Requirements, Constraints, and Verification Activities

Peter Feiler
Feb, 2014
Outline

Architecture Centric Requirements
Requirements and Safety Hazards
Verification Activities and Assurance Cases
Bridge to traditional requirements representations

Stand-alone requirement specification (?)

Traceability into system architecture specification

Requirements and Design Information
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.
Stakeholder Needs and Requirement Categories

Table 2. Example of Stakeholder Requirements Classification. (SEBoK Original)

<table>
<thead>
<tr>
<th>Type of Stakeholder Requirement</th>
<th>Types of System Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service or Functional</td>
<td>Functional Requirements</td>
<td>Describe qualitatively the system functions or tasks to be performed in operation.</td>
</tr>
<tr>
<td>Operational</td>
<td>Performance Requirements</td>
<td>Define quantitatively the extent to which system performance and are or are not task.</td>
</tr>
<tr>
<td>Interface</td>
<td>Usability Requirements</td>
<td>Define the quality of system use.</td>
</tr>
<tr>
<td>Environmental</td>
<td>Interface Requirements</td>
<td>Define how the system is accessible to users.</td>
</tr>
<tr>
<td>Environmental</td>
<td>Interactions</td>
<td>Define the operational conditions of the system, including human element, systems or internal elements.</td>
</tr>
<tr>
<td>Utilization</td>
<td>Modes and/or States</td>
<td>Define the various operational states.</td>
</tr>
<tr>
<td>Human Factors</td>
<td>Capabilities</td>
<td>Define the limits on the operational provided system element, or element.</td>
</tr>
<tr>
<td>Logical</td>
<td>Adaptable Requirements</td>
<td>Define potential extension, growth conditions.</td>
</tr>
<tr>
<td>Design and Realization</td>
<td>Design Constraints</td>
<td>Define constraints on weight, size, power, or other characteristics.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Process Constraints</td>
<td>Define the limits on the operational provided system element, or element.</td>
</tr>
<tr>
<td>Project Constraints</td>
<td>Environmental Conditions</td>
<td>Define the environmental conditions such as temperature, humidity, salt, dust, or other sociocultural environment.</td>
</tr>
<tr>
<td>Project Constraints</td>
<td>Logistical Requirements</td>
<td>Define the logistical conditions, such as personnel, spare parts, transport, and so on.</td>
</tr>
<tr>
<td>Business Model Constraints</td>
<td>Policies and Regulations</td>
<td>Define relevant and applicable regulatory, economic, health or safety requirements.</td>
</tr>
<tr>
<td></td>
<td>Cost and Schedule Constraints</td>
<td>Define, for example, the cost of living in a particular region.</td>
</tr>
</tbody>
</table>

Leveson System Theoretic Framework

System, operational environment, development and V&V process

Software Engineering Institute | Carnegie Mellon
Reqs, Constraints, and VAs Feller, Feb. 2014
© 2014 Carnegie Mellon University
System Specification and Different Categories of System Requirements

Environmental Assumption about external component

Requirements Guarantees Assumptions

Precondition Postcondition Invariant

Developmental Requirements

Modifiability

Assurability

Quality attribute utility tree

Mission Requirements

Function

Behavior

Performance

dependability

Reliability

Safety

Security

Exceptional condition

Implementation constraints
Requirements and Architecture Model Elements

A requirements object can apply to multiple model elements individually

- System type implies each (implementation and) instance
- Manually selected model elements

A requirement applies to a collection of model elements?

- The collection together has to satisfy a constraint? E.g., implementation constraint
Requirement as Conditional Constraint

Role of what: expression (constraint)

- expresses requirement condition independent of architecture model
- Ensures that requirement is correctly specified in architecture model
- Pointer to the specification in the architecture model?

Role of when

- When does it apply: e.g., which operational or failure mode
- When assumptions for expression evaluation are met (has a property value then value is < x)
- When in the development process does it have to be satisfied by the implementation? (VA)
Different VAs are interpreting the requirement in different forms: AADL Spec or RDAL expression. Or in other form embedded in a theorem or Simulink script, etc.) that would need to be validated.

