The Evolution of CIRCA, a Theory-Based AI Architecture with Real-Time Performance Guarantees

David J. Musliner, Michael J. S. Pelican
Honeywell Laboratories
{david.musliner,mike.pelican}@honeywell.com

Robert P. Goldman
SIFT, LLC
rpgoldman@sift.info

Kurt D. Krebsbach
Lawrence University
kurt.krebsbach@lawrence.edu

Edmund H. Durfee
University of Michigan
durfee@umich.edu

Abstract

This paper summarizes the evolution of our research on the Cooperative Intelligent Real-Time Control Architecture (CIRCA), one of the first AI architectures designed specifically for hard-real-time environments and architecturally-enforced performance guarantees. Beginning with the objective of providing reliable real-time execution of automatically-generated plans, CIRCA research progressed to define a rigorous link between planning models, execution semantics, and performance guarantees. Formal verification techniques and automatic abstraction methods were then incorporated to improve the rigor and performance of the planning system. Multi-agent negotiation and coordination capabilities were developed to demonstrate performance guarantees spanning distributed CIRCA agents. As the limitations of CIRCA's fully-guaranteed semantics became clear, the research grew to include probabilistic versions of the problem and new solution methods. Versions of CIRCA are capable of reasoning about durative concurrent actions, exogenous events and adversaries, nondeterministic actions, and probabilistic actions and events.

Introduction

This paper provides a brief overview of the research to date that has focused on CIRCA, the Cooperative Intelligent Real-Time Control Architecture. CIRCA is one of the first fully-implemented AI planning and execution architectures that supports hard real-time performance guarantees. CIRCA research has explored a broad spectrum of related areas including reliable plan execution semantics, adversarial reasoning, heuristic search guidance, the link between formal verification and planning, meta-control of deliberation in time-constrained domains, probabilistic planning, and multi-agent planning and coordination. In this paper we show how these topics tie together and relate to the general problem of building embeddable intelligent systems that provide formal, provable properties while retaining the complex, unpredictable elements of state of the art intelligent planning and scheduling algorithms.

CIRCA's performance guarantees are constructed and enforced by components that are based on an underlying theory of reactive plan execution; the theory describes how planned reactions interact with an external world (including external sources of change). The CIRCA executive is carefully designed and implemented to enforce the semantics of the plan execution theory. Most of the CIRCA capabilities described in this paper are still active and available in the current codebase, so we describe them in the present tense, even though some were developed more than 15 years ago.

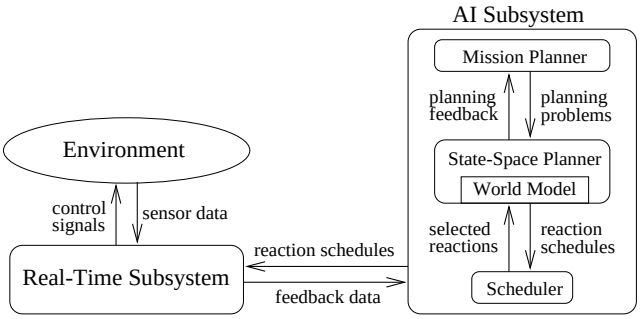

Figure 2: The CIRCA architecture combines intelligent planning and adaptation with real-time performance guarantees.

Classic CIRCA

In the beginning, we were mainly interested in automatically building real-time control plans to control things like robots in dangerous and adversarial environments. The key aspect of such domains is the potential for catastrophic failure—if the controlled agent (*e.g.*, a UAV) does not respond to some threat (*e.g.*, a surface-to-air missile launch) within a certain time limit, then it risks failing completely. So "Classic" CIRCA is designed to reason about such domains and automatically build and execute reactions that defeat such threats (Musliner, Durfee, & Shin 1993; 1995).

