Integration of Scade with AADL+BLESS

Brian R Larson
brl@ksu.edu

February 1, 2017

SCADE AADL Capabilities

At the October, 2016, standard committee meeting, ANSYS presented preliminary work integrating their SCADE software design tools with AADL.

SCADE and “Scade" is the title of both the language and tools.

- **SCADE Suite** is used to design software–both dataflow and state machine “operators"
- **SCADE System** uses SysML to design architectures
- **SCADE Display** is used to design user interfaces

When I found that SCADE Suite used finite-state machines, which could be both simulated, and generated DO-178c qualifiable code, I proposed an experiment to try to integrate Scade with AADL and BLESS.

Thanks

Guiherme Goretkin of ANSYS attended the Pittsburgh meeting in October, and has been very helpful.

ANSYS SCADE Support has also been quickly responsive.

BLESS Language

The Behavior Language for Embedded Systems with Software was created to prove that behavior meets specification.

BLESS is BA with non-executed, temporal-logic, assertions. Logic is first-order predicate calculus augmented with two, simple, temporal operators, `@` `^`, defining when predicates are evaluated. Couldn't declaratively specify pacemaker behavior with other temporal logics (I could find in 2007).

Assertions on ports, together with a component invariant declaratively specify behavior.

Work on BLESS started when BA was first proposed (2007). BLESS has formal definitions of every construct.

BLESS Proof Tool

Transforms proof outlines into inductive proofs of correctness guided by human selection of tactics.

Connections get assume/guarantee contracts. Component composition proves containing component's invariant from the invariants of its subcomponents.

Simulation and Code Generation

Q) How do I know that my proved-correct source will execute the same way when deployed?

A) DO-178c qualifiable code generation.

Q) How do I know the behavior does what's needed?

A) Simulation scenarios for early validation.

Scade provides what's missing from BLESS tools.

Transitions

Both Scade state machines and BLESS have transitions with:

Source, destination, transition condition, action.

Subject	Scade Suite	AADL+BLESS
timing	synchronous	periodic aperiodic
interaction	signals (streams)	ports
i/o	use interface ID	port input/output
previous/next	X 'last	X'
hierarchy	Operator	process, system
states	hierarchical	complete, execution
final	resume	thread stop
condition	boolean expression	dispatch, execution
data types	constructed	constructed
action	just :=	loop, branch communication, :=
persistence	implicit	X' := X
action	states, too	only transitions

Diagram → Text

Scade diagrams are easy to make and understand.

Once converted to text, cannot go back to diagram.

AADL Graphical Editor keeps diagram & text consistent.[1]

Want to use GE for BLESS state machines.

[1] Thanks Phil!

Windows 10

I hate Windows. Scade only on Windows and Linux.

Struggled with Windows 10.

Probably Ubuntu Linux would have been easier.

Scade Import

Metamodel to metamodel translation.

Did not work.

Fixed much, but still doesn't work.

Composite components to composite operators

```
L1, L2, L3 := oper(L4, L5, L6, L7)
```

Converting AADL connections to local variables was easy.

Converting components with in/out ports identifiers to function-like use of operators was difficult.

Effort suspended.

Flattening Actions

Because Suite actions are assignments, BLESS (BA) actions need to be ‘flattened’.

Loops and alternatives need to be transformed into states with simple actions where all loops and alternatives use transition conditions.

Coincidentally, this is how JP’s formal semantics for BA work. While merging JP’s formal semantics into BLESS, substitutions of complex actions with more, but simpler, states and transitions are included with each of the loops or alternatives.

Splitting Complete States

Entering a complete state suspends execution until next dispatch.

This behavior is obtained in Suite by splitting complete states into two, and making the guard of the transition between them the dispatch condition.

Each Suite automaton get a "dispatch" port.

Isolette Demo

Findings

- Scade (Suite) is industrial-strength, with ample, clear documentation and responsive customer service.
- Scade (Suite) still produces difficult-to-understand error messages.
- BLESS state machines can be mapped onto Suite state machines, with some accommodations for events and thread dispatch.
- Translating Suite to BLESS seems much easier, but didn't get it working (OSATE importer is buggy, and annotation with BLESS assertions just started).

With Scade Architect?

Closer semantics to AADL (ports, events, dispatch)

Already attaching AADL properties to Scade objects, for BLESS assertions

Try Architect+Suite state machines -> BLESS verification conditions

Try reactive behavior (VVI) with continuous-time specification.

Create guidelines for design style to prove correctness