Architecture-led Incremental System Assurance (ALISA) Demonstration

Peter Feiler
Feb 2017

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Assurance & Qualification Improvement Strategy

Assurance: Sufficient evidence that a system implementation meets system requirements

|-------------------------------------------|-----------------------------------------------|---------------------------------------------|----------------------------------------------------------|

- **Mission Requirements**
  - Function
  - Behavior
  - Performance

- **Survivability Requirements**
  - Reliability
  - Safety
  - Security

**Model Repository**
- Architecture Model
- Component Models
- System Implementation
- System configuration

**Operational & Failure Modes**
- Resource, Timing & Performance Analysis
- Reliability, Safety, Security Analysis

**Architecture-centric Virtual System Integration (ACVIP)**
Incremental Lifecycle Assurance (ALISA)
Value of Requirement Uncertainty Awareness

Textual requirement quality statistics

- Current requirement engineering practice relies on stakeholders traceability and document reviews resulting in high rate of requirement change.

Managed awareness of requirement uncertainty reduces requirement changes by 50%

- 80% of requirement changes from development team
- Expert requirement uncertainty assessment
  - Volatility, Impact, Precedence, Time criticality
- Focus on high uncertainty areas
- Engineer for inherent variability

<table>
<thead>
<tr>
<th>Requirements error</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete</td>
<td>21%</td>
</tr>
<tr>
<td>Missing</td>
<td>33%</td>
</tr>
<tr>
<td>Incorrect</td>
<td>24%</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>6%</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>5%</td>
</tr>
</tbody>
</table>

NIST Study

<table>
<thead>
<tr>
<th>Selection Precedence</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Precedence</td>
<td>9</td>
</tr>
<tr>
<td>Medium Precedence</td>
<td>3</td>
</tr>
<tr>
<td>High Precedence</td>
<td>1</td>
</tr>
</tbody>
</table>

No experience of concept, or environment. Historically volatile

Some experience in related environments. Some historic volatility

Concept already in service. Low historic volatility

Figure 8. Precedence measurement scale

Rolls Royce Study
Three Dimensions of Requirement Coverage

Guarantees
Assumptions

Invariants
Implementation constraints
Exceptional conditions

System interactions, state, behavior

Constraints/Controls
System
Behavior
State
Resources

Environment

Design & operational quality attributes

Performance
Transaction Throughput
Modifiability
Change COTS
Utility
Availability
Security

Data latency on customer DB to < 200 ms
Deliver video in real time.
Add CORBA middleware in < 20 person-months.
Change Web user interface in < 4 person-weeks.
Power outage at site 1 requires traffic redirected to site 2 in < 3 seconds.
Network failure detected and recovered in < 1.5 minutes.
Credit card transactions are secure 99.999% of the time.
Customer DB authorization works 99.999% of the time.

Fault Propagation Ontology

Omission errors
Commission errors
Value errors
Sequence errors
Timing errors
Replication errors
Rate errors
Concurrency errors
Authentication errors
Authorization errors

Fault impact & contributors

System Under Control
Behavior
State
Actuator
Input
Output
Sensor

Control System
Behavior
State
Output
Input

Architecture-led Incremental System Assurance
Feb 2017
© 2015 Carnegie Mellon University
Three Dimensions of Incremental Assurance

Incremental assurance through virtual system integration for early discovery
Return on Investment study by SAVI*

Priority focused architecture design exploration for high payoff
Measurable improvement (Rolls Royce)

Compositional verification and partitions to limit assurance impact

*System Architecture Virtual Integration (SAVI) Aerospace industry initiative
Modeling Notations in ALISA Prototype

ReqSpec: Represent stakeholder goals, system requirements
- Document-based & architecture-led
- Verifiable system requirements
- Coverage and uncertainty

Verify: Verification plans of verification activities against artifacts
- Reasoning logic of how verification activities satisfy requirement
- Via verification methods (manual, automated) on models/code
- Assumptions, preconditions on verification methods

Alisa: Composition of verification plans into assurance cases
- Verification of AADL model artifacts
- Across layers of system architecture
- Assurance tasks as filtered views of assurance plans

Assure: Manage assurance case instance execution and results
- Multi-valued logic evaluation of verification action & results
- Acceptable risk factors (e.g., design assurance levels)
- Filtered execution of assurance plans (based on category tags)
ReqSpec: Textual Notation for Draft RDAL Annex

