

# The Case for TUFs and UA Scheduling in RT UML Profile: A Real-Time Scheduling/Operating System Perspective

Binoy Ravindran\*, E. Douglas Jensen<sup>†</sup>, Peng Li\*, Haisang Wu\*, and Shahrooz Feizabadi\*

\*ECE Dept., Virginia Tech  
Blacksburg, VA 24061, USA

{binoy, peli2, hswu02, shahrooz}@vt.edu

<sup>†</sup>The MITRE Corporation  
Bedford, MA 01730, USA

jensen@mitre.org

## Abstract

*This position paper makes the case for incorporating time/utility functions (TUFs) and the paradigm of utility accrual real-time scheduling in the planned, updated version of the UML Profile for Schedulability, Performance, and Time. The case is made by arguing that the key underpinning of the current state-of-the real-time practice — the priority artifact — and that of the current state-of-the real-time art — deadline-based timeliness optimality — are grossly inadequate for specifying application timeliness objectives, for reasoning about timeliness behavior, and for performing resource management that can dependably satisfy timeliness objectives in many large-scale, dynamic real-time systems. We argue that TUFs and utility accrual scheduling provide a more generalized, adaptive, and flexible approach. Further, new research on utility accrual scheduling have significantly advanced the state-of-the-art of that paradigm. We survey these recent advances to provide the rationale for our case.*

## 1 Introduction

The UML Profile for Schedulability, Performance, and Time or the “RT UML” Profile (RTP) [4] is considered to play a central role in the emerging paradigm of model-driven development of real-time systems [18]. The goal of model-driven development is to automatically generate a concrete code implementation from its corresponding software model.

A key aspect of RTP is the automated analysis of UML models. This allows one to quantitatively annotate the UML model of a real-time system with its QoS requirements (e.g., deadlines), use a RTP-compliant analysis tool to conduct schedulability analysis, and verify the satisfaction of QoS properties (e.g., ensure that all application dead-

lines are satisfied). Such automated analysis is facilitated by traditional real-time computing theory (e.g., schedulability analysis with RMA and DMA algorithms) [13]. Commercial tools supporting schedulability analysis with RMA and DMA algorithms are also readily available (e.g., Rapid RMA [19]).

Time-critical application QoS is expressed and handled in traditional real-time computing practice [7] using the *priority* artifact. Priorities have significant shortcomings, especially for large-scale, dynamic real-time systems, including:

(1) In general, mapping application time constraints to priorities is not tractable in practice. Doing so results in significant loss of information regarding the time constraints. This makes it difficult to reason about application timeliness behavior, thereby complicating the development of resource management algorithms that can dependably satisfy time constraints;

(2) Priority assignments are not modular for expressing urgency, because they require global knowledge of all priority assignments. Such global knowledge is often difficult to obtain because of the multiplicity of groups that are typically involved in the development of large-scale, real-time systems. Furthermore, such groups often have different limits on the amount of information that they can access (e.g., organizational boundaries, security clearances); and

(3) Urgency of an application activity is typically orthogonal to the relative importance of the activity, but a priority cannot express both (urgency and importance). This causes difficulties in reasoning about timeliness behavior and to perform resource management, especially during resource constrained situations that dynamic real-time systems are often subject to. During resource constrained situations, completing activities that are more functionally important than those which are more urgent is often desirable. Thus, a clear distinction has to be made between urgent activ-

ities and important activities. Such distinctions are difficult to specify with priorities.

The obvious solution to this problem is to provide application designers with the direct abstraction of time constraints (instead of mapping time constraints to priorities), and to use that abstraction to reason about timeliness behavior and perform resource management. The conventional real-time computing theory [14] does exactly that.

However, that theory is fundamentally limited to the deadline time constraint. In conventional real-time theory, deadlines are mapped to fixed priorities in algorithms such as RMA and DMA [13] or are directly used for scheduling and resource management in algorithms such as EDF [6].

Deadlines have the significant shortcoming that they can only be considered as either:

- (1) A binary-valued expression in the sense that a deadline is either met or not met; or
- (2) A linear-valued expression for the penalty of lateness (lateness = completion time – deadline) — i.e., the penalty of being late per unit time is constant regardless of the activity’s lateness.

Thus, deadlines still cannot make the distinction between urgency and importance — a serious limitation during resource constrained situations (as explained previously).

