

A Hybrid Systems Approach to Computer-Aided Control Engineering

Wolf Kohn, John James, Anil Nerode, Karan Harbison, and Ashok Agrawala*

In this article we provide a tutorial-style overview of ideas for a new Computer-Aided Control Engineering (CACE) environment. We discuss these ideas through repetition of two themes. The first theme is that the CACE environments should support a thread from requirements development to implementation. While some CACE environments come close to this today, none provides the end-to-end thread we envision. The second theme is that the environment should support the automatic generation of automata that simultaneously comply with discrete and continuous constraints. Success in this theme would reduce the need to conduct extensive simulations and build prototypes in order to deliver and incrementally change computer-controlled systems. An example shows how such an environment can assist in addressing a recurring problem in manufacturing systems: repair of production schedules required by changes in logical or continuum constraints on factory operation. We argue that a declarative, hybrid-systems approach to off-line design and on-line generation of reactive control is needed for achieving synchronization, scalability, integration, and incremental construction of large-scale, computer-controlled systems.

Introduction

Recent results of a joint IFAC-IEEE study on challenges of computer science in industrial application of control emphasize the need for improvements in CACE tools, including investigating use of hybrid systems theory [1]. Hybrid systems are those systems described by compositions of logical models and evolution models. We provide a tutorial-style overview concerning how recent results in establishing mathematical foundations for hybrid systems can be applied to provide improved tools for implementation of computer-controlled machines.

Current CACE environments [such as [24, 25, 26, 27, 28, 29]] provide dramatically improved methods for entering both logical and continuum models of

*A version of this paper was presented at the IEEE/IFAC Joint Symposium on CACSD, Tucson, March 1994. Kohn and James are with Intermetrics Inc. Their work is partially sponsored by SDIO/IST grant DAAH04-93-C-0113. Nerode is with the Mathematical Sciences Institute. His work is supported in part by Army Research Office contract DAAL03-91-C-0024 and by DARPA-U.S. Army AMMCCOM contract DAAA21-92-C-0013 to ORA Corp. and by SDIO/IST grant DAAH04-93-C-0113. Harbison is with the Automation & Robotics Research Institute at the University of Texas at Arlington. Agrawala is with the University of Maryland Department of Computer Science.

complex systems and conducting simulation experiments with closed-loop and open-loop system behavior. In addition, some [such as [28, 29]] generate control programs based on those experiments. However, while this approach works well for small-scale systems, the experimental approach for integration of logical and continuum models is a major source of increased costs and decreased reliability for large-scale systems. The importance of achieving a synthesis capability for design and implementation of computer-controlled machines was underscored by John Cassidy in his plenary talk at the 1993 American Control Conference (ACC) when he said “77% of the software of control systems is for implementation of logic and scheduling and 23% of the software is for implementation of control algorithms. We still do not have a methodology for integrating logic and control algorithms.”¹ He emphasized the importance of creating such a methodology for integration of logic and scheduling software with control programs, the former based on set-based techniques and the latter based on differential equation theory, control systems theory and analysis techniques. We provide ideas for resolution of this dilemma by building a thread from user requirements to system implementation and thoughts on tools to aid in application of this approach.

To provide a more flexible means of supporting user input of both logical and continuum constraints we discuss a recently-developed methodology for formal statements (specification) of system requirements [22]. This scenario-based requirements analysis method has been applied to develop and update requirements for several complex systems engineering problems.

To provide a means of analysis, design, and on-line reactive control of the resulting hybrid system models, we discuss a generic multiple-agent hybrid control architecture (MAHCA). MAHCA is based on extensions to the theory of *relaxed variational control*. The basic theory was developed by L.C. Young in the 1970s [10] but implementation of the approach has been computationally infeasible. New results support computational feasibility of relaxed variational control [2, 3, 4, 5, 6, 7, 8, 9]. This new theory extends the concepts, principles, and algorithms of declarative control theory and merges them with principles of concurrent computing and dynamical hybrid systems in a formal framework based on variational models. The result is an optimization-based approach for achieving near-optimal, closed-loop performance of systems described by compositions of logical models and continuum models. The fundamental result is a constructive algorithm for realization, at each update interval, of an executable automaton that simultaneously complies with logical and continuous-time constraints. Demonstrations of both single-agent and multiple-agent versions of declarative control architectures are currently underway [42].

To provide a means of executing the programs (automata) generated by MAHCA, we discuss creation of a link with the Maruti distributed, real-time operating system [43]. Maruti’s formal distributive executive is compatible with the equational nature of MAHCA.

¹John Cassidy, director of research, United Technologies Corp., plenary speech, “Control Technology and the 21st Century,” American Control Conference, San Francisco, CA, June 2, 1993.