Objective

- Goal-oriented architecture-led requirement specification
- Explicit record of requirement decomposition, refinement, and evolution
- Integration with safety analysis via Error Model Annex V2
- Basis for incremental life-cycle assurance of evolving systems
- Bridge to existing requirements documentation practices

Stakeholder Goals

- Descriptions of intent that may be in conflict with each other
- Use cases: UML/SysML or ITU UCM

System Requirement Specification

- Specification of a consistent set of system requirements that must be met by a delivered system
- Basis for verification and assurance plans
Use Scenarios

External stakeholder requirement documents

• Leave external: requirements reference document entry (see document)
• Import as goal document
• Capture concepts, functions, and architecture elements in AADL model
  • Basis for a system requirement specification
Use Scenarios

External system requirement documents

• Leave external
  • Capture concepts, functions, and architecture elements in AADL model
  • Create reqspec requirement proxies with reference to document entry (see document)
  • Annotate requirement with model reference (for)
  • Import as requirement document
    • Capture concepts, functions, and architecture elements in AADL model
    • Annotate requirement with model reference (for)

Resulting ReqSpec specification is basis for verification and assurance plans
Stakeholder Goals Notation

Goal specification

- Associated with AADL model element
- Description, rationale
- Refinement, evolution, conflict resolution
- Traceability to stakeholders and DOORS requirement and other documents

- Document focused goal organization
  - Goals in document sections
  - Incrementally evolve model and identify model elements

- Model focused goal/requirement organization
  - Requirements for a model component
Names, Concepts, and Systems

Are the following the same:
Simple Control System
Simple Controller
System Controller

We define SCS as system type.
As we add requirements we add features, e.g., power.
As we define a model of the operational context we are explicit about the necessary and possibly missing interactions.
Textual Requirements for a Patient Therapy System

1. The patient shall never be infused with a single air bubble more than 5ml volume.
2. When a single air bubble more than 5ml volume is detected, the system shall stop infusion within 0.2 seconds.
3. When piston stop is received, the system shall stop piston movement within 0.01 seconds.
4. The system shall always stop the piston at the bottom or top of the chamber.

Same Requirements Mapped to an Architecture Model

- Importance of understanding system boundary
- We have effectively specified a system partial architecture

U Minnesota Study
System Requirement Specification Notation

Requirement specification

- Verifiable contract via verification plans
  - Satisfiable design goal
- Associated with AADL classifier or element in classifier
- Description, rationale
- Parameterization by variable
- Refinement, decomposition, inheritance, evolution
- Verifiable by verification activities
- Formal specification of predicate
- Traceability to goals, hazards
- Traceability to DOORS requirement and other documents
Requirement Traceability Reports

BIRT based prototype

- Traceability to stakeholders
- Traceability between requirements and goals
- Context command invoked on instance model, reqspec/goal file
- Source code available as org.osate.reqtrace
- Extensible for additional report templates

ReqSpec compiler validations

- Reported via Eclipse Diagnostics/Markers
  - Missing stakeholder, missing goal, duplicate goal/requirement
  - Cycles in refine hierarchy
- Report generation from Eclipse Markers
Requirement Sets

System Requirement Set

- One per component classifier (identified by for clause)
- List of system requirement declarations
- Inherited requirements according to extends hierarchy
- Inclusion of individual global requirements or global requirement sets

Global Requirement Set

- Reusable set of requirements
- For inclusion in system requirement sets
- Configurable into assurance plans
User-extensible Categories

Requirement categories for filtering and coverage
Can be organized into category sets

```
1*Quality [  
2   Behavior State Performance 
3   Latency Timing Security Safety Reliability Availability 
4   CPUUtilization MemoryUtilization 
5   NetworkUtilization Mass ElectricalPower 
6 ] 
7*Phase [  
8   StakeholderRequirements SystemRequirements PDR CDR 
9   ArchitectureDesign DetailedDesign Implementation UnitTest 
10   SystemTest 
11 ] 
12*Layer [  
13   Tier1 Tier2 
14   Tier3 Tier4 Tier5 
15 ] 
16*Kind [  
17   Assumption Guarantee Consistency Constraint Exception 
18 ]
```
Minimize Impact of Frequent Requirement Changes

Val constants can be defined for requirement set or individual requirements.