CIRCA Architecture

CIRCA is an autonomous, self-adaptive control architecture designed specifically for mission-critical domains. As illustrated in Figure 2, CIRCA combines on-line planning and scheduling systems in its AI Subsystem (AIS) with a very simple, very predictable real-time plan executive (the

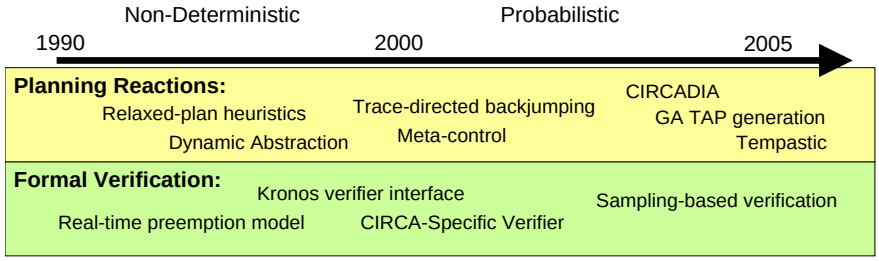

Figure 1: For over 15 years, CIRCA research has explored numerous topics related to real-time intelligent control based on theoretically-grounded performance guarantees.

Real-Time Subsystem, or RTS). CIRCA dynamically creates time-constrained reactive control plans (cyclic loops of Test-Action Pairs, or TAPs) based on its expectations about future world states and its own potential actions.

The RTS is responsible for executing TAP plans in a completely predictable fashion, so that their execution matches the model used by the AIS. The RTS meets this criterion for TAP execution because it has no other function; it simply loops over the cyclic schedule of TAPs, testing and executing them repeatedly. Even communication into and out of the RTS is encapsulated within TAPs, so that all RTS activity is scheduled explicitly.

The Planner and Scheduler, on the other hand, perform the complex, unpredictable reasoning required to develop guaranteed control plans, and the performance of these subsystems must not interfere with the RTS' predictable execution. To achieve this isolation, each control plan executed on the RTS is designed both to achieve system goals and to ensure system safety throughout the range of environmental states that are anticipated during and after the accomplishment of the goals. So the RTS keeps the system safe while the Planner and Scheduler try to build the next control plan; the planning operation is *not* constrained to meet domain deadlines.

CIRCA Theory

The full details of the underlying CIRCA theory and executive design are beyond the scope of this paper, but are available in several other publications. An intuitive treatment of the theory and executive design is available in (Musliner 1993), while (Goldman, Musliner, & Pelican 2002) provides a more formal description in terms of timed automata. Here we briefly overview the theory underlying CIRCA's performance guarantees, to set the context for describing the architectural evolution.

Unlike most planning and scheduling systems that build plans as linear or partially-ordered action sequences, CIRCA builds plans that are actually reactive *controllers*, designed to sense and react to different world states (situations) within strictly-enforced time bounds. Each TAP has a boolean test expression that distinguishes between states where a particular action is and is not to be executed, and a timing constraint specifying the maximum time allowable between TAP executions. In this context, a "state" is a particular assignment of values to features or variables that describe the world and

```
#<TAP 2>
Tests: (AND (IRU1 BROKEN)
 (OR (AND (ACTIVE_IRU NONE) (IRU2 ON))
     (AND (ACTIVE_IRU IRU1) (ENGINE ON))))
Action: select_IRU2
Max-delay: 2 seconds
```

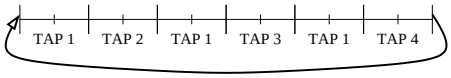

Figure 3: A sample Test-Action Pair and TAP schedule loop from a sample domain controlling redundant spacecraft inertial reference units (IRUs).

which can be sensed by the CIRCA-controlled agent. A sample TAP and an associated TAP schedule loop are shown in Figure 3. When executing a TAP, the RTS evaluates the test expression and, if it returns true, the RTS executes the corresponding action. CIRCA's Scheduler module uses the TAP timing requirements when it builds looping TAP schedules.

