E. Douglas Jensen's
Real-Time for the Real World

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

Last updated: 02/29/2004 18:15:39

Time/Utility Functions

A deadline is limited in expressiveness by its singular inflection point and linear timeliness metric (lateness). To overcome that, Jensen [Gouda et al. 77] created a more general and expressive model for time constraints. That work was continued by Jensen's Archons Project at CMU [Jensen et al. 85] [Locke 86] [Clark 90] and by Jensen's research and product development projects at subsequent employers [ccur reference] [dec reference] and collaborators [sri references]. The model was initially called "time/value functions" (still the term most other researchers on this topic use), then (to help clarify the term) "benefit functions," and finally "time/utility functions."

In this model, an action time constraint expresses the utility (either reward or penalty) for completing the action as a function of when the action is completed. See Figure 1.

News

Real-Time:

Real-Time Overview
Time Constraints
Deadlines
Time/Utility Functions
Time Constraints Scopes and Priorities
Sequencing
Sequencing Criteria
Timeliness Optimality
Predictability
Hard and Soft Real-Time
Sequencing Algorithms
Worked Examples
Coastal Air Defense

AWACS Tracker

History

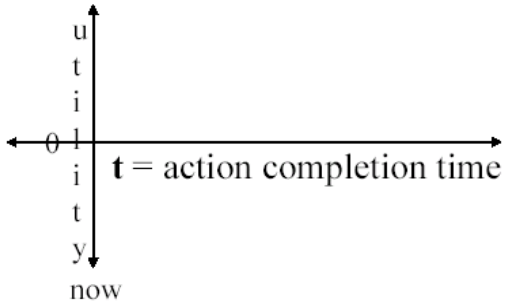

Figure 1: Time/Utility Function Axes

For simplicity, here we will limit the discussion to functions that are either non-convex or constant; therefore, there are no local minima that are not global (i.e., if a function decreases, it must not increase).

"Now" on the x (time) axis is the current time. Temporarily, we will assume for simplicity that a time-constrained action's release time (i.e., when it becomes eligible for execution) is "now;" we generalize to arbitrary release times on the Sequencing page.

The time at which a function ceases to be defined (i.e., the maximum value of its pre-image) is its *terminal time*. When the terminal time of a ready action's time/utility function is reached, that thread is removed from the ready queue. If the terminal time of an executing action is reached, that action receives an exception (which usually will, but need not, result in it being aborted).

Utility may be cumulative as a function of the progress of (e.g., execution time received by) the action, as discussed on the Progressive Utility page, but here we confine ourselves to the case where the utility is established only when the action completes. More generally, utility may be a multi-variable function of other factors in addition to time (e.g., power consumption, duty cycle), but such cases are beyond the

scope of this introductory discussion.

Constant time/utility functions - i.e., $f_k(t) = U_k \ \forall t$ in the functions' domains - represent non-time-constrained actions; their utilities are one way of expressing their relative importances. See Figure 1.

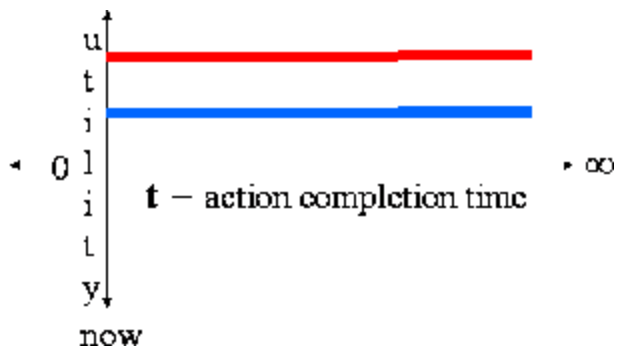

Figure 1: Constant Time/Utility Functions

This has the advantage of allowing both time-constrained and non-time-constrained actions to be scheduled coherently by the same algorithm (as discussed on the Sequencing and Sequencing Algorithms pages).

A hard deadline is expressed functionally as a binary unit-valued downward step at the deadline time: a singular positive utility value "1" if the deadline is met; otherwise "0" if the deadline is missed. (A hard deadline time/utility function is the inverse of scheduling theory's *unit penalty* function.) See Figure 2.