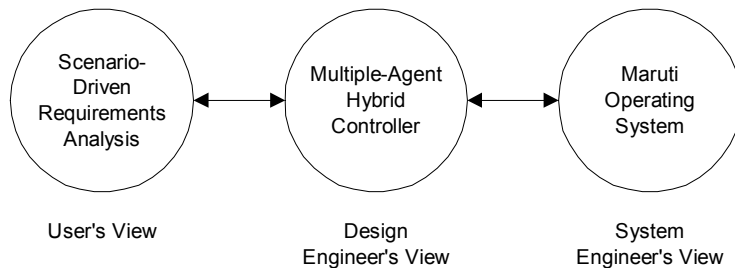


Figure 1: Multiple views of control engineering problem

Realization of our approach will provide an *end-to-end methodology*: requirements formulation and analysis, modeling, design, simulation, performance analysis, and implementation. This approach supports multiple views of the control engineering problem, corresponding to the multiple disciplines involved in the process (see Fig. 1). The tools described are available individually but have not been integrated into a hybrid system CACE environment. We believe construction of such an environment will provide a path to realization of high-safety, high-assurance, high-reliability computer-controlled devices [9]. Other research efforts [such as [26, 35, 36, 37, 38, 39, 40]] are ongoing concerning development of alternative approaches for analyzing, designing, and implementing hybrid systems. Advances in these areas would support creation of analogous hybrid-systems CACE environments.

The article is organized into six sections. Section 2 provides the overall system architecture. Section 3 summarizes the scenario-based requirements capture methodology. Section 4 provides an outline of the multiple-agent architecture and clarification of hybrid systems ideas using a two-agent example of implementing on-line planning, scheduling, and control for manufacturing. Section 5 provides an outline of the Maruti implementation. Section 6 provides a summary of the generic CACE architecture.

CACE Architecture

One of the basic problems encountered in the development of control systems is the necessity of achieving consensus concerning system requirements and the ability of the system to meet those requirements. A consistent source of disagreement is the fact that multiple disciplines are involved in the process and long time delays may be encountered between receipt of an initial system specification by the design engineer and user testing of the delivered product. Fig. 1 is intended to convey the intent that multiple views of the system model are needed to support the different responsibilities (and interests) of the players in the solution of the control engineering problem. We expect that support for realization of these multiple views would be achieved by a reference architec-

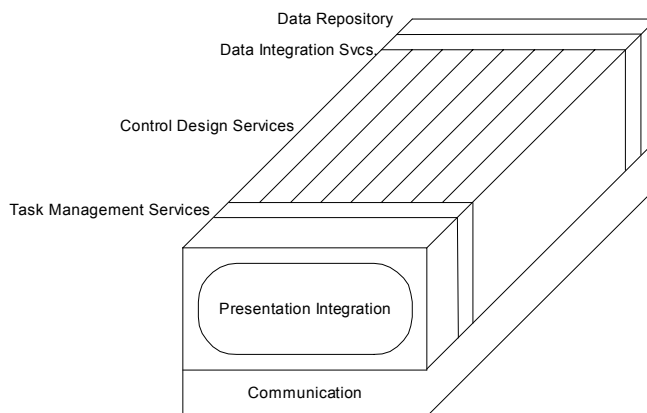


Figure 2: NIST-ECMA software engineering environment reference architecture

ture similar to the National Institute of Standards and Technology-European Computer Manufacturers Association (NIST-ECMA) software engineering environments reference architecture (see Fig. 2), which is generally followed in current developments [24, 25, 26, 27, 28].

Scenario-Based Requirements Analysis

Our use of scenario-based requirements analysis supports modeling the system at an appropriate level of abstraction. Requirements capture is a continuing challenge in the development of computer-controlled systems that assist humans in the control of complex processes. Rigorous definition of requirements, which supports ease of implementation, conflicts with narrative description of requirements, which supports ease of understanding and validation that the implemented system meets user requirements.

The use of scenarios is an easily understood and readily modified way of discussing complex processes. The Advanced Research Projects Agency (ARPA) Domain-Specific Software Architectures (DSSA) program [12, 13] has recently emphasized scenario-driven, object-oriented requirements analysis (SDOORA) as a reasonable approach to performing domain analysis and creating a domain architecture (see Fig. 3). SDOORA [22] provides a structured approach to capturing the way individuals use scenarios to organize individual and team involvement in controlling complex processes and expressing these results in a formal manner. The creation of a scenario as a set of related objects is accomplished in three phases:

