

AADL EMV2 Errata

Peter Feiler

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

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.

DM17-0314

Errata

Issue #11/#26: out propagation condition in terms of subcomponent out propagation

Issue #28: typeset on right of transition or out propagation conditions should be a single type

Issue #29: Allow state declarations in component error behavior

Issue #12: EMV2 does not deal with reference to array elements for subcomponents and features

Revision

Name path consistency

Trigger condition logic

Version management of EMV2
Implementation of approved errata

Name Path Consistency

We have syntax for applies to in properties (or reference values)

- In core: [a.b.c@\(emv2\)es.ep](#)
- In EMV2: “escape” to core: [^sys.a.b.c@ep.et](#) escape character and @ without annex name
- In EMV2: es.ep.et: et is type thus should use {et} as we do for propagation constraints ep{et}

In core element references in EMV2 constructs

- Subcomponents & state in composite: s1.s2.estate1 => [^s1.s2@estate1](#)
- subcomponent out propagations in out propagation conditions:

Logical expression issues

Binary exclusive or Operators

- Keyword OR: XOR semantics
- $A1 \text{ XOR } A2 \text{ XOR } A3$
 - Boolean logic interpretation: $A1 = T \wedge A2 = T \wedge A3 = T \Rightarrow T$
 - Intended interpretation: XOR (A1, A2, A3), i.e., one failure only

Event and state based condition evaluation

- Error events are occurring independently
 - Only one at a time can trigger a transition
 - Should not directly impact outgoing propagation condition
- Error propagations reflect error state
 - Evaluation of multiple is possible

Logical operators on propagations

Existing operators

- k **ormore** n: inclusive or starting with k subset
- k **orless** n: inclusive or up to k subset

Proposal: add k **of** n operator

- Any subset of size k failure
- 1 **of** (a1, a2, a3) exactly one failure
- Use case: triple redundant sensors
 - If one fails go to degraded mode
 - If 2 or more fail go to failstop mode

Support arrays

- 1 **ormore** (p1, p2, p3)
- 1 **ormore** (p[3])

Binding points: potentially unknown # of propagation sources

- Does conditional logic make sense on bindings

Principals of Fault Logic Operators

Operators on collection of fault triggers

- We already have k or more(T)
 - T = collection of triggers
 - $n = \text{size}(T)$
- Related to the count of how many trigger conditions hold
 - $\text{count}(T)$
- $\text{count}(T) = n$: **All**(T)
 - same as multi-operand **and**
- $\text{count}(T) = k$: **k of**(T)
 - 1 of(T) same as multi-operand exclusive or
 - Often equated with k of n voting
- $k \leq \text{count}(T) \leq n$: **k or more**(T)
 - 1 or more(T) same as inclusive or
 - Sometimes equated with k of n voting
- $0 \leq \text{count}(T) \leq k$: **k or less**(T)
- $j \leq \text{count}(T) \leq k$: **j..k of**(T)

Practical Set of Operators

Minimal set of operators

- All, 1 of
- Others can be expressed in terms of those two

Nested operators: expressions as elements of T

- Restricted to a normal form
- Full set

Context specific restrictions

- Composite state expressions
 - States only
- Transitions
 - Meaning of multiple transitions between same states (“or” or “xor”)
 - No “and” of multiple events (“xor” since events are not simultaneous?)
- Out propagation conditions
 - Implicit state: state on left of statement
 - Incoming propagations only (incl. subcomponent out prop)
- Detection conditions
 - State, incoming propagations