Another drawback of classical deadline-based scheduling [6] is the (counter-intuitive) *domino* effect that activities suffer during resource constrained situations under such algorithms [15]. The domino effect occurs as deadline-based algorithms (during resource constrained situations) favor activities that have a high likelihood for missing their deadlines over those that have a low likelihood for doing so.

The most serious drawback of deadlines is their semantic limitation: Deadlines cannot express time constraints that are non-binary and non-linear in the sense that the utility attained for activity completion *varies* (e.g., increases, decreases) with the activity completion time.

These limitations of priorities and deadlines (and that of priority-driven and deadline-driven algorithms) have been illustrated in the construction of complex, large-scale, dynamic real-time systems [2, 16].

In Section 2, we describe TUFs and the utility accrual scheduling paradigm that overcome these limitations. We discuss new advances in TUF research in Section 3. The new results significantly advance the state-of-the-art of this paradigm — hence our argument for including TUFs in RTP and thereby allowing systems engineers to reap the advantages of the paradigm. Finally, we conclude the paper in Section 4.

## 2 Time/Utility Functions and Utility Accrual Scheduling

Jensen’s time/utility functions [8] generalize the deadline constraint. A time/utility function (or TUF) specifies the utility to the system of completing an application activity as an application- or situation-specific function of when that activity completes.

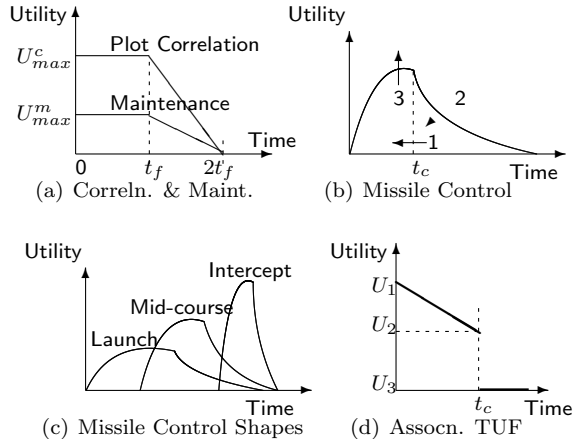


Figure 1: Example Time Constraints Specified Using TUFs: (a,b,c) GD/CMU Air Defense TUFs; and (d) MITRE/TOG AWACS Association TUF

Figure 1 shows example time constraints specified using TUFs from two complex, large-scale, dynamic real-time systems. These include: (1) a coastal air defense system built by General Dynamics (GD) and Carnegie Mellon University (CMU) [16] and (2) an AWACS surveillance tracker system built by The MITRE Corporation and The Open Group (TOG) [2].

Figures 1(a), 1(b), and 1(c) show time constraints of the air defense system. While Figure 1(a) shows TUFs of the *plot correlation* and *track maintenance* activities, Figure 1(b) shows the TUF of the *missile control* activity. Figure 1(c) shows how the TUF of the missile control activity dynamically changes. Figure 1(d) shows the TUF of the *track association* activity of the AWACS tracker.

Observe that the classical deadline constraint is a binary-valued downward “step” shaped TUF. Further, note that a TUF precisely captures the semantics of time constraints that are non-binary and non-linear. Moreover, TUFs decouple activity importance and urgency — i.e., urgency is measured on the X-axis and importance is denoted by utility on the Y-axis.

When time constraints are expressed with TUFs, the scheduling optimality criteria are based on factors that are in terms of maximizing accrued utility from those activities—e.g., maximizing the sum, or the expected sum, of the activities’ attained utilities. Such criteria are called Utility

Accrual (UA) criteria and sequencing (scheduling, dispatching) algorithms that consider UA criteria are called UA sequencing algorithms. In general, other factors may also be included in the optimality criteria, such as resource dependencies and precedence constraints.

UA scheduling was first introduced by Jensen, Locke, and Tokuda in [8] and later elaborated on in Locke’s thesis [15] and Clark’s thesis [3]. Subsequently, there have been very few efforts that built upon Locke’s and Clark’s algorithms. Most of the subsequent efforts focussed on downward step TUFs (or deadlines) and sought to improve the timeliness behavior during overload situations [9, 17]. Exceptions include [1, 20], which considered non-step TUFs.

While Locke’s algorithm allowed almost arbitrary TUF shapes, it was restricted to resource-independent activities. Clark’s algorithm, called DASA, advanced Locke’s model by allowing resource dependencies including those that arise due to shared resources subject to mutual exclusion constraints. DASA also provides assurances on timeliness behavior such as timeliness optimality during under-load situations — i.e., no deadline-misses occur during those situations. However, DASA is restricted to downward step TUFs.