1. Preliminary Requirements Analysis Phase
2. Object-Oriented Requirements Analysis Phase

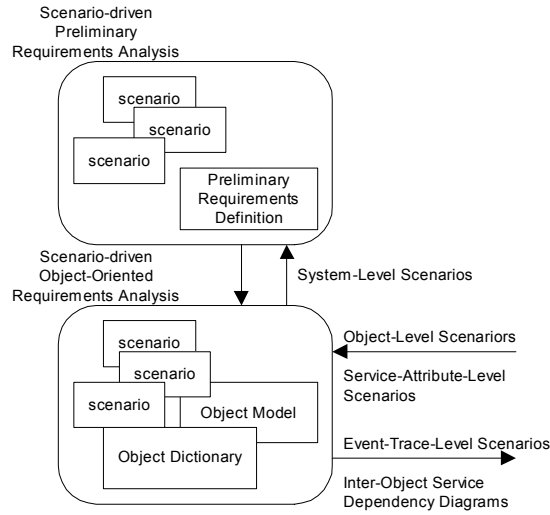


Figure 3: Scenario-driven, object-oriented requirements analysis method

3. Creation of the Object Dictionary

While this methodology has been used to capture composite design requirements and is being used in the engineering design process for complex, computer-controlled systems (such as an autonomous land vehicle), the requirements are currently not captured in a format that is compatible with MAHCA.

The Multiple-Agent Hybrid Control Architecture (MAHCA)

Control Architecture Components—A Network of Agents

The CACE environment of Fig. 2 will use control design services applications to provide the multiple-disciplinary views of Fig. 1 to the different groups involved in the control design process. The mechanism used to synchronize both the logical and continuum values of control design parameters is hybrid system theory [2, 3, 4, 5, 6, 7, 8, 9]. The reference architecture that provides the basis for integration of the multiple views is the Multiple-Agent Hybrid Control Architecture (See Figs. 4, 5, and 6) [7, 8, 9, 15].

MAHCA consists of a variable network (Fig. 5) of control agents (Fig. 4). Each agent is formally structured; that is, its behavior is characterized by a *model* encoded in logic clauses in a hierarchy. This model includes constraints that enable both the detection of logic failure (i.e., the model and the behavior observed by the agent are not in agreement) and the activation of structural adaptation processes as a reaction to the failure. Logic failures are triggered

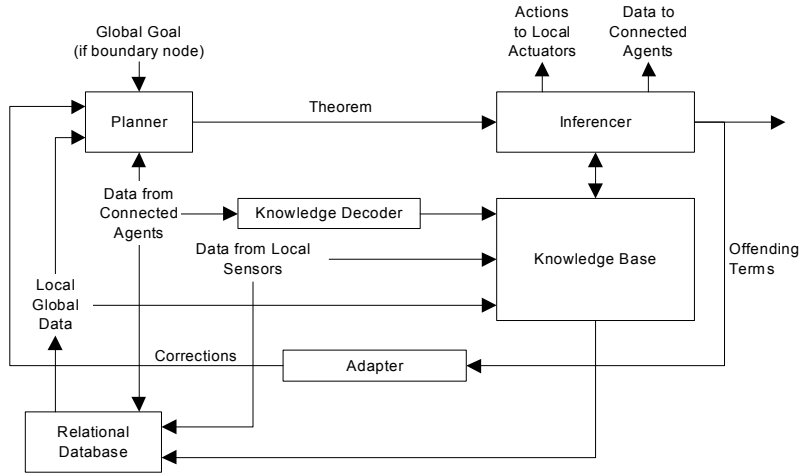


Figure 4: Control agent

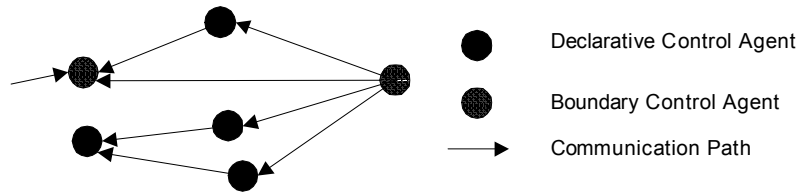


Figure 5: Network of cooperating control agents

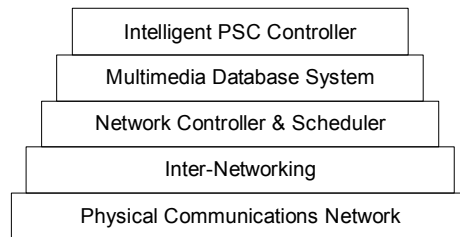


Figure 6: Hierarchical organization of the network

by predictable and unpredictable events affecting the behavior of the processes under control of the agent. Structural adaptation is accomplished by the modification of the logic clauses, according to a set of modification rules, or by the creation or deletion of agents in the network. The appropriate reaction to a failure is logically *deduced* from a dynamic knowledge base which contains relations encoding the operational parameters characterizing the process under control.