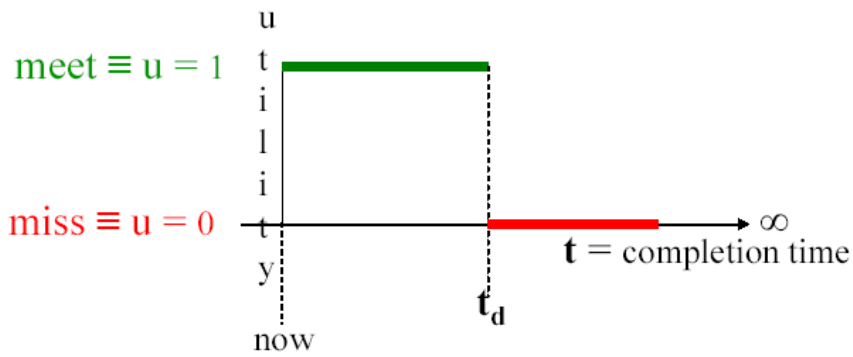

Figure 2: Hard Deadline Time/Utility Function

A deadline is expressed functionally in either of two ways:

- a downward step function (a hard deadline is the special case where the downward step function is unit-valued)
- linear functions based on lateness

Figure 3 illustrates that a deadline (unlike a hard deadline) function may have any utility values on each side of the deadline time. These utility values are one way of expressing the relative importance of actions.

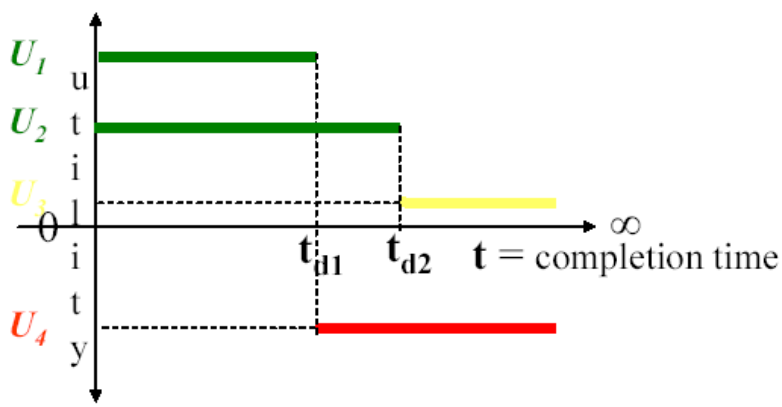

Figure 3: Deadline Time/Utility Step Functions

When a deadline is expressed functionally in terms of lateness, as shown in Figure 4, it has a slope of +1 and is inconsistent with the convention of the time/utility function model that larger utility values correspond to better timeliness. (t is just an arbitrary scaling constant proportional to $-t_d$.)

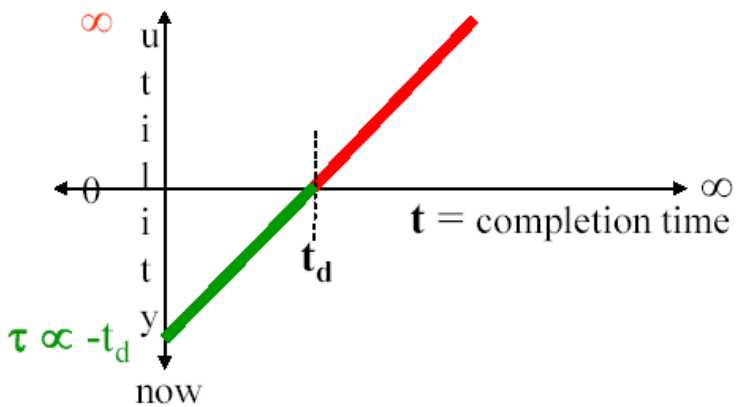

Figure 4: Deadline Time/Utility Function Based on Lateness

Instead, a deadline's timeliness metric can be based on inverting lateness and expressing tardiness and earliness in terms of that:

- -lateness = deadline - completion time
- tardiness = min[0,-lateness]
- earliness = max[0,-lateness]

as illustrated in Figure 5.

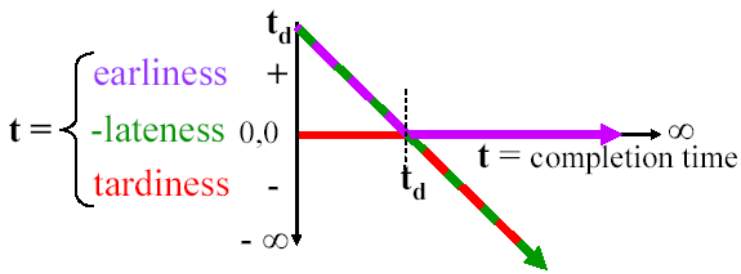

Figure 5: Deadline Inverted lateness Metrics

Then a deadline can be expressed functionally in terms of the inverted lateness with a slope of -1, consistent with the convention of the time/utility function model that larger utility values correspond to better timeliness. See Figure 6.

