Timeout Runtime Service for AADL

Brian R Larson

October 3, 2016

An AADL Runtime Service for Timeout?

BA and BLESS have timeout as a dispatch trigger.

Should there be an AADL runtime service for it?

It *is* part of the `dispatch` condition. Dispatching threads is a major function of the AADL runtime.

As an action item, I attempted to write the text for a V3 runtime timeout service.

AADL Standard Timed Dispatch

The AADL standard defines a *timed* dispatch protocol with timeout:

5.4.2 Thread Dispatching

(41) For a thread whose dispatch protocol is `timed`, a dispatch request is the result of an event, event data, or remote subprogram arrival, or it occurs by an amount of time specified by the `Period` property since the last dispatch.. In other words, the `Period` represents a time-out value that ensure a dispatch occurs after a given amount of time if no events, event data, or remote subprogram calls have arrived or are queued.

The `Enabled` function by default has the value true if there exists a port or provides subprogram access (p) in the set of features that can trigger a dispatch (E) with an event, event data, or call in its queue, or time equal to the `Period` has expired since the last dispatch, i.e., ∃ p in E: p ≠ ∅v t = `Period`. t is reset to zero at each dispatch.

This evaluation function may be redefined by the Behavior Annex (see Annex Document D). The `Wait_For_Dispatch` invariant is that no event, event data, or call is queued, i.e., $\forall$ p in E: p $= \varnothing \wedge t <$ `Period`. The `Compute_Entrypoint` of the port triggering the dispatch, or if not present that of the thread, is called. If a timeout occurs, i.e., the dispatch is triggered at the end of the period, the `Recover_Entrypoint` is called.

The timeout needed for BA and BLESS is very similar, just reset by events with duration either by property or parameter.

BA/BLESS Timeout Dispatch Trigger

BA and BLESS allow timeouts to be used for thread dispatch conditions.

```
dispatch_trigger_condition ::=
| . . .
| completion_relative_timeout_catch
| dispatch_relative_timeout_catch

dispatch_relative_timeout_catch ::= timeout

completion_relative_timeout_catch ::= timeout
  [ ( timeout_reset_port { or timeout_reset_port }* ) ]
  behavior_time
```

BLESS allows timeout to be a dispatch trigger as part of a dispatch condition.

No Runtime AADL Service for Timeout

None of the defined AADL runtime services performs timeout.

Because timeout is implemented by the dispatcher (for threads), a runtime service will ensure that all implementations of BA/BLESS behave similarly.

Proposed Language

`Timeout(TimeoutDuration,ResetPorts[,DispatchPort])`

The `Timeout` runtime service is called to create a timer to generate an event signaling timeout. An event at any of the listed `ResetPorts` resets the timer to the `TimeoutDuration`, and starts the timer. When `TimeoutDuration` has elapsed since the most-recent event at a port in `ResetPorts`, an event is sent on the `DispatchPort`. Changes in the value of `TimeoutDuration` are ignored until the next reset.

This implicit `DispatchPort` is intended to be included in a `DispatchPorts` parameter of `Await_Dispatch`, and its `DispatchConditionFunction`.

```
subprogram Timeout

features
  --reference to time value for duration of timeout
  TimeoutDuration: requires data access;

  --list of ports that can reset (restart) the timeout
  ResetPorts: in parameter <implementation-defined port list>;

  --port which may cause dispatch
  DispatchPort: in parameter <implementation-defined port>;

end Timeout;
```

I fail to see why it is a output dispatch port. I guess you mean an “in" DispatchPort from your definition below.

It would be also relevant to have it conformant to the `timed` scheduling policy. In particular this part 5.4.2 (41) For a thread whose dispatch protocol is timed, a dispatch request is the result of an event, event data, or remote subprogram arrival, or it occurs by an amount of time specified by the Period property since the last dispatch..[..]

If a timeout occurs, i.e., the dispatch is triggered at the end of the period, the Recover_Entrypoint is called.

a) call it Set_Timeout, all other services have a verb as prefix

b) What are the scheduling policy of reference for using this service ? I guess at some point we will need a “BA_Defined" scheduling policy to cater for the various options supported by BLESS/BA that contradicts the standard dispatch invariant and enabled function. In particular those related to timeouts.

Yes, it reads BAD, mostly because

c) Can you (do we want to ?) distinguish timeouts that are to be semantically mapped to error states ?

d) Naive question: how do you invalidate a timer ?

e) Duration is discutable, I would like also absolute time, as in many critical systems (non medical ones)

I guess a point to be further refined for Pittsburgh ;-)

My hand-written notes are too cryptic about where the request for a timeout related runtime service came from. It came up during the discussion of comments on the V2.2 ballot, but none of the ballot comments raised it as an issue.

My notes indicated that the runtime service was related to resetting the time out. The challenge is to define the service as abstractly as possible to not imply a specific implementation and not allow consideration for others. That's what I tried to do when defining the runtime services in the current core AADL standard.

So let me step back and understand the need first: The current set of service call definitions in the core AADL standard are all calls that an application developer could make use of.

I would not expect application code to contain calls to reset timeouts as that would change the declared execution semantics. I also would not expect runtime services for creation of timers for use in application code as users could create periodic dispatch behavior even for aperiodic threads.

When generating a runtime executive I can see some runtime services that are not available to the application. For example, when representing the runtime executive implementation as AADL for code generation purposes.

I would expect such runtime services to be part of the code generation annex. Even then it is not clear how far we need to go in terms of having a service to create timers, e.g., to implement periodic threads. Depending on the operating system this may be expressed as a configuration file that is interpreted rather than a sequence of calls to create timers.

I can see the code generation process making use of a runtime service for resetting timeouts associated with the thread or with individual incoming ports that trigger dispatches (Reset_Timeout with parameters).

How best to get timeout?

Proposed: Reset with event on listed port instead of dispatch, and used in a dispatch condition function.

What's needed is very similar to what's done for `timed` dispatch, so what's the problem?