Each agent acts to control activities at a node in the network and can also structurally adapt in accordance with pre-specified or deduced constraints. Agent adaptation is achieved by creating derived clauses as logical consequents of encoded ones, by combining rules, by relaxing them, by deactivating them and combinations of the above. Thus, our architecture provides a knowledge-based, formal framework for deducing on-line feedback control laws and reactive strategies for processes involving multiple decision makers (or agents). Our architecture provides a unique, generic approach for achieving closed-loop control of distributed, interactive processes characteristic of large-scale systems. Namely, each agent has a view of the network and the near-optimum model is the chattering combination of the models [6, 7, 8, 9]. Furthermore, maintenance of consistency among the different views is achieved through reactive update of logic and continuum constraints shared with an equivalent agent. Also, the structural adaptation characteristics of the architecture explicitly addresses the need for controlling the variable quality of service (QOS) classes and performance parameters in the unavoidable presence of sensory and knowledge uncertainty. The single-agent architecture is being implemented as a low-level controller of the angular position of the tip of a flexible rod, and a demonstration of the multiple-agent controller has been implemented as a distributed (notional) controller of multiple weapons systems engaging multiple targets [42].

While fast processes are locally controlled by individual agents, more slowly-varying processes and higher-level logic are controlled through interactions of the network of agents. These control interactions governed by the user-specified models are achieved through a layered set of hardware and software systems (as in the Open System Interconnect layered model) shown in Fig. 6. MAHCA top-level control is achieved through interaction of individual agents and is implicit without umpire. Thus, the Intelligent PSC Controller shown at the top of Fig. 6 is for an individual agent in the distributed architecture. The manufacturing planning scheduling and control (PSC) problem [15] will be developed as an example below. Structural adaptation of the network of agents is accomplished by the modification of the logic clauses, according to a set of modification rules, or by the creation or deletion of agents in the network. A significant consequence of this structural adaptation approach is the ability to support on-line modification of a hierarchical partitioning of the manufacturing process such that, from the highest level of abstraction, decomposition of manufacturing tasks into a *continuum hierarchy* is supported [9, 15].

Structural adaptation from a given, fixed hierarchy of tasks occurs in response to a logic failure to deduce actions consistent with an agent's knowledge base, inputs from other agents in the network, and values of current sensor readings. The appropriate reaction to a failure is logically deduced from a dynamic

knowledge base that contains relations encoding the operational parameters characterizing the process under control.

How MAHCA Works

A control agent (Fig. 4) is composed of five functional elements—a *Knowledge Base*, a *Theorem Planner*, an *Inferencer*, a *Knowledge Decoder*, and an *Adapter*. These five components act together to achieve an epsilon-optimal [8, 9] solution to the local-agent hybrid control problem. The sequence of steps leading to *generation* of programs that comply with the *current* specifications and parameter values are:

1. The original problem is reformulated as a calculus-of-variations problem on a carrier manifold of system states. The carrier manifold is the combined simulation model of the network and the simulation models at the nodes, a manifold on which the (evolution of) state trajectories occur. We are to find control functions of the state of the system for the global and local problems that minimize a non-negative cost function on state trajectories whose minimization perfectly achieves all the required goals of the manufacturing planning, scheduling, and control problem.

2. One replaces the variational problem with a convex problem by convexifying, with respect to u , the state rate, the Lagrangian $L(x, u)$ that is being minimized. The convexified problem has a solution that is a measure-valued (weak, or L.C. Young) solution to the original problem. This is a chattering control that chatters appropriately between the local minima of the original problem so as to achieve close to the global minimum of the original problem. This solution, however, is only abstract and gives local and global control functions of *time*.

3. To get control functions of *state* instead, we convert the convexified problem to the appropriate Hamilton-Jacobi-Bellman equation form. An “e-solution” of this equation for the appropriate boundary conditions gives valid infinitesimal transformations on the state space representing the generators of feedback controls, i.e., control functions of state, not time, the solutions dual to the original ones. The weak solution obtained is approximated to by global and local “chattering control” programs.

4. The controls that are possible at a given state are a cone in the tangent plane, and move with the tangent plane. If one follows the optimal control as one moves in state, the near optimal controls needed are algebraically represented by the Christoffel systems of an affine connection, a recent result of Kohn-Nerode-James [9]. The Christoffel symbol representation gives the real-time computation of the global and local automata, or control programs, needed to govern the communications network and the approximations at nodes in order to meet the prescribed goal. The global program takes responsibility for local updates in real time. Required dynamics of the global system is achieved without central control (umpire) for the distributed system through enforcing global continuity conditions at each node.

To our knowledge, the Kohn-Nerode approach to hybrid systems is the only

theory to both (1) provide a solid mathematical foundation to unify logical and evolution models and (2) be computationally feasible. The methodology, while promising, is very new, and extensive research is required to understand both the most effective ways to construct the unified models and to build the necessary interfaces to existing systems. The rapid acceptance of the theory by major research institutions attests to its relevance to fundamental issues in several disciplines.

