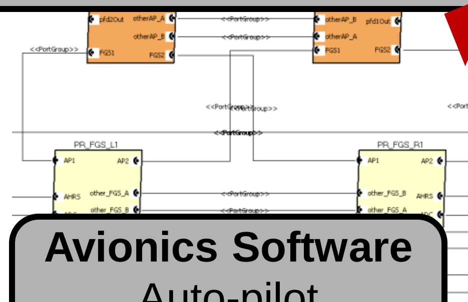


Hydraulic System
Pump failure due to SW

Avionics System
Unresponsive flaps affect flight control

The Software

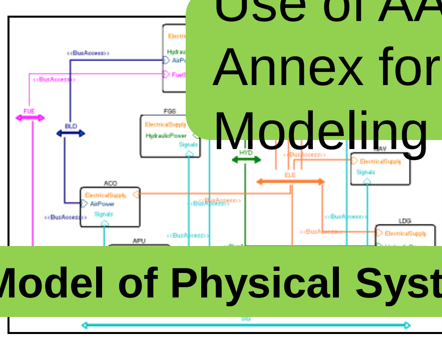
Hydraulic Control Software
Pump control



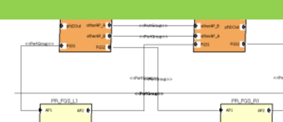
Avionics Software
Auto-pilot
Flight control