Val constants can be used in descriptions, predicates, and as parameters in verification activities.

```kotlin
system requirements scsreqs for SimpleControlSystem::SCS [ val MaximumLatency = 20 ms requirement R1 : "SCS weight limit" [ val MaximumWeight = 1.2 kg category Quality.Mass description this "shall be within weight of " MaximumWeight // second condition verifies that MaximumWeight is same as the property value WeightLimit value predicate MaximumWeight == #SEI::WeightLimit see goal SCSgoals.ng2 ]

requirement R2 : "SCS sensor to actuator response time limit" [ description this "shall have a sensor to actuator response time within " MaximumLatency category Quality.Latency see goal SCSgoals.g1 ]

```
AADL Model as System Specification

### Scenario 1
Property association value changed by user
Referenced in description instead of `val` variable

### Scenario 2
Val variable value automatically set as property value for certain verification methods

### Scenario 3
Consistency between `val` variables and property values
Consistency check: explicit or as part of verification method registry

```text
system SCS
features
    power: in feature PhysicalResources::Power;
    force: out feature;
modes
    operational: initial mode;
    standby: mode;
properties
    ACVIP::Aliases => ("SCS", "Simple Controller");
    SEI::PowerBudget => 2.5 w applies to power;
    SEI::WeightLimit => 1.2 kg;
    Physical::Voltage => 12.0V applies to power;
annex emv2 (**
    use types ErrorLibrary;
    use behavior ErrorLibrary::FailStop;
    error propagations
        force: out propagation {ServiceOmission};
        flows
            es1: error source force;
        end propagations;**);
end SCS;
```

value predicate MaximumWeight := #SEI::WeightLimit
Shared Predicates and “Compute” Variables

Think of “compute” variable as free variable in lambda expression.

verification activity result is bound as compute value
Global Requirements

```
global requirements globalReq
[
    requirement connected : "All features of a component are connected"[
        description "All features of a component are connected."
        when alisa_consistency.ModelConditions.isLeafComponent()
        category Kind.Consistency
        development stakeholder sei.phf
    ]
]

requirement Allconnected : "All features of all components are connected"[
    description "All features of leaf components are connected."
    category Kind.Consistency
    development stakeholder sei.phf
]
```

Reusable requirements, typically consistency constraints and completeness checks.

Inclusion in system requirement sets.
Applicable for component itself or recursively for all subcomponents.

```
include Peter.reql for self
include globalReq.Allconnected for self
include globalReq.connected
```

Allconnected: the verification method is assumed to recurse the component hierarchy.

Connected: the requirement is applied only to components that satisfy when condition
Requirements Decomposition

Derive requirements on subcomponents from system requirements

- Associate requirement with subcomponent in the system implementation
  - Context specific requirement
  - Requirements on subsystem to be acquired (purchased, contracted out)
- Requirement on classifier of subcomponent
  - Applies to all subcomponent instances referencing the classifier
  - Represented by property of subcomponent classifier
  - System specification used in implementation to verify that design meets system requirement
Verification & Assurance Plan Notation

Method registry
- Reusable verification methods on models and other artifacts
- Method categories for filtering and coverage
- Currently supported: Java, Resolute, analysis plugin, manual

Verification plans
- Claims aligned with system requirement specifications
- Multiple verification activities on different artifacts
- Implementation and use verification
- Conditional and backup activities (pass, fail, error)
- User definable selection categories
- Description, Rationale

Compositional assurance plans and tasks
- Configuration of assurance responsibility
- Focused assurance view based in category filters
Registry of Verification Methods

Verification Method Registry Implementation

Existing OSATE Analysis plugins
Resolute claim functions
Reflective execution of Java/Xtend functions
Execution of Junit-based code tests
More to come (e.g., Agree, Simulink, SCADE, manual)
Why And/Or Assurance Logic is Insufficient

- Assurance case: focus on final result
  - Multiple pieces of evidence must be satisfied ("and")
  - Design alternatives, different operational modes ("or")
- Resolute experience during development
  - "and": if first verification action fails, others are not evaluated
    - Equivalent to compiler stops on first error
  - "or": any satisfied verification action is fine. Pick the cheapest one to execute
  - "implies": failing first verification action results in success
Verification Activities

Multi-valued verification activity results

- Verification activity result states
  - Success, fail, error, tbd
- Compositional Argument Expressions
  - All [ va+]: a collection of independent Vas
  - Va1 then Va2: Execution of Va2 dependent on success of Va1
  - Va1 else Va2: Execute Va2 only if Va1 produces negative result
  - Va1 else [fail: Va21 timeout: Va22 error: Va23]