Agent Knowledge Base

Insufficient space is available to discuss the actions performed by each of the agent components. However, in this section we provide some detail of the knowledge base component since it is the component that supports declaration of logic and continuum constraints. The Knowledge Base consists of a set of equational first-order logic clauses with second-order extensions. The syntax of clauses is similar to the Prolog language. Each clause is of the form

$$\text{Head} < \text{Body} \tag{1}$$

where Head is a functional form, $p(x_1, \dots, x_n)$, taking values in the binary set $[true, false]$ with x_1, x_2, \dots, x_n variables or parameters in the domain D of the controller. The symbol $<$ stands for logical implication. The variables appearing in the clause head are assumed to be universally quantified. This form of a declaration is commonly called a Horn clause.

The Body of a clause is a conjunction of one or more logical terms,

$$e_1 \wedge e_2 \wedge \dots \wedge e_m \tag{2}$$

where \wedge is the logical *and*. Each term in (2) is a relational form. A relational form is one of the following: an equational form, an inequational form, a covering form, or a clause head. The logical value of each of these forms is either *true* or *false*. A relational form e_i is true for precisely the set of tuples of values S_i of the domain taken by the variables where the relational form is satisfied, and is false for the *complement* of that set.

The logical interpretation of (1) and (2) is that the Head is *true* if the conjunction of the terms of Body are jointly *true* for instances of the variables in the clause head. These clauses, which are application-dependent, encode the requirements on the closed-loop behavior of the system under control. In fact the closed-loop behavior, which is defined in [15] in terms of a variational formulation, is characterized by continuous curves with values in the domain of the variables in a clause head. This continuity condition is central because it is equivalent to requiring the system to look for actions that make the closed-loop behavior satisfy the requirements.

The denotational semantics of each clause in the knowledge base [3] (see Fig. 7) is one of the following: a conservation principle, an invariance principle, or a constraint principle.

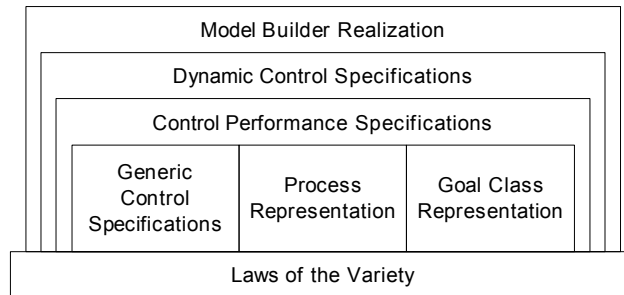


Figure 7: Knowledge base organization

Conservation principles are one or more clauses about the balance of a particular process in the dynamics of the system, the goals, or the computational resources. For instance, in the modeling of a manufacturing shop floor, a conservation clause might describe the material balance in the manufacturing process as viewed by each of two agents relative to the current list of outstanding orders. The conservation principle is central in characterizing the dynamics of manufacturing processes since it provides a general formulation of the relationships between producers and consumers.

Invariance principles are one or more clauses establishing constants of motion in a general sense. These principles include stationarity, which plays a pivotal role in the formulation of the theorems proved by the architecture, and geodesics. The importance of invariance principles lies in the fact that they provide references for the detection of unexpected events. For example, in an adiabatic chemical process, the enthalpy is constant, under normal operating conditions. An equational clause that states this invariance has a ground form that is constant; deviation from this value represents deviation from normality.

Constraint principles are clauses representing the engineering limits of actuators or sensors and, most importantly, behavioral policies. For instance, in a chemical reactor, the characteristics of the speed of response of the values controlling the input products or temperature are given by empirical graphs (e.g. pressure vs. velocity) and a strategy for the interpolation of the data in the graphs. The clause database is organized in a nested hierarchical structure as illustrated in Fig. 7. The bottom of this hierarchy contains the equations that characterize the algebraic structure on which the terms of relational forms are defined: an algebraic variety. At the next level of the hierarchy, three types of clauses are stored: Generic Control Specifications, Plant Representation, and Goal Class Representations. The generic control specifications are clauses expressing general desired behavior of the system. They include statements about stability, complexity, and robustness that are generic to the class of declarative rational controllers. These specifications are written by constructing clauses that combine laws of the kind that use the Horn clause format described earlier. The Plant Representation is given by clauses characterizing the dynamic be-

havior and structure of the plant, which includes sensors and actuators. These clauses are written as conservation principles for the dynamic behavior and as invariance principles for the structure. As for the generic control specifications, they are constructed by combining variety laws in the equational Horn clause format. The next level of the hierarchy involves the Control Performance Specifications. These are typically problem-dependent criteria and constraints. They are written in equational Horn clause format. They include generic constraints such as speed and time of response, and qualitative properties of state trajectories [2, 3, 4]. Dynamic control specifications are equational Horn clauses whose bodies are modified as functions of the sensor and goal commands. Finally, model builder realization clauses constitute a recipe for building a procedural model for variable instantiation and theorem proving.