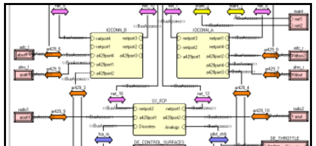
Computer System

Computer System
Bus failure



Use of AADL Error Model Annex for Fault Propagation Modeling

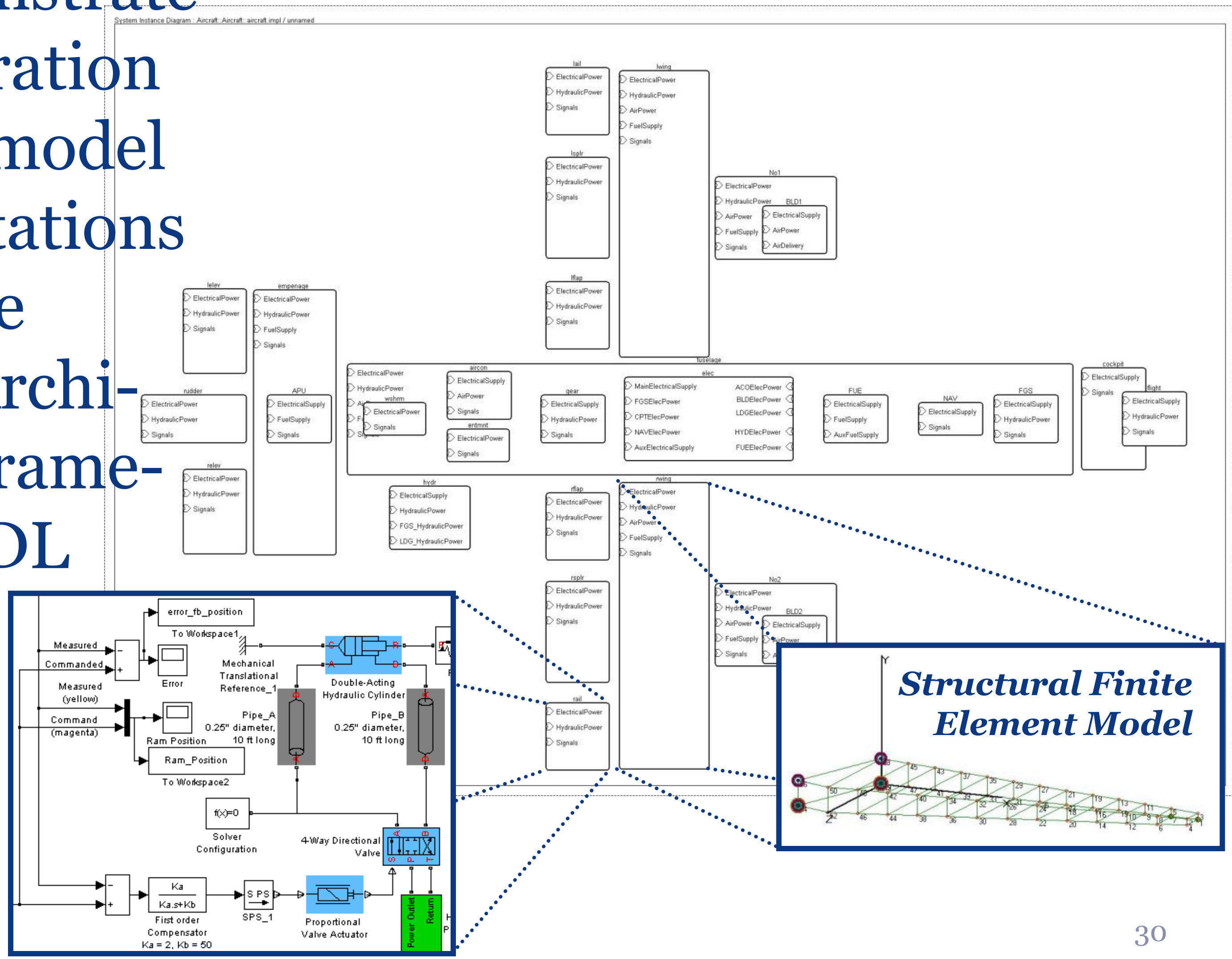




AADL Model of Physical System, Application Software, Computer System

Behavior Analysis Demo. – Aims (4)

4. To demonstrate the integration of multi-model representations within the EPoCD Archi-tecture Frame-work AADL model.



Mechatronic Actuator Model

AADL and Safety-Criticality

Fault management

- Architecture patterns in AADL
  - Redundancy, health monitoring, …
- Fault tolerant configurations & modes

Dependability

- Error Model Annex to AADL
- Specification of fault occurrence and fault propagation information
- Use for hazard and fault effect modeling
- Reliability & fault tree analysis

Behavior validation

- Behavior Annex to AADL
- Model checking
- Source code validation

```
package errormodels
public
  annex error_model {**
    -- simple error model
error model Basic
features
    Failed : error event;

    Error_Free: initial error state;
    Permanent_Failure: error state;

    Visible_Failure: in out error propagation;
end Basic;

error model implementation Basic.Nominal
transitions
     Error_Free -[Failed, in Visible_Failure]-> Permanent_Failure;
     Permanent_Failure -[out Visible_Failure]-> Permanent_Failure;
properties
     Occurrence => poisson 10E-4 applies to Failed;
     Occurrence => poisson 10E-6 applies to Visible_Failure;
end Basic.Nominal;
```

Consistency checking of safety-criticality levels

AADL Error Annex

AADL annex that supports various forms of reliability and safety analysis

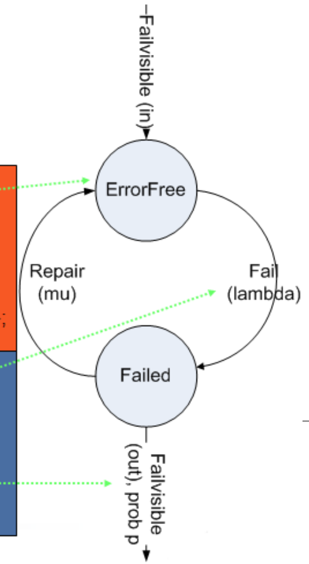
Defines error model

- State transition diagram that represents normal and failed states
- Error models can be associated with hardware components, software components, connections, and “system” (composite) components

Error model consists of

- State definitions
- Propagations from and to other components
- Probability distribution and parameter definitions
- Allowed state transitions and probabilities