The world model and planning algorithm that the AIS uses to develop TAP plans are detailed in (Musliner, Durfee, & Shin 1995). For our purposes, it is sufficient to understand that the model is a modified state/transition graph in which states correspond to complete descriptions of the world (modulo some level of abstraction), and three types of transitions represent the ways the world can change. *Temporal transitions* represent time and ongoing processes. The timing behavior of a temporal transition is related to the rate of the process it represents: for example, the process of activating a spacecraft's inertial reference unit may take some minimum amount of time to complete. *Event transitions* represent occurrences outside the agent's control, while *action transitions* represent the intentional actions that can be taken by TAPs. Transitions have precondition and postcondition expressions that describe how they can link together states in the planned world model. CIRCA can control the timing behavior of action transitions by setting the timing constraints of the TAPs it builds.

To build plans, CIRCA begins with a set of goal descriptions, a set of initial world states, and a set of transition descriptions that detail the types of events, actions, and processes possible in the world. Some of the transitions are

identified as leading to a distinguished failure state, and CIRCA must build a TAP plan that makes failure unreachable while also doing its best to achieve the other goal conditions. The basic planning algorithm conducts a fairly standard type of heuristically-guided forward search with backtracking, expanding the set of reachable states by applying the uncontrollable temporal and event transitions and deciding on an action choice for each state. An action can be planned to *preempt* undesirable transitions (*e.g.*, temporal transitions to failure) by constraining the action to execute quickly enough to definitely occur before the undesirable transition could possibly occur. This notion of preemption is the core aspect of CIRCA planning that allows the system to reason about real-time performance guarantees and system safety.

World Model

Several key aspects of this real-time planning and control problem distinguish Classic CIRCA from most AI systems [1], including:

• **Exogenous Events —** Unlike most planning systems, CIRCA considers exogenous sources of change in its environment, including adversaries. Because the focus is on making real-time safety guarantees, all exogenous processes and events are assumed to possibly happen at any time they could— in fact, the system takes Murphy's Law to the extreme, expecting that anything bad that *can* happen *will* happen, at the worst possible time.

• **Time and Preemption —** CIRCA tries to build plans that prevent failures through one primary mechanism: disabling the preconditions that allow an adversary (or the environment) to cause a failure. In addition to handling the logical elements of this sort of planning, CIRCA also must ensure that its safety-preserving actions will be taken quickly enough. In other words, CIRCA has to ensure that the right action is taken at the right time, and this timing may be dictated in part by consideration of the uncontrollable environment/adversaries. Unlike most temporal planning models, CIRCA does not label state with specific times; instead, it uses a purely relative (and non-Markov) temporal model that allows the system to compactly represent continuously-executing control loops (*e.g.*, as long as you're flying, any time someone shoots a radar-guided missile at you, deploy chaff and begin evasive maneuvers).

• **Nondeterministic Actions —** CIRCA's action models can be nondeterministic, having multiple sets of postconditions. For example, the model of a `start-engine` action may either result in the engine being started or not. Combining the unique CIRCA temporal model with the notion of indexical-functional variables (Agre & Chapman 1987) and nondeterministic actions allows CIRCA to efficiently reason about looping plans (reactive controllers) without overly-precise models of system dynamics (*e.g.*, to hammer in a nail, keep hitting it until it is flush) (Musliner 1994b).

[1] See (Musliner *et al.* 1995) for an overview of Real-Time AI approaches.

• **Continuous Embedded Operation —** CIRCA is also designed to persist through changing missions that cannot be entirely pre-planned, so planning and execution occur concurrently and new plans need to be sent down to the RTS and begin execution without sacrificing the system's safety guarantees. Thus CIRCA can reason explicitly about the safe transfer of control between two different reactive controllers, implemented by different TAP schedules.

Formal Verification

CIRCA's world model is non-Markovian in the sense that the abstracted temporal model means that the path of transitions followed to reach a state can affect which transitions are possible out of that state, because of delays. For example, preempting a temporal transition to failure from one state may not disable that failure transition, but instead lead to a new state where it is still applicable; in this case the process represented by that temporal transition will have continued to run, so the safe time remaining in the new state is reduced. Naturally, this complicates the process of reasoning about the temporal model, and motivates our use of formal model checking to verify the required preemption properties that are necessary to ensure that a plan is guaranteed to avoid failure and keep the system safe (Musliner, Goldman, & Pelican 2000).