Statement of the Central Problem of Feedback Control in Terms of the MAHCA Framework

Epsilon optimality for hybrid systems is the result of: (1) explicit statements about the qualitative and quantitative models of system components, system inputs, and disturbance uncertainties, (2) the desired qualitative and quantitative closeness constraints of the closed-loop system, and (3) the generation of programs (automata) to force closed-loop compliance with those constraints [8, 9]. The problem we discuss is an on-line modification of the evolution of the system:

$$\dot{X}(t) = f(X(t), u(t), t), \quad (3)$$

where

$$t \in I, \text{ an interval of real line, } M, \text{ and } U \text{ are manifolds,}$$

$$u : I \rightarrow U \text{ is measurable, and } X : I \rightarrow M.$$

Our goal is to design a control law

$$u(t) = \gamma(X(t), t), \quad (4)$$

which alters the closed-loop evolution

$$\dot{X}(t) = F(X(t), t) = f(X(t), \gamma(X(t), t), t), \quad (5)$$

with

$$\dot{X}(t) : I \rightarrow TM_x, \text{ Tangent space at } x$$

$$F : M \times I \rightarrow TM, \text{ Tangent bundle}$$

so that closed-loop qualitative and quantitative performance requirements are met.

Explicit statement of qualitative and quantitative models and disturbance uncertainties

We model the qualitative and quantitative behavior of the system using explicit rules which constrict the evolution of the system. The knowledge base of the controller is a composition of external rules of the form:

$$p_i(G, S, A, E) \text{ if } e_1^i(x_1, \dots, x_n) \wedge \dots \wedge e_m^i(x_1, \dots, x_n) \\ \wedge \text{unify}(x_1, \dots, x_n; G, S, A, E), \quad (6)$$

where

G : Goal variables, S : Sensor variables,

A : Actuator variables, E : Evaluation variables

and internal rules of the form:

$$q_i(y_1, \dots, y_m) \\ \text{if } e_1^i(y_1, \dots, y_m) \wedge \dots \wedge e_m^i(y_1, \dots, y_m), \\ \text{where } e_i^j(y_1, \dots, y_m) \text{ is of the form:}$$

$$w_i^j(y_1, \dots, y_m) = v_i^j(y_1, \dots, y_m) \text{ or} \\ w_i^j(y_1, \dots, y_m) \neq v_i^j(y_1, \dots, y_m) \text{ or} \\ w_i^j(y_1, \dots, y_m) \leq v_i^j(y_1, \dots, y_m) \text{ or} \\ p_k(G, S, A, E) \text{ or} \\ q_k(y_1, \dots, y_m) \quad (7)$$

with the syntax of the clauses similar to the Prolog language (Eqs. (1) and (2)).

The rules are written to capture both qualitative and quantitative constraints on the system operation. Enterprise process dynamics are modeled in terms of the conservation of the flow of one or more enterprise process variables through a distributed *logic communication network* of control agents. *Conservation principles* are one or more clauses about balance of a particular process in the dynamics of the system, the system goals, or the computational resources of the system. For example, in transportation networks, the flow of the sum of vehicles is conserved; in communication networks the flow of unsatisfied demand for network services is conserved; and in manufacturing the flow of product through the factory floor creates a demand for services at work cells (demand for short) which is synchronized through the conservation of the flow of demand through the logic communication network. For communications enterprises, the service demand represents the demand for connections, bandwidth, or higher quality of service. For manufacturing enterprises the flow of unsatisfied demand between producers and consumers can be used to generate actions to implement factory planning, scheduling, and control functions [15]. For transportation systems the flow of vehicles and the (opposite direction) flow of highway voids can be used to generate programs for traffic control [41].

As an example, consider a clause representing conservation of computational resources:

$$\begin{aligned}
& \text{comp}(\text{Load}, \text{Process Op_count}, \text{Limit}) \\
& \quad < \text{process}(\text{process_count}) \\
& \quad \wedge \text{process_count} \cdot \text{Load1} - \text{Op_count Load} \\
& \quad \quad \wedge \text{Load1 Limit} \\
& \quad \quad \wedge \text{comp}(\text{Load1}, \text{Process}, \text{Op_count}, \text{Limit})
\end{aligned}$$

where “Load” corresponds to the current computational burden, measured in VIPS (Variable Instantiations Per Second), Process is a clause considered for execution, and Op_count is the current number of terms in process.

