Appears in Proc. Workshop on Autonomy Control Software
Autonomous Agents '99, Seattle, May 1999

Real-Time is Not an Option

David J. Musliner
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
musliner@htc.honeywell.com

We Need Real-Time

As intelligent autonomous systems move out of research labs and into real world applications, two things become immediately apparent: first, the humans don't go away; and second, the clock never stops. The 1999 AAAI Spring Symposium on Adjustable Autonomy began addressing the first issue by investigating how autonomous systems must remain adjustable and flexible, to act as peers and collaborators with humans. In this extended abstract, I discuss the second issue: time.

In particular, real time. Not simulated time that waits for the autonomous system to finish deliberating. Not "soft real-time" that means humans get annoyed waiting for a result to show up on a screen. **Hard Real-Time.** Autonomous systems controlling mobile vehicles, spacecraft, refineries, and other major application domains face inflexible real-time deadlines. If they fail to meet those deadlines, catastrophic consequences can result: lost lives, environmental damage, millions or billions of dollars up in smoke or down the drain.

Building control systems that operate properly in hard real-time environments is always challenging; building complex autonomous control systems for these domains is still a research topic (Musliner *et al.* 1995; Garvey & Lesser 1994). This paper discusses some of the requirements that real-time domains place on systems, and presents a set of clarifying questions that help assess how well a control system design meets those requirements.

Real-Time is Not Just Fast

You've probably heard the rant before, but just in case: "hard real-time" doesn't mean "fast." In an environment that poses catastrophic threats if deadlines are not met, we must only deploy systems that are *predictably fast enough* to avoid failure. We must establish *a priori* that these systems can maintain safety despite the contingencies that may arise. Of course, *fast enough* is relative: an autonomous automobile may face millisecond deadlines to avoid collisions at traffic speeds; an autonomous greenhouse may only have to respond in seconds. The point about hard real-time domains is their intolerance of late results, at any time scale.

But Where Do We Come In?

Extensive research in real-time systems has focused on developing operating system support for predictability, formal methods for verifying system safety properties, and testing methods to validate performance guarantees (Shin & Ramanathan 1994). The key missing link is synthesis: how to automatically construct real-time control systems. And that's where we come in.

AI researchers have been synthesizing (planning) and hand-building autonomous control systems for decades. Our techniques are growing in maturity, we are finally sharing tools and methods, and now we *must* address the real-time issues to effectively deploy these systems in complex, dangerous environments.

Time is Not Just Another Resource

All systems must effectively manage their resources to "play well with others." However, real time is not, in general, negotiable; other agents don't have more that they can loan you. The environment's clock is an absolute, objective, and completely uncooperative entity. Even worse, real-time is unusual in that just thinking about the resource uses it up; we cannot simply slather on more resource management code to "handle" real-time.

Real-Time Cannot Be Retrofitted

Real-time is also one of the toughest/most interesting aspects of autonomy because it generally cannot be hacked around or retrofitted: you can't take an inherently unpredictable system and wrap it up and make the result predictable. Likewise, representation hacks won't result in a system that is simply "inelegant real-time." If a system does not meet the core requirements outlined below, it cannot provide the requisite real-time performance. Retrofitting to those requirements is not a promising avenue of research. We *must* design for real-time in the first place.

Requirements to be Real-Time

To be suited for application in a hard real time domain, a system needs to be truly predictable. This means the system must account for:

Asynchrony — The world is not synchronized or fully predictable, and real time is really continuous. Thus events and processes occurring in the world are asynchronous from the control system's perspective, and simplified synchronized models of behavior (e.g., turn-taking games, atemporal contingency plans) will not suffice. Instead, the system must be designed with the understanding that the world truly operates in parallel, and either polling or interrupts must be used to keep the control system aware of ongoing environmental changes.