Spec of requirement independent of AADL model
actualLatency < MaxLatency of 20ms from input x to output y

Spec of requirement in context of AADL model
a pointer into the AADL model to the spec there, or in terms of properties in AADL model.
Actual_Latency < Latency with actual latency defined as computed value with method point to the analysis formula.

Verification Activity (VA): use of constraint to ensure the spec is correctly reflected in AADL (should be separate from formal spec of requirement)

Latency Property value of System S returns the maximum expected latency. The analyzer (method used in VA) evaluates actual latency against this max latency.

Or there is a property constant MaxLatency and MaxLatency is used as value for the Latency property on the system type.
On Quality of Requirements

Verification of requirement artifact:
Quality of requirements
Coverage: with respect to system spec
Complete: All states have transitions.
Input/output value, timing, rate, sequence
Consistent: overlapping /conflicting x>5 and x<4

Verification method (VM): theorem
Verification activity (VA): application of VM to a model to assess whether some constraint is met.

Verification Activity (VA) of AADL model:
Syntax, semantic, consistency rules for component types (implementations constraints, interaction contracts)

We could explicitly specify such development requirements that apply to any system development.
We can write theorems that check for it.

Stakeholder Requirements
Stakeholder
Goal

Traceability coverage as DOORS supports. Some listed in RDAL under analysis (e.g., all goals/reqs have stakeholders or are derived.

System Spec
Formal spec

RDAL Req Spec
RDAL Req
RDAL Req
RDAL Req

System model

Verifying the quality of collection of requirements

3.1.1 IEEE 830-1998 Characteristics of a Good Requirements Specification
(1) The IEEE Recommended Practice for Software Requirements Specifications [IEEE 830-1998 (5)] identifies 8 properties that a good requirements specification should exhibit:
   a. Correctness
   b. Unambiguity
   c. Completeness
   d. Consistency
   e. Ranking for Importance and Stability
   f. Verifiability
   g. Modifiability
   h. Traceability

Verification of development process requirements: VA on a req spec (set of requirements) or that two representations are consistent (req/AADL spec)
Integrated Model of Safety Hazards and Requirements

Original system requirements

I used special sub-category of requirement to represent exceptional condition/obstacle/hazard.

Derived safety requirements linked to original requirement specification

Safety hazard as exceptional condition that must be addressed through derived requirements or evidence of absence

Traceable to error source specification in AADL model

Fault Ontology as part of EMV2 Standard

Leveson STAMP/STPA
Requirement Decomposition

RDAL Req Spec

System Spec

System Realization

Formal Tech Spec (pre/post/ass/guar)

Impl Constraints

Rationale

On specific or all implementations

Constraint that talks about the parts and their interactions (connectivity, partial ordering)

Example one controller at a time controls system

Two (redundant) instances of sensor

Interaction contracts along connections

Requirement

System

System

T/S

T/S
Compositional Verification of Requirements

Theorems represent a system requirement, its presence in the AADL model, or verification method of implementation against requirement.

Establish validity of subrequirement/assumption and implementation verification activity.

Argumentation implicit in and of proof list. In some examples it was embedded in top-level expression of proof logic.

In an annex library: Some are generic, some only work for specific system types or implementations.

Resolute statements currently not integrated with RDAL, but the theorem corresponds to a library of Lute functions and the prove the RDAL Req expression invoking a single Lute lib function.

Mixture of system and process requirement verification.

Req/Verification activity instance: the prove is evaluated for each instance of the system implementation.

Application of proofs results in an instance of an assurance case record.
Compositional Verification

Verification within the space of architectural design/decomposition

VA compositional formula: may use subcomponent properties, i.e., requirements that it assumes are met.

Example: latency calculation based in latency specs of subcomponents.

Input-output transformation substitution (model checking)

VA guarantee-assumptions: output to input types (SAVI functional integration)

Assumption/guarantee, pre/post: Resolute/BLESS

Verified by AADL compiler, functional integration checker, Lute equivalent, BLESS, etc.

Example requirement: ActualLatency < MaxLatency of 250ms

In AADL spec: value(Latency) = 250 ms

Latency analysis (verification method) interprets Latency property as MaxLatency
Combining AADL and Simulink Based Verification

Simulink to AADL bridge

Architecture compliance

Consistency checking between Simulink architecture structure and AADL-based architecture spec.