Conservation principles always involve recursion, whose scope is not necessarily a single clause as in the example above, but with chaining throughout several clauses.

The conservation of unsatisfied demand for manufacturing

In planning, scheduling, and control of workpieces through a discrete-manufacturing shop floor, a conservation clause describes the material balance in the manufacturing process as viewed by each agent relative to the current list of outstanding orders (our example in Section 4.5 is based on two agent).

The general structure of the conservation rule for agent i is as follows:

$$\begin{aligned}
& \text{conservation_unsatisfied_demand}(G_i, U_i, X_i, A_i, \\
& \quad \quad \quad S_i, [W_{i,j}], T, DELTA) < \\
& U_i(t) = G_i(t) - \int_{-\infty}^t r_{i,i}(t, \sigma, X_i(\sigma), A_i(\sigma)) U_i(\sigma) d\sigma + V_i(t) \\
& \quad \quad \quad \wedge V_i(t) = \sum_{\substack{N_i \\ j, j \neq i}} W_{i,j}(t) \\
& \quad \quad \quad \wedge T_i = T + DELTA \\
& \wedge \text{conservation_unsatisfied_demand}(G_i, U_i, X_i, A_i, \\
& \quad \quad \quad S_i, [W_{i,j}], T_1, DELTA) \quad (8)
\end{aligned}$$

In (8), the first equational term relates the unsatisfied demand $U_i(t)$ for agent i at the current time to the unsatisfied demand $U_i(s)$, $s < t$ in the past and the net current demand $V_i(t)$ of the other agents connected to i ; on i . In this term, $r_{i,i}$ is the rate of production function of agent i on i . G_i , A_i , and S_i are respectively the goals, actions and sensor values of agent i . This function depends on the time, state of the agent X_i , and current action A_i . The second term in the body of the clause determines the net demand on agent i due to all other agents connected to it. The third term updates the time horizon of validity of the current instance of the clause. Finally, the last term implements the recursion.

The optimization problem

A state-space oriented assertion of Bellman's optimality principle is that an optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. For hybrid systems, application of this principle leads to the Hamilton, Jacobi, Bellman (HJB) relaxed variational form [8]:

$$\begin{aligned}
 V_i(Y, \tau) &= \inf_{\alpha_i} \int_{\tau} L_i(\Psi_i(\tau, Y), v_i|_p(G_i(\tau, p))) d\alpha_i(p, d\tau) \\
 \frac{\partial V_i}{\partial \tau} &= \inf_{\alpha_i} H\left(Y, \frac{\partial V_i}{\partial Y}, \alpha_i\right) \\
 Y(t) &= p \\
 \tau &\in [t, t + \Delta)
 \end{aligned} \tag{9}$$

where p is a point in the carrier manifold, V_i are tangent vectors at each point and a_i are coefficients of the tangent vectors.

Kohn-Nerode-James optimality result

Consider vector fields in the tange bundle of infinitesimal actions (see Fig. 8) and the tangent at point x in the carrier manifold.

What happens when moving to point $x + Dx$?

In ordinary calculus we would use the directional derivative to answer the question. In the mathematics of manifolds, the Levi Civita theorem provides a mechanism for connecting the tangent at x to the tangent at $x + Dx$.

In the Kohn-Nerode formulation for hybrid systems, there is no distance metric but there is a concept of closeness based on the Kohn-Nerode definition of continuity for hybrid systems. The application of the definition of continuity leads to the construction of the HJB equation and its solution as the chattering combination of infinitesimal control actions (Figure 8).

The appropriate interpretation of this solution is that the coefficients of the tangent vectors satisfy the Livi Civita continuation equation (affine connection) from x to $x + Dx$ and coincides with the HJB equation from x to $x + Dx$.

This means that the Bellman Optimality Principle is the consequence of continuation in the hybrid systems formulation [8, 9]. The effect is that instead of having the previous limited result for linear systems that *if a solution exists*, then the Linear Quadratic or H^∞ result is exact, we can now use infinitesimal actions to both define the choices available, and then construct the HJB equation to be solved, secure in the knowledge that the T0 topology of hybrid systems tends to the Hausdorf topology in the limit (see [8] for a proof). The result is summarized below:

We note that both actions and disturbances are transformations on points in the carrier manifold. Each infinitesimal action or disturbance is represented as a derivation on the manifold M as discussed above. We construct an inference