Algorithms in [1, 20] allow non-step, but unimodal TUFs (i.e., those TUFs that never increase, once they decrease). Further, they are restricted to resource-independent activities.

### 3 New Advances in UA Scheduling

In spite of the generality of TUFs and UA scheduling, they have drawbacks that have apparently been impediments to their widespread usage. The most significant drawbacks include:

- (1) Lack of UA algorithms, especially for non step TUFs and that consider more general activity and resource models;
- (2) Lack of assurances on timeliness behavior of TUF/UA-scheduled systems;
- (3) Higher overhead of UA algorithms (than conventional priority-based and deadline-based algorithms); and
- (4) Lack of tool support for application designers for composing TUFs.

Recent and ongoing research in UA scheduling have overcome most of these drawbacks. Several new UA scheduling algorithms have been recently developed. These include the Generic Utility Scheduling (or GUS) algorithm [10], Combined Utility Accrual algorithm (or CUA) [22], Resource-Constrained Utility Accrual algorithm (or RUA) [23], Stochastic Utility Accrual Scheduling (or S-UA) [11], Energy-Efficient Utility

Accrual Algorithm (or EUA) [21], Resource-constrained Energy-Efficient Utility Accrual Algorithm (or ReUA) [24], and Memory-Aware Utility Accrual Scheduling Algorithm (or MSA) [5].

GUS allows almost arbitrary TUF shapes and resource dependencies. The CUA algorithm extends the TUF model with the notion of joint utility, where activities accrue utility depending upon the completion time of *other* activities. RUA allows arbitrary TUF shapes, resource dependencies, and multi-unit resource request models (GUS and DASA only allows a single-unit request model). RUA also provides the same timeliness optimality as that of DASA during under-loads.

While RUA and DASA provides assurances on timeliness behavior similar to that of hard real-time scheduling algorithms [13], like GUS and CUA, they are restricted to deterministic activity models, where activity worst-case execution times are known with absolute certainty.

This restriction is overcome in algorithms including S-UA, EUA, and ReUA. These algorithms allow stochastic activity models, where activity execution times and inter-arrival times are stochastically described. While S-UA allows non-increasing, unimodal TUFs and resource dependencies, EUA is restricted to step TUFs and no resource dependencies. However, EUA considers system-level *energy* consumption — an important concern in emerging battery-powered, embedded real-time systems. ReUA advances EUA’s model by allowing non-increasing, unimodal TUFs and resource dependencies.

S-UA, EUA, and ReUA provide statistical guarantees on timeliness behavior including probabilistically-satisfied lower bounds on utility accrued by each activity, besides maximizing the sum of the activities’ attained utilities. Being concerned with energy, EUA and ReUA also minimize system-level energy consumption.

The MSA algorithm considers another important resource of real-time systems: memory. While memory is statically allocated in traditional systems, MSA allows memory to be dynamically allocated, thereby permitting greater flexibility, and maximizes activities’ attained utilities. MSA also allows unimodal TUFs and resource dependencies and provides the same timeliness optimality as that of DASA and RUA during under-loads.

Experimental implementations of most of these new UA algorithms are also available. Implementations of the algorithms using a middleware-level, real-time POSIX-compliant framework [12] have been made. Measurements from the implementations have shown that the algorithms have reasonable overheads even at a middleware level. For example, in [12], we have observed that the overhead of a DASA scheduler is in the magnitude of

sub-milliseconds, even for a ready queue of over 60 threads on a 500MHz Pentium platform.

Another area of ongoing research is developing software tools that facilitate the design of TUFs for applications. Automatically or semi-automatically composing TUFs from application timeliness requirements is another important impediment to the usage of TUFs and UA algorithms. This is currently being addressed.

## 4 Conclusions

We conclude that the TUF/UA paradigm is important in dynamic real-time systems, and that the state of the art has recently significantly advanced. A variety of new scheduling and resource management algorithms have been recently devised. These algorithms overcome many of the traditional drawbacks of the original approaches, and expand the coverage of TUF/UA scheduling.

We believe that emerging real-time systems can significantly benefit from incorporating the TUF/UA paradigm in RTP. This will allow new real-time systems to leverage the adaptivity and flexibility of the paradigm, yet reap the advantages of traditional real-time theory (e.g., timeliness optimality during under-loads, statistical timeliness guarantees).