Each time the CIRCA SSP makes a heuristic decision about what action should be taken in a state, it uses a verifier to confirm that failure is not reachable and that all the planned preemptions will occur as expected. This means that the verifier will be invoked before the plan (controller) is complete. At such points we use the verifier as a conservative heuristic by treating all unplanned states as if they are "safe havens." Unplanned states are treated as absorbing states of the system, and any verification traces that enter these states are regarded as successful. Note that this process converges to a sound and complete verification when the controller synthesis process is complete.

Incremental Verification: Our earliest efforts to incorporate model checking verifiers used off-the-shelf systems such as Kronos (Yovine 1998). However, because those systems are designed for batch verification of system designs, they are tremendously inefficient when used in the inner loop of the CIRCA planning engine, completely rebuilding their verification traces as each new action decision was made. Therefore, we implemented a CIRCA-Specific Verifier (CSV) that takes advantage of several key aspects of the CIRCA planning problem and is fully incremental. The CSV system can be orders of magnitude faster than the Kronos-based approach, without sacrificing verification accuracy or precision (in fact, the CSV has a more accurate model of the executive's behavior than the atrophied Kronos interface).

Trace-Directed Backjumping: When the verifier finds that the distinguished failure state is reachable, it can return a trace illustrating a path to failure. By mapping this

failure trace onto the search stack choice points, CIRCA can pinpoint the decisions that are responsible for failure, and *backjump* to revise the most recent implicated decision. This backjumping avoids revisiting more-recent but irrelevant decisions, and can considerably improve the efficiency of the search *without sacrificing completeness*.

Heuristic Search

Despite its temporal abstractions and other advantages, the CIRCA state space is highly exponential and explodes quickly. Our efforts to manage this complexity have resulted in several research contributions:

• **Plan Graphs for non-Closed-World Models —** As with all state-space searches, heuristic guidance is critical. Fortunately, the early work on plan-graph (or "relaxed plan") heuristics occurred just as CIRCA matured. Based on McDermott's original work (McDermott 1999), we developed our own planning graph heuristic that combined the now-standard relaxation/abstraction elements (*e.g.*, ignoring negative interactions) with CIRCA-specific elements including nondeterministic outcomes and exogenous events.

• **Dynamic Abstraction Planning (DAP) —** The intuition behind DAP is simple: in some situations, certain world features are important, while in other situations those same features are not important (Goldman *et al.* 1997). By representing only the important features, DAP allows CIRCA to avoid enumerating many unique but functionally-equivalent states. DAP begins with a maximally abstract world model (only distinguishing failure and non-failure states) and incrementally adds more information to a state's representation when necessary to improve the plan. By automatically selecting the appropriate level of abstraction at each step during the planning process, DAP can significantly reduce the size of the search space.

• **Bad Smell —** Even with backjumping, the SSP might waste time repeatedly attempting to find a solution for "failed" states. Note that, because CIRCA's state space model has non-Markov temporal semantics, the action choices (including reaction timing) that may occur before a failed state can be the cause of an anticipated failure, and it may be possible to revise those earlier action decisions in a way that makes a "failed" state no longer a failure. So these states should not be completely eliminated from the search for a good plan. For this reason, we wanted to control the SSP search so that it would try to avoid states that had previously failed. We gave such states a "bad smell," so that the planner would prefer actions that avoided them wherever possible. This mechanism and its motivation are roughly analogous to aspects of Tabu search (Glover & Laguna 1993).

TAP Scheduling

The CIRCA TAP scheduling problem is fairly simple, but it has two unique aspects. First and foremost, the tasks being scheduled are automatically generated, so they are not as well-organized and optimized as human-generated tasks might be. One simple but confounding result is that TAP timing specifications do not fall on simple harmonic frequencies, so the least common multiple (LCM) of the TAP periods is generally extremely large. As a result, traditional schedulers that attempt to schedule calendars of task executions out to the LCM of the task periods (such as the Maruti scheduler (Levi *et al.* 1989)) will often be completely unable to deal with TAPs.