The Sense/Act Gap — The assumption that sensing, selecting reactions, and performing actions takes no time is invalid in hard real-time domains. Instead, a guaranteed control system must be designed to explicitly manage asynchronous world changes that may occur between the time a particular set of sensed data is acquired and the time the system's response action can be selected and completed (Musliner, Durfee, & Shin 1994). This time gap between sensing and action may make the selected action problematic: an action that was appropriate for the sensed state may cause disaster when finally executed. For example, if an autonomous car sees a green traffic light, thinks for too long, and then moves into the intersection, the light may have turned red already. A safe real-time control system must avoid inadvertently causing or enabling failures.

Communication Time — Just as with sensed information, any communications between the real-time control system and other systems will take time, and that delay must be accounted for in any safety-critical aspects of system operation. For example, the CIRCA real-time executive relies on repeated downloads of new reactive plans (Musliner, Durfee, & Shin 1993). The time it takes to perform these downloads must be explicitly accounted for in the executive's operations, so that the communications activities do not interfere with the ongoing real-time reactive control plan.

Continuous Operation — Continuous operation is problematic for many deployed systems, from databases to telephone switches. In real-time autonomous systems, the problem is even more acute because the automatic memory management features (e.g., garbage collection) that frequently address non-real-time continuous operations (and make prototyping easier) are yet another consumer of processing time that must be carefully predicted and controlled.

If your system can "check" all of these boxes, then it is ready for prime time in mission-critical real-time applications.

Real-Time Metrics: Guiding Questions

The "real-timeness" of a system is not a single dimension, and cannot be measured by a single metric. Instead, I've formulated the following questions to provide an initial guide to assessing the real-time qualifications of a system, and its potential for providing performance guarantees. This is not a complete list, but a beginning to be built upon. It is focused on the currently-popular multilayer architectures that use a planning system to synthesize plans on the fly, while a reactive executive runs those plans robustly, filling in uncompleted details and managing most of the real world's contingencies through hand-coded reactions.

Domain — Does the domain always permit some solution that ensures safety? If not, can portions of the domain's state space or feature space be broken out to isolate those aspects for which safety cannot be guaranteed? For example, suppose a mobile robotic domain presents moving hazards that can hit and

damage the robot and also the threat of meteorites crashing down upon the robot. Using existing motion planning techniques and conservative geometric reasoning, it is possible to guarantee safety from the moving hazards (Kohout, Hendler, & Musliner 1996). But the robot may have no way to move fast enough to escape a meteorite, even if it could detect it. We can isolate the consideration of the meteorite threat and still make claims about the system's guaranteed, real-time performance against moving hazards.

Planner — Is the safety of the system dependent on the planner producing a result? If so, can the planner always find a solution, quickly enough, if one exists? Restated: is the planner complete and guaranteed time bounded? If not, is it an anytime planner that can guarantee an acceptable (if not optimal) result within a bounded time?

Executive —

Can the executive ensure that plans/reactions will be executed quickly enough to respond to threats? Is the executive's set of contingency handlers guaranteed to handle the expected contingencies within some modeled range of domain behavior, or is it just hand-coded?

Operating System — Can the OS ensure that the executive and all other required processing will be provided the necessary resources, without delay?

Hardware — Is the hardware reliable enough to support the demands of the processing requirements, under a given set of fault assumptions?

References

Garvey, A., and Lesser, V. 1994. A survey of research in deliberative real-time artificial intelligence. *Journal of Real-Time Systems* 6(3):317–347.

Kohout, R. C.; Hendler, J. A.; and Musliner, D. J. 1996. Guaranteeing safety in spatially situated agents. In *Proc. National Conf. on Artificial Intelligence*, 909–914.

Musliner, D. J.; Hendler, J. A.; Agrawala, A. K.; Durfee, E. H.; Strosnider, J. K.; and Paul, C. J. 1995. The challenges of real-time ai. *IEEE Computer* 28(1):58–66.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Trans. Systems, Man, and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1994. Predictive sufficiency and the use of stored internal state. In *Proc. AIAA/NASA Conf. on Intelligent Robots in Field, Factory, Service, and Space.*

Shin, K. G., and Ramanathan, P. 1994. Real-time computing: A new discipline of computer science and engineering. *Proceedings of the IEEE* 82(1):6–24.