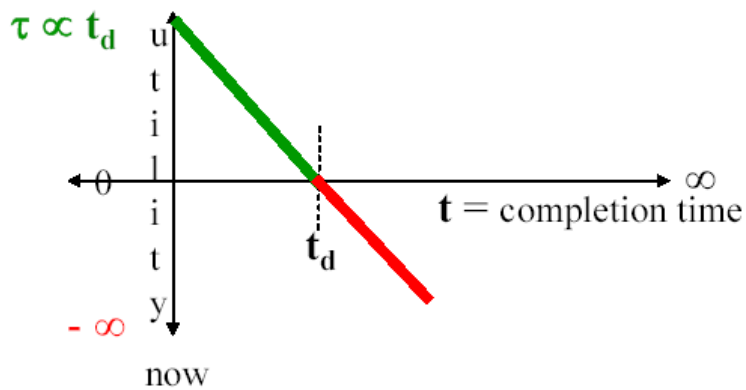

Figure 6: Deadline Time/Utility Function Based on Inverted Lateness

A *general time constraint* is any functional relationship between the time when the thread execution point reaches the end of the time constraint scope, and the utility to the system of when it does. Clearly soft deadlines are special cases of general time constraints. See Figure 7.

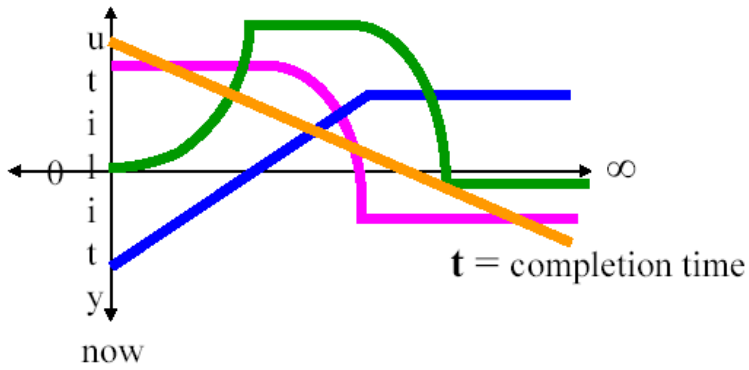

Figure 7: General Time Constraints

Time/utility functions can reduce system complexity, perhaps contrary to initial impressions.

Any system and its environment has inherent complexities and anomalous situations. The model for designing and using the system should facilitate understanding and managing those complexities and anomalies, not obscure them and add more of its own. Time/utility functions may appear more complicated than simpler-looking conventional real-time models (cyclic executives, and priorities). But the conventional real-time models often are so poorly matched to the realities of a system and its environment – using them can be a primary source of complexities and anomalies – and thus costs and undependability ("If the only tool you have is a hammer...").

Suppose a system has actions whose time constraints are naturally expressed as depicted in Figure 7 (a realistic example based on experience with actual deployed real-time computing systems). If conventional priorities are the only facility available, attempting to manipulate them to roughly approximate the effect of using time/utility functions is extremely complex. The other alternative is the widely used Procrustean Bed approach of abstracting the time constraints

into deadlines and then mapping those into priorities (an NP-hard mapping problem feasible only in small scale systems). The complexity of those two steps is typically compounded by the resulting need for increased hardware resource capacity.

Careful engineering and implementation are necessary to determine if, and ensure that, the higher costs of time/utility function time constraints and scheduling algorithms yield commensurately better application and system timeliness (since they do not always do so, especially in simple systems).

On the next page, we look at time constraints as a programming construct -- their lexical scoping, and their semantics in comparison to priorities.

References

Clark 90 Clark, Raymond K., *Scheduling Dependent Real-Time Activities*, Ph.D. Thesis, CMUCS-90-155, School of Computer Science, Carnegie Mellon University, 1990.

Gouda et al. 77 Gouda, M.G., Han, Y.W., Jensen, E.D., Johnson, W.D., Kain, R.Y., *Distributed Data Processing Technology, Vol. IV, Applications of DDP Technology to BMD: Architectures and Algorithms: Chapter 3, Radar Scheduling; Section 1, The Scheduling Problem*. Honeywell Systems and Research Center, Minneapolis, MN. September 1977.

Jensen et al. 85 Jensen, E. Douglas, Locke, C. Douglass, and Tokuda, Hideyuki, *A Time-Driven Scheduling Model for Real-Time Operating Systems*, Proc. Real-Time Systems Symposium, IEEE, 1985.

Locke 86 C. Douglass Locke, *Best-Effort Decision Making for Real-Time Scheduling*, Ph.D. Thesis, CMUCS-86-134, Department of Computer Science, Carnegie Mellon University, 1986.

Next: Time Constraint Scoping and Priorities

Add to Favorites

Print Page

Download a PDF copy of this page

View Site Changes XML | **Site Updated 02/29/2004** | **Legal**