Basis for Simulink based simulation as verification against an AADL specification, and basis for AADL based Resolute/Agree verification of Simulink StateFlow behavior.
Combining Requirement Validation & Compositional Verification

Combining Resolute, RDAL, Lute, Agree, others

RDAL Verification Activity: Simulink or SCADE based verification. Result to be reflected in requirement satisfaction status.

Evidence records below claims that requirements have been met

Process requirements as verification activities. Req object expression used as verification activity. What should go in a Req object vs. a VA object?

Requirement quality (coverage, consistency) and assumption evidence

Safety hazards have safety requirements that address them when satisfied

Linkage to Resolute proofs as verification activities.
Assurance Case Record

GSM, CAE, Eurocontrol safety case, Confidence map
OMG Structured Assurance Case Metamodel (SACM)

System and SW Requirements

Claim: Req instance

Evidence: VA instance

Formal Tech Spec
(pre/post/ass/guar)

Impl Constraint

Requirement quality verification evidence

Decomposed to

Verification against quality requirements

Compositional evidence

Argumentation: implicit or explicit And/Or/Implies logic & rationale

System architecture

System

System Realization

Subsystem

Design/Source code

T/S

Requirement quality verification evidence

Rationale

Requirement

Refined to

Requirement

 Requirement
Claim and VA Status

- **Claim and VA Status**

- **System Powertrain**
- **System ETC**

- **Model Variant Continuous**
  - **Continuous**
  - **Model Variant**

- **Input dataset**
- **Verification method**
- **Result dataset**
- **Test parameter**

- **SVM Objects**

- **Claim**
  - **Stat use**
  - **Stat us**
  - **Status**

- **VA**
  - **Status**
  - **TBD, InProgress, Pass, Fail, Unknown**

- **Verification Activity**
  - **Simulated environment test**

- **VA Instance**: Evidence object with state invoking VA spec
  - **Verified**: pass/fail
  - **Satisfied**: quantitative 0..1

- **Argumentation logic**
  - Which combinations of VA

- **VA spec**: Verification method (VM), applied to model, with input/param to produce expected output/meet specified constraint.
  - May include a script invoking VM multiple times with different parameters/data sets

- **Some Vas may be long-running**
Verification Method & Script Library

**Verification Method 1**
- methodScript
- Model spec
- Model parameter Spec
- Dataset file spec

**Verification Method & Script Library**
- Defined once Scripts provided

**Model Execution Script**
- Test parameter = xx

**Verification Method Instantiation**
- Once per activity
- Selected model spec drives actual parameters
- Test parameter = yy

**MatLab scripts**
- Model Execution Script Template
- Model Comparison Script Template
- Multi-dataset Model Execution Script Template

**Model Execution Implementation**
- ExecAPI
  - Matlab
  - mdl
  - Java
  - SimPlant

**SVM**

© 2014 Carnegie Mellon University
When should a VA be performed and expected to be satisfied (based on development phases)?

Traceability to requirement. Verification that requirement is correctly reflected in respective parameters.

This method tests if time the output spends between two thresholds is smaller than some upper bound.

Rise Time is defined as the time required for the throttle plate angle response to a step change in pedal position to rise from 10% of the steady state value to 90% of the steady state value.

The rise time for step changes from closed to fully open is 100ms.
Verification (&Valid) Artifacts and Ontology

We have different categories (types) of verification activities.

Need a place where the result records are stored for post-mortem examination (counter examples, simulation traces, etc.)

Can be a person “executing” a checklist

Correct version of tool with correct switches set
OMG SACM Meta Model

SACM Evidence Metamodel consists of 18 class diagrams. SACM Evidence Metamodel is delivered as a single UML subpackage ‘Evidence’ of SACM.

The SACM Evidence Metamodel consists of the following logical parts:

- Evidence Items
- Formal Elements
- Evidence Assertions
- Administration

The Evidence Items part defines the physical evidence, provided in the form of documents, records, and sometimes other material exhibits.