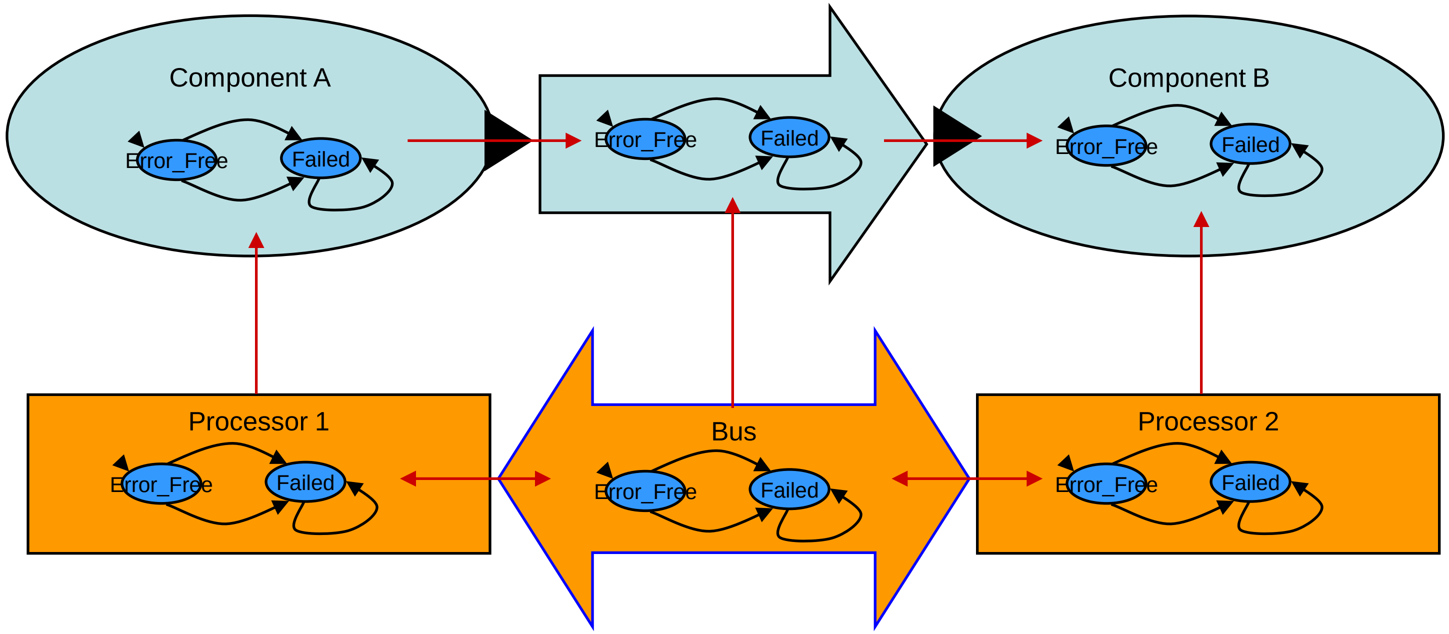
```
error model example
features
ErrorFree: initial error state;
Failed: error state;
Fail: error event {Occurrence => poisson lambda};
Repair: error event {Occurrence => poisson mu};
Failvisible: in out error propagation {Occurrence => fixed p};
end example;
error model implementation example.general
transitions
ErrorFree-[Fail]->Failed;
Failed-[Repair]->ErrorFree;
ErrorFree-[in Failvisible]->Failed;
Failed-[out Failvisible]->Failed;
end example.general;
```