The second special aspect of CIRCA's scheduling problem is that, instead of a period specification, each TAP is given to the Scheduler with a specification of the maximum acceptable invocation separation. CIRCA specifies invocation separations because synchronous behavior is not necessary for the control tasks it plans. These unique constraints led us to develop novel TAP scheduling approaches that can significantly outperform simple adaptations of existing periodic-task scheduling algorithms (Musliner 1994a).

Meta Control

The Adaptive Mission Planner (AMP) is responsible for the highest-level control of a CIRCA agent, managing the agent's long-term goals and the agent's deliberation activity. The agent's long-term mission may be divided into phases, each of which requires its own safety-preserving and goal-achieving reactive plan. For example, our UCAV scenarios include missions that have phases such as ingress, attack, and egress. The ingress phase is distinguished from the attack phase both by the characteristics of the flight path (*e.g.*, a nap-of-earth stealthy approach vs. a popup maneuver very near a target) and by the expected threats (*e.g.*, the types of missile threats present at different altitudes) and goals (*e.g.*, reaching the target zone vs. deploying a weapon). The AMP reasons about long-term goals, problem structures, and approaching deadlines to decide what the near-term goals should be, and what problems the near-term reasoning should be focused on. Because the phase plans may need to be created under time pressure as a mission executes, the AMP can make tradeoffs in which mission goals and safety threats are considered in each phase. The AMP's meta-control functions intelligently allocate the CSM deliberation effort to different mission phases, solving the problems with differing levels of safety and goal-achievement depending on how much deliberation time is available (Musliner 2001; Goldman, Musliner, & Krebsbach 2003; Musliner, Goldman, & Krebsbach 2003).

We take an approximate decision-theoretic approach to the CIRCA deliberation scheduling problem: decision-theoretic, because we attempt to optimally allocate the CSM's reasoning time; approximate because full formulations of the problem are intractable and some formulations involve an infinite regress. CIRCA's earliest meta-control used coarse-grain modifications to the problems it solved to trade planning time against plan quality (Musliner 2000). In later work (Goldman, Musliner, & Krebsbach 2001), we developed a Markov Decision Process (MDP) model of the deliberation scheduling problem, controlling which of several possible problems the system should work on at any time. Since the MDP may be very large and difficult to solve,

we also presented greedy (myopic) approximations to the optimal solution. In those experiments we showed that a discounted myopic approximation technique provided good performance with very limited computational costs. We also compared the performance of the discounted greedy approximation with other strawman agents that attempt to manage deliberation using easy-to-compute heuristics.

Distributed CIRCA: Multi-Agent Performance Guarantees

We have also investigated methods for extending the real-time performance guarantees that single-agent CIRCA provides to small teams of agents. At the highest level, the AMP's primary responsibility is managing an individual agent's tasks and coordinating with other agents to achieve the overall team mission. The AMP does this by determining what tasks are its responsibilities through negotiation with other cooperating agents, and then arranging to have plans (controllers) generated to successfully address those tasks during the execution of the mission.

In this context, a team of CIRCA agents must arrange to have different agents responsible for different goals and threats, depending on their available capabilities and resources (*e.g.*, ECM equipment and weapons loadout). Using a Contract-Net-like arrangement (Smith 1977), the AMPs submit bids to handle these responsibilities. For each mission phase, the CIRCA agents must have plans, or controllers, that are custom-designed (either before or during mission execution) to execute the mission phase and make the best possible effort to achieve the goals and defeat the threats associated with the phase. When necessary, the agents can build coordinated plans that communicate at runtime to ensure real-time coordination across a team of agents.

The most critical form of coordination for real-time safety guarantees is *coordinated preemption*, in which a set of complementary reactions executed by distributed agents detect threats and take action to preempt hazardous transitions. Two key issues underlie these "You sense, I'll act" plans:

• **Planned communication —** The agents must recognize the need to explicitly communicate (both sending and receiving) at a rate fast enough to satisfy the coordinated preemption timing constraint. In our example, the sensing agent must agree not only to detect the hot spot fast enough, but also to tell the other agent about the opportunity quickly enough. Likewise, the acting agent must focus sufficient attention on "listening" for a message from the sensing agent at a high enough frequency that it can guarantee to both receive the message and act on the opportunity, all before the deadline.