The Formal Elements part defines the logical assertions, provided in the form of individual propositions. These propositions use an external vocabulary related to the subject area for which an argument is being provided. The Formal Elements part defines a subset of an OMG Semantics of Business Vocabularies and Business Rules (SBVR) fact model in the form of atomic formulations based on fact types with roles bound to individual concepts. SBVR is not used directly because of the semantic differences between fact models in linguistic models as they are defined in SBVR, conceptual models and “asserted fact models” involved in evidence collection and evaluation. Formal Elements represent a conceptual model underlying the entire assurance case.

Evidence Assertions part defines various statements that can be made about the evidence items, such as documents, records and exhibits, and their relations to the subject area claims. Evidence Assertions include statements that are related to various essential properties of evidence items. A large group of statements are the so-called evidence evaluations, including assertions of the evidentiary support (relations between evidence items and the subject area claims), assertions related to the interpretation of physical evidence and document, assertions about the conflicts in evidentiary support and resolutions of these conflicts. Other statements are assertions related to provenance, custody and timing of the evidence items and evidence evaluations. The last group of statements qualify the evidentiary support that evidence items confer on the subject area claims.

The Administration part defines an EvidenceContainer element that organizes individual evidence items and evaluations.
OMG SACM Meta Model

SACM Argumentation Metamodel

9.1 Argumentation Class Diagram
  9.1.1 ArgumentationElement class (abstract)
  9.1.2 Argumentation Class
  9.1.3 ArgumentElement Class (Abstract)
  9.1.4 Assertion Class (Abstract)
  9.1.5 ReasoningElement Class (Abstract)
  9.1.6 InformationElement Class
  9.1.7 CitationElement Class
  9.1.8 Claim Class
  9.1.9 EvidenceAssertion Class
  9.1.10 ArgumentReasoning Class
  9.1.11 AssertedRelationshipClass (Abstract)
  9.1.12 AssertedInference Class
  9.1.13 AssertedEvidence Class
  9.1.14 AssertedChallenge Class
  9.1.15 AssertedCounterEvidence Class
  9.1.16 AssertedContext Class

Evidence Elements

10.1 Evidence Elements Class Diagram
  10.1.1 EvidenceElement (abstract)
  10.1.2 EvidenceItem (abstract)
  10.1.3 Exhibit
  10.1.4 Document
  10.1.5 Record
  10.1.6 FormalElement (abstract)
  10.1.7 FormalObject (abstract)
  10.1.8 FormalAssertion (abstract)
  10.1.9 EvidenceGroup

10.2 EvidenceAssertions Class Diagram
  10.2.1 EvidenceAssertion (abstract)
  10.2.2 EvidenceProperty (abstract)
  10.2.3 EvidenceEvaluation (abstract)

11.1 ExhibitProperties Class Diagram
  11.1.1 Exhibit Property
  11.1.2 HasElectronicSource
  11.1.3 IsPartOf
  11.1.4 HasMedia
  11.1.5 IsBasedOn

11.2 DocumentProperties Class Diagram
  11.2.1 Document Property
  11.2.2 HasVersion
  11.2.3 IsExpressedInLanguage
  11.2.4 HasSecurityClassification
  11.2.5 IsReleasableTo
  11.2.6 Originality
  11.2.7 OriginalityLevel (enumeration)
  11.2.8 Consistency
  11.2.9 ConsistencyLevel (enumeration)
  11.2.10 Completeness
  11.2.11 CompletenessLevel (enumeration)
  11.2.12 Reliability
  11.2.13 ReliabilityLevel (enumeration)
  11.2.14 ExtendedDocumentProperty
Implications on RDAL

Additional Requirement and verification activity categories

• Clarification of condition/expression role in Req (claim)
• Multi-valued state, persistence vs. dynamic evaluation
• State determined by argument logic vs. cond/expression

Explicit representation of argument (logic to determine satisfaction)

• Currently alternative OR and decomposition AND

Elaboration of verification activity

• Multi-valued state - persistence
• Verification activity assumptions
• Long running verification activities

Library of verification methods

• Notations supported by extension
• External methods invoked by script
Contact Information

Peter H. Feiler
Principal Researcher
RTSS
Telephone: +1 412-268-7790
Email: phf@sei.cmu.edu

Web
Wiki.sei.cmu.edu/aadl
www.aadl.info

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Customer Relations
Email: info@sei.cmu.edu
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257