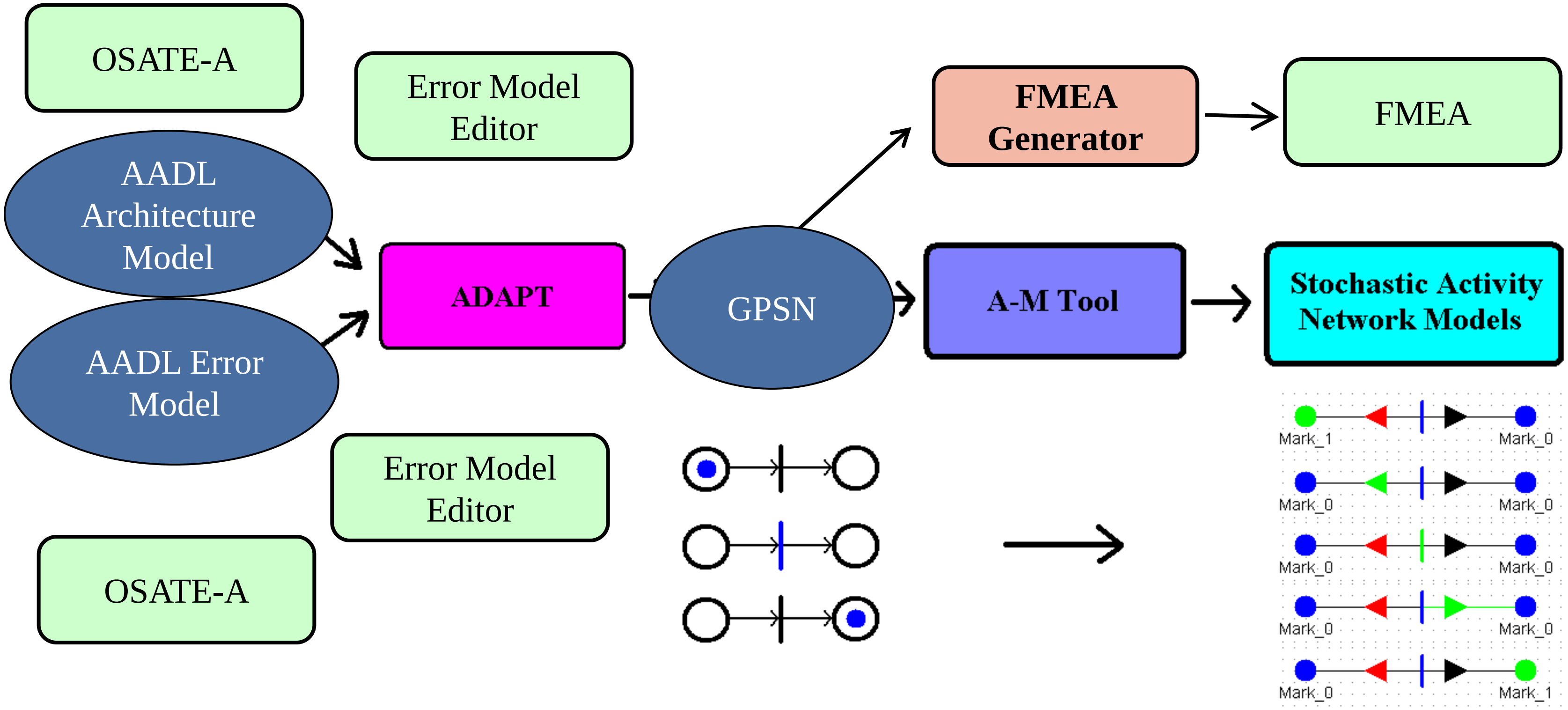
Leverage Connectivity in AADL Models

Fault propagation at the application logic level, at the hardware level, and between the two levels.

- Provides compositional model specification approach
- Architecture defines propagation paths for software and hardware