• **Distributed causal links —** The distributed agents must be able to represent and reason about changes to their world that are not directly under their control, but which are predictable enough to be relied upon for a preemption guarantee. For example, in our scenario, the sensing agent must rely on the acting agent to take the appropriate action in time to guarantee that the data collection is performed in time. In complementary fashion, the acting agent must construct a plan that honors its commitment to the acting agent. If one of the agents cannot construct a plan that satisfies its commitments, it must inform the others.

Probabilistic CIRCA

As we applied CIRCA in increasingly demanding applications, it became apparent that the architecture's theoretically-strong stance on performance guarantees was not flexible enough to deal with many real-world domains. In some problems, some action choices expose an agent to more possible failure-causing events than preemptive actions can be assured to avoid. More generally, there are many domains that are too dangerous to ever ensure 100% safety; in these domains, we'd like the system to make tradeoffs between mission performance and safety criteria.

Faced with a domain that cannot be made 100% safe, Classic CIRCA fails to find a plan. And even if a 100% safe plan can be found, it may be less than satisfactory. For example, when we built a domain model for Classic CIRCA to control an aircraft in which the landing gear could fail, and there was no way to repair the landing gear or land safely with it broken, the system quickly constructed a safe plan: sit on the runway and don't take off. Unfortunately, this plan did not achieve any of the non-safety-related mission goals.

Trading off some degree of safety in order to achieve important mission-related goals requires that CIRCA make careful choices about which potential failures it is most safe not to be prepared for. Probabilistic CIRCA does this by trimming away enough of the most unlikely transitions to failure that the remaining ones can be guaranteed preempted. When developing this variation of CIRCA, we developed a variety of techniques for estimating probabilities of reaching states and traversing transitions to failure; the non-Markovian aspects of CIRCA in particular make it challenging to assess the probabilities of transitions to failure that persist across sequences of states.

The cumulative probabilities of the transitions that have been trimmed to achieve a subset of the space that can be safely controlled indicate the degree of risk that would be incurred should the control plan be followed. With this information, informed tradeoffs between risk and mission goals are possible (Atkins, Durfee, & Shin 1996).

This in turn raises the question of what can or should happen if one of the trimmed transitions actually occurs, putting the agent into a state that its control plan isn't prepared to handle. Our extensions consider all of the states just outside of the control envelope and develop tests to detect when such a state has been reached. A TAP is formed with this test, where the corresponding action involves replacing the current TAP schedule with another one that is intended to at least maintain safety. For example, in our aircraft domain, if the landing-gear failure was trimmed from the initial control plan, the TAP detects this failure and implements a control plan for circling the airport, with the expectation that this will buy time for the AIS to use in formulating an appropriate control plan for recovering from this situation. Al-

ternatively, the appropriate control plan for this contingency might have been developed ahead of time, in which case it would be swapped in immediately (Atkins, Durfee, & Shin 1997).

GSMDPs: To capture a more powerful and precise notion of probabilistic guarantees, we began exploring a modified CIRCA world model in which the fixed worst-case delays associated with transitions were replaced by probability distributions of possible delays. The idea then is to build plans that allow a certain level of safety risk, as long as the overall probability of failure remains below some specified threshold. It turns out that the resulting model is a Generalized Semi-Markov model, and is thus extremely intractable. Even assessing whether a particular controller meets the safety threshold is not analytically computable. So first we developed a sampling-based approach to probabilistic verification, deriving formal bounds on how many simulated plan executions (samples) had to be generated to ensure, with a certain level of confidence, that a plan met the desired safety threshold (Younes & Musliner 2002). We then explored various approaches to extending the CIRCA CSM to build plans in this probabilistic model, as well as other techniques (Younes, Musliner, & Simmons 2003).

Conclusion and Future Directions

In summary, the CIRCA architecture combines soft-real-time reaction planning and scheduling with hard-real-time reactive plan execution, all tailored to realistic non-closed-world domains. The architecture has been extended in several major directions, including formal verification of performance guarantees, multi-agent team guarantees, and probabilistic plans with well-understood risk/reward tradeoffs. Ongoing challenges include improving the scalability of the planning algorithms and extending the representational power of the domain description language to include task hierarchy and decomposition information.