## References

- [1] K. Chen and P. Muhlethaler. A scheduling algorithm for tasks described by time value function. *Real-Time Systems*, 10(3):293–312, May 1996.
- [2] R. Clark et al. An adaptive, distributed airborne tracking system. In *IEEE WPDRTS*, volume 1586, pages 353–362, April 1999.
- [3] R. K. Clark. *Scheduling Dependent Real-Time Activities*. PhD thesis, Carnegie Mellon University, 1990. CMU-CS-90-155.
- [4] B. P. Douglass. *Real-Time UML: Advances in The UML for Real-Time Systems*. Addison-Wesley, 3rd edition, 2004.
- [5] S. Feizabadi, B. Ravindran, and E. D. Jensen. Msa: A memory-aware utility accrual scheduling algorithm, 2004. Under review at IEEE/ACM/IFIP CODES+ISSS. <http://www.ee.vt.edu/~realtime/submissions.html>.
- [6] W. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21:177–185, 1974.
- [7] IEEE and OpenGroup. The open group base specifications issue 6, 2001.
- [8] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time systems. In *IEEE RTSS*, pages 112–122, Dec. 1985.
- [9] G. Koren and D. Shasha. D-over: An optimal on-line scheduling algorithm for overloaded real-time systems. In *IEEE RTSS*, pages 290–299, Dec. 1992.
- [10] P. Li. *A Utility Accrual Scheduling Algorithm for Resource-Constrained Real-Time Activities*. Phd dissertation proposal, Virginia Tech, 2003. <http://www.ee.vt.edu/~realtime/li-proposal03.pdf>.
- [11] P. Li, B. Ravindran, E. D. Jensen, and H. Cho. Assurance-driven, stochastic utility accrual scheduling, 2004. To be submitted. <http://www.ee.vt.edu/~realtime/submissions.html>.
- [12] P. Li, B. Ravindran, S. Suhaib, and S. Feizabadi. A formally verified application-level framework for real-time scheduling on posix real-time operating systems, 2004. Under second review at IEEE TSE. <http://www.ee.vt.edu/~realtime/>.
- [13] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.
- [14] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, New Jersey, 2000.
- [15] C. D. Locke. *Best-Effort Decision Making for Real-Time Scheduling*. PhD thesis, Carnegie Mellon University, 1986. CMU-CS-86-134.
- [16] D. P. Maynard, S. E. Shipman, et al. An example real-time command, control, and battle management application for alpha. Technical report, CMU Computer Science Dept., December 1988. Archons Project Technical Report 88121.
- [17] D. Mosse et al. Value-density algorithm to handle transient overloads in scheduling. In *IEEE ECRTS*, pages 278–286, June 1999.
- [18] B. Selic. Model-driven development of real-time software using omg standards. In *IEEE ISORC*, pages 4–6, 2003.
- [19] Tri-Pacific Software, Inc. Rapid rma. <http://www.tripac.com/NEW/prod-fact-rrm.html>. Last updated, October 1999.
- [20] J. Wang and B. Ravindran. Time-utility function-driven switched ethernet: Packet scheduling algorithm, implementation, and feasibility analysis. *IEEE TPDS*, 15(2):119–133, February 2004.
- [21] H. Wu, B. Ravindran, and E. D. Jensen. Cpu scheduling for statistically guaranteed real-time performance and improved energy-efficiency for battery-powered embedded systems, 2004. Under review at IEEE/ACM/IFIP CODES+ISSS. <http://www.ee.vt.edu/~realtime/submissions.html>.
- [22] H. Wu, B. Ravindran, and E. D. Jensen. Utility accrual scheduling under joint utility and resource constraints. In *IEEE ISORC*, 2004. <http://www.ee.vt.edu/~realtime/>.
- [23] H. Wu, B. Ravindran, E. D. Jensen, and U. Balli. Utility accrual scheduling under arbitrary time/utility functions and multiunit resource constraints, 2004. To be submitted. <http://www.ee.vt.edu/~realtime/submissions.html>.
- [24] H. Wu, B. Ravindran, E. D. Jensen, and P. Li. Energy-efficient, utility accrual scheduling under resource constraints for mobile embedded systems, 2004. Under review at ACM EMSOFT <http://www.ee.vt.edu/~realtime/submissions.html>.