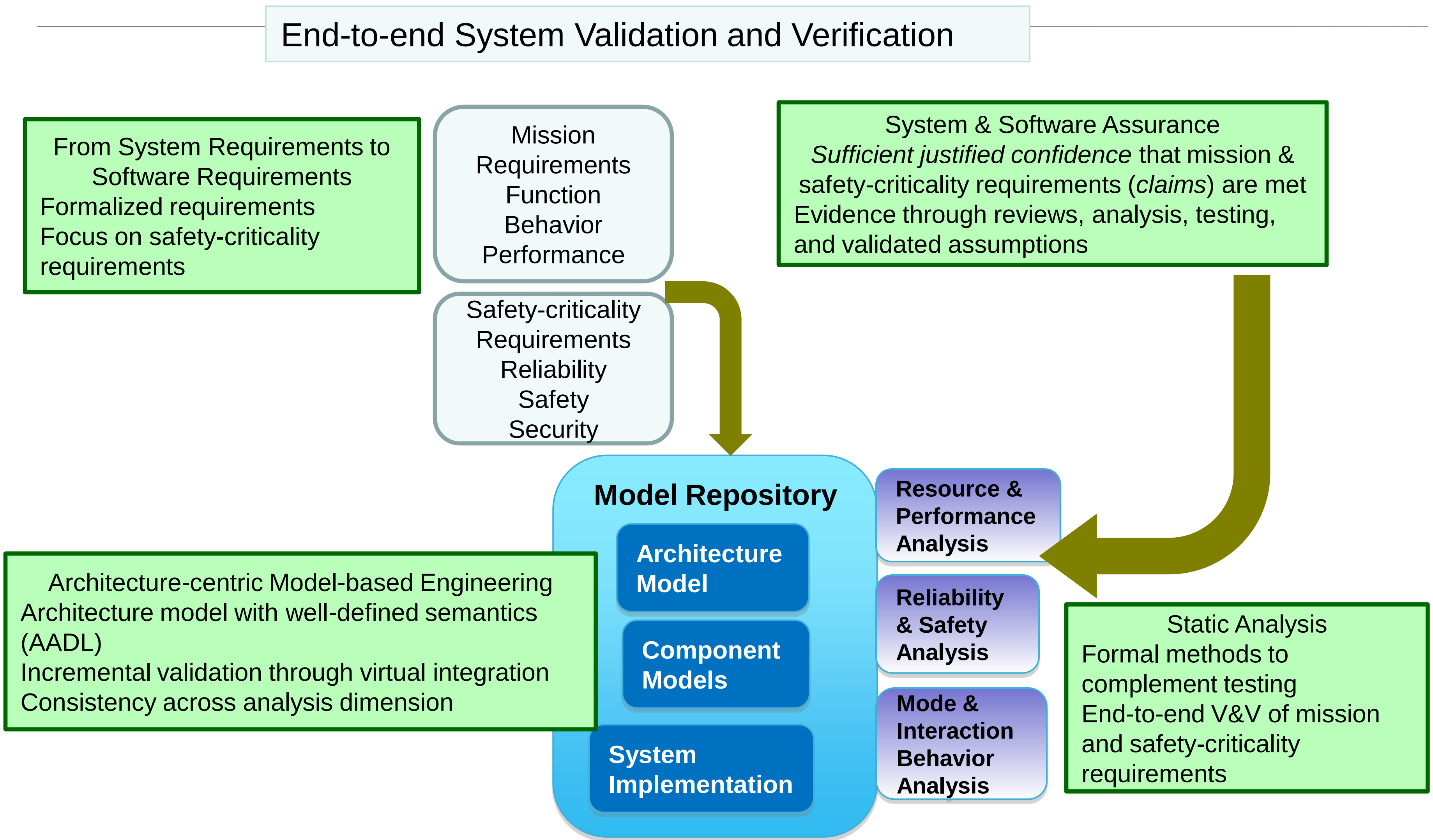


AADL transformation

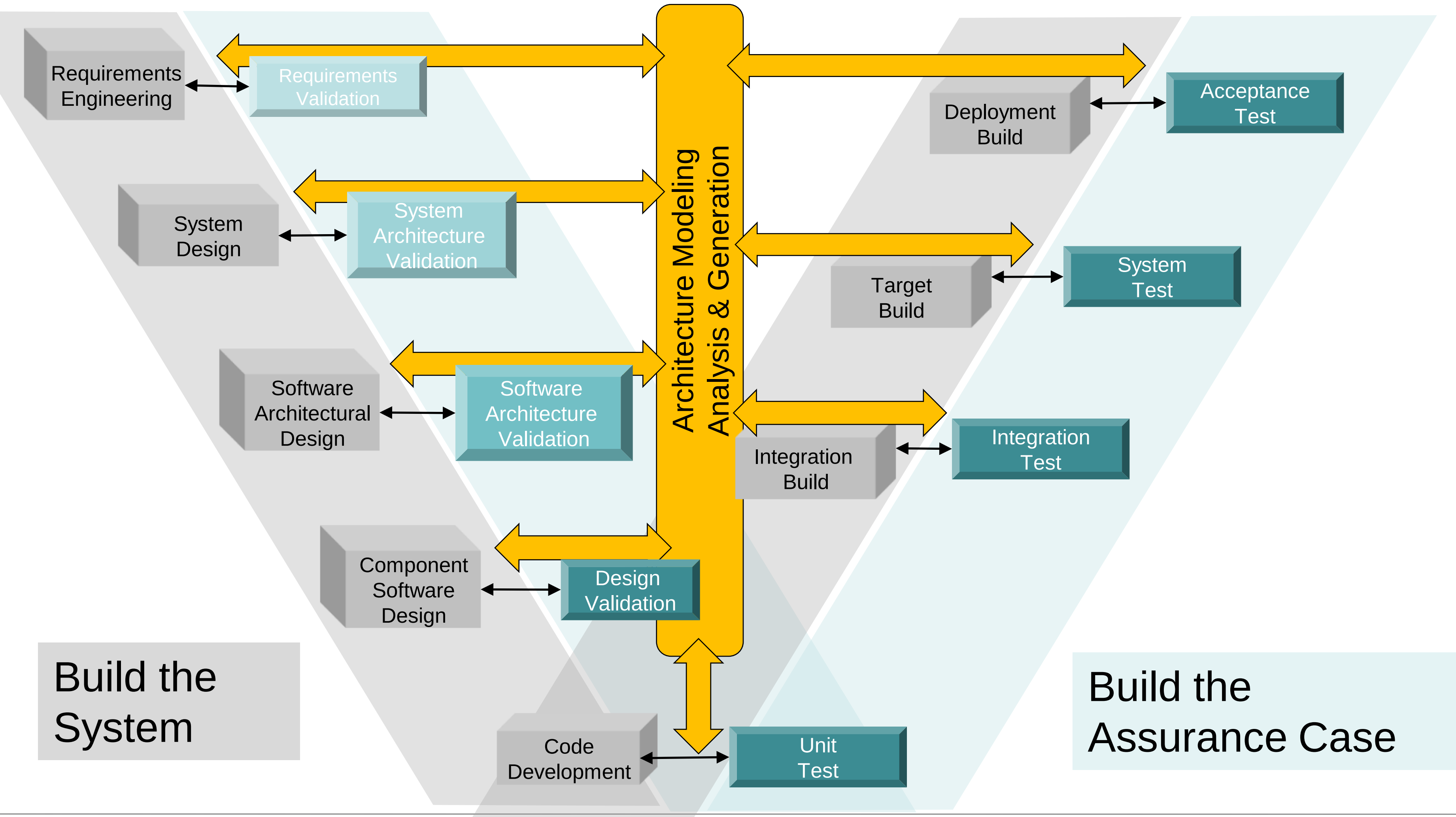


- ADAPT Tool (Ana Rugina, LAAS-CNRS)
  - *Packaged as an eclipse plug-in*
  - *Takes in AADL architecture and error behavior information*
  - *Converts to a general stochastic petri net*
  - *Outputs GSPN information to an XML file*
- ADAPT-MOBIUS Converter
  - *Takes in the ADAPT XML file.*
  - *Converts a GSPN to a Mobius Stochastic Activity Network*
  - *Outputs SAN information to an XML format.*

Reliability Validation & Improvement Framework



Incremental Architecture-centric Validation & Verification Improves Qualification Confidence



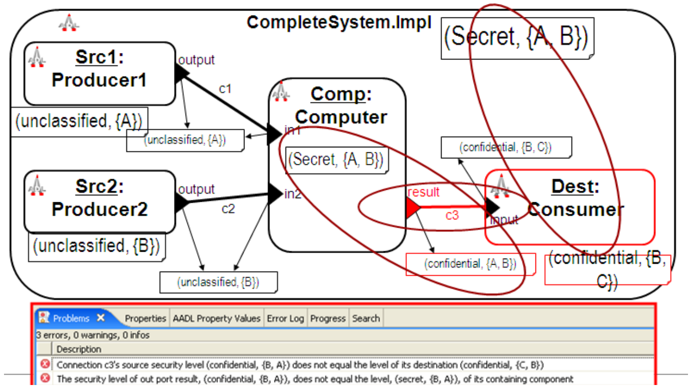
AADL: Security Modeling

Confidentiality concerns that sensitive data should only be disclosed to or accessed/modified by authorized users, i.e., enforcing prevention of unauthorized disclosure of information.

