



[Home](#) | [Search](#) | [Contact Me](#)

My personal manifesto about the widely misunderstood field of real-time computing...

Navigation

Introduction
About Me
Real-Time
Distributed Real-Time
Distrib. Real-Time Java
Real-Time Java
Real-Time CORBA
Real-Time Resources
Our Documents

News

Real-Time:

[Real-Time Overview](#)

[Time Constraints](#)

[Deadlines](#)

[Time/Utility Functions](#)

**[Time Constraint Scopes
and Priorities](#)**

[Sequencing](#)

[Sequencing Criteria](#)

[Timeliness Optimality](#)

[Predictability](#)

[Hard and Soft Real-
Time](#)

[Sequencing Algorithms](#)

[Worked Examples](#)

[Coastal Air Defense](#)

Last updated: 02/29/2004 18:13:22

Time Constraint Scoping

A time constraint has an explicit or implicit lexical *scope* -- a region of code executed by the thread while time-constrained.

The most natural and flexible approach is for the application programmers to have explicit scope delineation constructs for directly specifying actual thread completion time constraints that are inherent in the application -- using a hard deadline example, `BeginTimeConstraint {HardDeadline,t,...}, EndTimeConstraint`. See Figure 1.

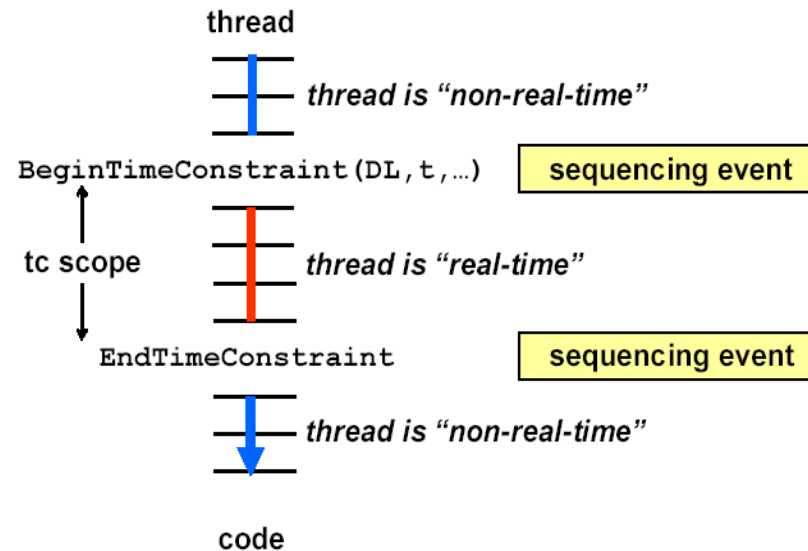
[AWACS Tracker](#)[History](#)

Figure 1: Time Constraint Scope (Hard Deadline Example)

If the thread execution point reaches the `EndTimeConstraint` statement by the deadline time t , then the deadline is met.

Otherwise an appropriate exception may be thrown when the deadline time passes, causing one or more exception handlers to execute. Typically, an exception handler executes under the time constraint of the next outer-most scope, if any; otherwise it may execute with a designated time constraint..

In either case, the timeliness of the thread's execution through the deadline's scope is measured by the value of the time constraint's time/utility function (or degenerate special case) at the time the thread execution terminates (whether before or after the deadline).

The passage of a thread execution point into or out of a time constraint scope is a sequencing (e.g., scheduling) event.

While the thread is time-constrained (executing within the time constraint scope), it can informally be thought of as a "real-time thread," and while it is not time-constrained (executing outside that scope) it can be thought of as a "non-real-time thread."

Time constraint scopes may be nested. The semantics of "nested" - i.e., how time constraints are composed - is clear in the case of deadlines: the tightest deadline is dominant. But for arbitrary time/utility functions, the semantics of nesting (composition) are much more complex. [More to follow]

Such constructs for specifying actual time constraints on code regions executed by a thread first appeared in the [Alpha distributed real-time OS kernel](#) [Clark et al. 93].

Subsequently, they became part of the [Real-Time CORBA 2.0](#) [OMG 01] specification; the [Distributed Real-Time Specification for Java](#) [Jensen 02] is expected to provide a similar capability. In a [distributed real-time system](#), time constraints must be interpreted uniformly, so the system must have global physical time that is appropriately well synchronized at each node.

But real-time computing systems that provide for using actual time constraints are the exception. The usual practice is that system software (e.g., the OS) provides only a priority mechanism, and the application programmers are required to establish the priority semantics by mapping time constraints (such as deadlines) onto them. Such mappings are complicated in all the but simplest, smallest, and least dynamic systems -- for example:

- priority assignments are not modular - they require global knowledge of all other priority assignments (whereas time constraints do not);
- the granularity of time constraints is typically much finer than that of priority ranges;
- semantics are associated with priorities by the users, the system and application software, and the hardware -

not always (or even usually) entirely coherently.

These priority mapping complications are exacerbated in [distributed real-time computing systems](#).

Given COTS system software that provides only priorities, necessitating mapping to emulate time constraints, the scope delineation constructs can be furnished by a library.

References

[Clark et al. 93](#) Clark, R.K., E.D. Jensen, and F.D. Reynolds, *An Architectural Overview of the Alpha Real-Time Distributed Kernel*, Proc. USENIX Workshop on Microkernels and other Kernel Architectures, pp 200-208, 1993.

[Jensen 02](#) Jensen, E. D., A. Wellings, R.K. Clark, and D.M. Wells, *The Distributed Real-Time Specification for Java: A Status Report*, Proc. Embedded Systems Conference/San Francisco, September 2002.

[OMG 01](#) Object Management Group, *Real-Time CORBA 2.0: Dynamic Scheduling Specification*, orbos/01-08-34, 2001.

[Next: Sequencing](#)

[Back to Time/Utility Functions](#)



[Add to Favorites](#)



[Print Page](#)



[Download a PDF copy of this page](#)

[View Site Changes](#) [XML](#) | [Site Updated 02/29/2004](#) | [Legal](#)