Mode specific verification activities

Parameterized verification activities

- Data sets as input parameters
Assurance Plans and Cases Execution

Assurance plan

- Configuration of assurance case:
  - how much of architecture to be verified,
  - which component specific and reusable verification plans to be included

Assurance plan instantiation & execution

- Automated verification activity execution
- Execution states: To be done, in progress, done, Redo
- Tracking of result state and reports

Assurance Task

- Filtered assurance plan instances based on requirement, verification, and verification activity selection categories
Automated Incremental Assurance Workbench

Identify Assurance Hotspots throughout Lifecycle

**Stakeholder Goals**

- Tier 0
  - Model
  - Ver Plan
  - Req

- Tier 1
  - Model+1
    - Model+2
    - Model+2'
    - Ver Plan
    - Req+1
  - Code

- Tier 2
  - Code
  - Model+2
  - Model+2'
  - Ver Plan
  - Req+2

**Abstraction Level**

- Low Level
  - Close to Implementation

- High Abstraction
**Original Aircraft Virtual Integration Demo**

**Aircraft: (Tier 0)**

**Aircraft system: (Tier 1)**
- Engine, Landing Gear, Cockpit, ...
- Weight, Electrical, Fuel, Hydraulics, ...

**LRU/IMA System: (Tier 2)**
- Hardware platform, software partitions
- Power, MIPS, RAM capacity & budgets
- End-to-end flow latency

**Subcontracted software subsystem: (Tier 3)**
- Tasks, periods, execution time
- Software allocation, schedulability
- Generated executables

**System & SW Engineering:**
- Mechatronics: Actuator & Wings
- Safety Analysis (FHA, FMEA)
- Reliability Analysis (MTTF)

**OEM & Subcontractor:**
- Subsystem proposal validation
- Functional integration consistency
- Data bus protocol mappings

**Repeated Virtual Integration Analyses:**
- Power/weight
- MIPS/RAM, Scheduling
- End-to-end latency
- Network bandwidth

**Proof of Concept Demonstration and Transition by Aerospace industry initiative**
- Propagate requirements and constraints
- Higher level model down to suppliers' lower level models
- Verification of lower level models satisfies higher level requirements and constraints

- Multi-tier system & software architecture (in AADL)
- Incremental end-to-end validation of system properties
Assurance Results

System SaviDemo: (S9 F1 T0 E0 tbd0 EL0 TS0)

- ClaimR1: The weight of the Aircraft system shall not exceed 70000.0 kg (S2 F0 T0 E0 tbd0 EL0)
- Evidence weightlimit: Perform full weight (mass) analysis. This includes net/gross weight
  - top AircraftSystem.SubsystemSpec: [A] Sum of weights 67935.000 kg below weight limit 70
- Evidence weightlimit2: Perform full weight (mass) analysis. This includes net/gross weight cc
  - top AircraftSystem.SubsystemSpec: [A] Sum of weights 67935.000 kg below weight limit 70
- System FGS: (S6 F0 T0 E0 tbd0 EL0 TS0)
- System ELE: (S1 F1 T0 E0 tbd0 EL0 TS0)
  - ClaimR1: The weight of the Electrical system shall not exceed...
  - Evidence weightlimit: Perform full weight (mass) analysis. This includes...
    - system ELE: [L] Sum of weights / Gross weight 75,000 kg (r2VA)
  - ClaimR2: The Electrical System shall be capable of handling at...
  - Evidence powercapacity: Analyze Electrical power demands at...
    - system ELE: ** ELE power budget total 24500.0 W exceeds
    - system ELE: budget total 24500.0 W within supply 25000.0 W

Assurance Result Metrics
Objective: Identify Assurance Hotspots
Aggregate success, fail, timeout, error, todo
Aggregate Else Success, then skip
Assurance Case Execution and Metrics

Assurance Metrics

- Requirement coverage measures
- Multi-valued verification result measures and their aggregates
  - Pass, fail, incomplete, conditions, backups
- Weighted requirement claims, verification activity results
  - Reflect importance, uncertainty, effectiveness

Guidance throughout life cycle

- Measurement based uncertainty areas (hotspots)
- Change impact prediction
References

ALISA
Online documentation:
https://rawgit.com/osate/alisa/develop/org.osate.alisa.help/contents/00-Main.html


ReqSpec
SEI TR: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=464370

JMR Shadow project:
http://resources.sei.cmu.edu/asset_files/specialreport/2015_003_001_447187.pdf

Model Examples
https://wiki.sei.cmu.edu/aadl/index.php/Models_examples