References

Agre, P. E., and Chapman, D. 1987. Pengi: An implementation of a theory of activity. In *Proc. National Conf. on Artificial Intelligence*, 268–272. Morgan Kaufmann.

Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 1996. Plan development using local probabilistic models. In *UAI*, 49–56.

Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 1997. Detecting and reacting to unplanned-for world states. In *Proc. National Conf. on Artificial Intelligence*, 571–576.

Glover, F., and Laguna, M. 1993. Tabu search. In Reeves, C., ed., *Modern Heuristic Techniques for Combinatorial Problems*. Oxford, England: Blackwell Scientific Publishing.

Goldman, R. P.; Musliner, D. J.; Krebsbach, K. D.; and Boddy, M. S. 1997. Dynamic abstraction planning. In *Proc. National Conf. on Artificial Intelligence*, 680–686.

Goldman, R. P.; Musliner, D. J.; and Krebsbach, K. D. 2001. Managing online self-adaptation in real-time environments. In *Proc. Second International Workshop on Self Adaptive Software*.

Goldman, R. P.; Musliner, D. J.; and Krebsbach, K. D. 2003. Managing online self-adaptation in real-time environments. In *Lecture Notes in Computer Science*, volume 2614. Springer-Verlag. 6–23.

Goldman, R. P.; Musliner, D. J.; and Pelican, M. J. 2002. Exploiting implicit representations in timed automaton verification for controller synthesis. In *Proceedings of the 2002 Hybrid Systems: Computation and Control Workshop*.

Levi, S. T.; Tripathi, S. K.; Carson, S. D.; and Agrawala, A. K. 1989. The MARUTI hard real-time operating system. *ACM Operating System Review* 23(3).

McDermott, D. 1999. Using regression-match graph to control search in planning. *Artificial Intelligence* 109(1-2):111–159.

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. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.

Musliner, D. J.; Goldman, R. P.; and Krebsbach, K. D. 2003. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In *Proc. Third International Workshop on Self Adaptive Software*.

Musliner, D. J.; Goldman, R. P.; and Pelican, M. J. 2000. Using model checking to guarantee safety in automatically-synthesized real-time controllers. In *Proc. IEEE Int'l Conf. on Robotics and Automation*.

Musliner, D. J. 1993. *CIRCA: The Cooperative Intelligent Real-Time Control Architecture*. Ph.D. Dissertation, University of Michigan, Ann Arbor. Available as University of Maryland Computer Science Technical Report CS-TR-3157.

Musliner, D. J. 1994a. Scheduling issues arising from automated real-time system design. Technical Report CS-TR-3364, UMIACS-TR-94-118, University of Maryland Department of Computer Science.

Musliner, D. J. 1994b. Using abstraction and nondeterminism to plan reaction loops. In *Proc. National Conf. on Artificial Intelligence*, 1036–1041.

Musliner, D. J. 2000. Imposing real-time constraints on self-adaptive controller synthesis. In *Proc. Int'l Workshop on Self-Adaptive Software*.

Musliner, D. J. 2001. Imposing real-time constraints on self-adaptive controller synthesis. In *Lecture Notes in Computer Science*, number 1936. Springer-Verlag.

Smith, R. 1977. The contract net: A formalism for the control of distributed problem solving. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, volume 1, 472.

Younes, H. L., and Musliner, D. J. 2002. Probabilistic plan verification through acceptance sampling. In *Proc. AIPS-02 Workshop on Planning via Model Checking*, 81–88.

Younes, H. L. S.; Musliner, D. J.; and Simmons, R. G. 2003. A framework for planning in continuous-time stochastic domains. In *Proc. Int'l Conf. on Automated Planning and Scheduling*, 195–204.

Yovine, S. 1998. Model-checking timed automata. In Rozenberg, G., and Vaandrager, F., eds., *Embedded Systems*. Springer Verlag.