Objective: Model security attributes for an architecture to verify that data is properly accessed and handled by users and applications.

Confidentiality frameworks

- Bell-LaPadula framework: military applications
- Chinese wall framework: commercial applications
- Access role/role-based access framework
- MILS



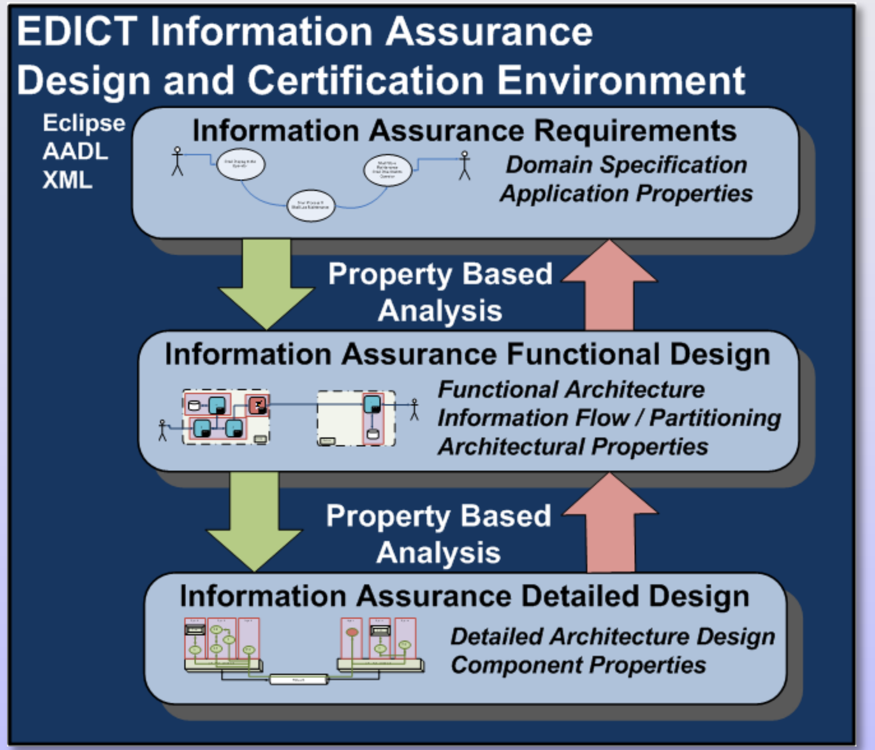
.

Model Based Analysis for Information Assurance

EDICT IA Vision

An integrated tool suite for the *specification*, *design*, *evaluation* and *deployment* of high confidence systems

- An innovative approach for
  - IA domain specific modeling and systematic evaluation and analysis
  - Integration with standard development and certification processes
- Utilize a Model Driven development approach to support the specification and evaluation of system properties *throughout the system lifecycle*
- *Support for modeling and analysis of MILS design approaches*
- Provide views and tools that are tuned to the needs of system stakeholders cross cutting concerns and activities
  - Architects – Security Engineers - Certifiers
- Utilize analysis after system deployment to support
  - Upgrades – Changes In Threat – Changes In Operations



Open Modeling and Tool Platform

- Eclipse Platform for tool portability and open integration
- AADL for system architecture modeling
- XML based information storage

Architecture-Centric Virtual Integration Impact

- Reduce the risks
  - Analyze system early and throughout life cycle
  - Understand system wide impact
  - Validate assumptions across system
- Increase the confidence
  - Validate models to complement integration testing
  - Validate model assumptions in operational system
  - Evolve system models in increasing fidelity
- Reduce the cost
  - Fewer system integration problems (SAVI ROI)
  - Fewer validation steps through use of validated generators

Back-Up

The SAVI demo video can be watched over the web at

www.aadl.info/aadl/savi/2009POCDemo/avsisaviPOCDemo35min.html

Design, Verification and Implementation of MILS Systems

Julien DELANGE, Laurent PAUTET

TELECOM ParisTech -- delange@enst.fr, pautet@enst.fr

Fabrice KORDON

LIP6, Univ. P & M. Curie -- fabrice.kordon@lip6r