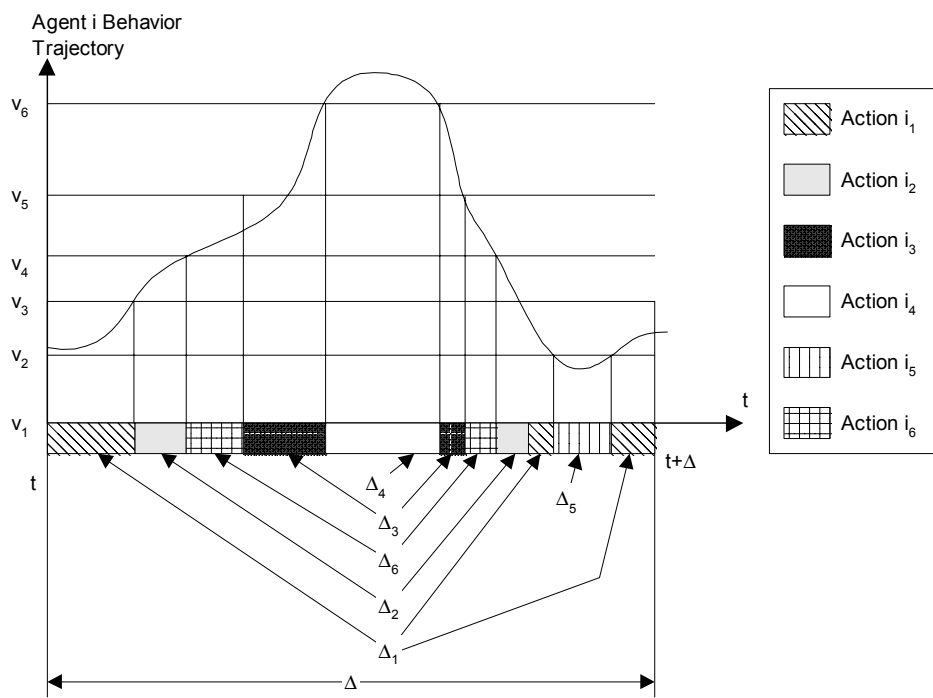


Figure 8: Agent i behavior as a chattering combination of infinitesimal actions

automaton which is the epsilon-optimal hybrid controller (D approximation). The evolution equations are:

$$\begin{array}{ll}
 q_{t+\Delta} = \delta(q_t, \omega_{t,t+\Delta}) & \text{neighborhood transition} \\
 \omega_{t,t+\Delta} & \text{solution of current equational terms} \quad (10) \\
 v|_q(\cdot) = \beta(q_t) & \text{infinitesimal action} \\
 u_{t+\Delta} = \exp(\Delta \cdot v|_q)(H(X, g)) & \text{current control law}
 \end{array}$$

The logic equations are:

$$\begin{array}{ll}
 Y_t = E(q_t) \cdot Y_t + K(q_t) & \text{Kleene-Schutzenberger Equation} \\
 & \text{Resolution of current relations} \quad (11) \\
 \omega_{t,t+\Delta} = S(Y_t) & \text{Selector function}
 \end{array}$$

Manufacturing Example – the PSC problem

Detailed models of enterprise-wide manufacturing processes necessarily include both qualitative and quantitative constraints. For example consider the problem of integrating high-level scheduling of the manufacturing production of a factory and low-level execution of manufacturing processes within a work cell of the factory [15]:

- *Qualitative characteristics and logical constraints:* The scheduler for a factory must produce the schedule by evaluating throughput, tardiness, work-in-progress, machine wear, and other metrics. Each of these input-output criteria can be measured as a number. These numbers are global criteria (which apply to the factory as a whole), and, since the criteria are not associated with a particular step in the manufacturing process, they treat the factory as a black box (i.e., to be analyzed by input-output considerations). Other metrics are associated with the quality of the product, which may be based on appearance, density, texture, thickness, or some other variable that can be sensed and/or approximated by an allowable range of values. Such metrics can be used to create an event-based switching function for altering production quantity and flow through the factory.
- *Quantitative characteristics and continuum constraints:* Contrast this with the problem of deciding what to do next at an individual work cell. The logical decisions discussed above are normally captured as events occurring at instants in time and must be compatible with the laws of physics which govern the continuous operation of motors, conveyor belts, sensors, actuators and so forth.

Implementation of factory planning, scheduling, and control requires synchronization of complex events and continuous processes between producers and consumers of work piece products between factory cells. The quality of factory

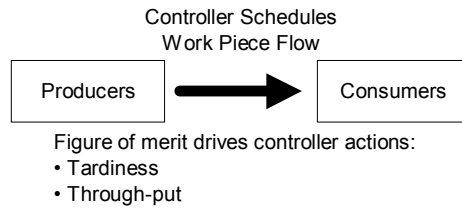


Figure 9: Controller performing as a scheduler

schedules in smoothly synchronizing these events and processes determines return on investment for factory components. Improved utilization would affect quality and increase return on investment. The quality of factory schedules also impacts customer satisfaction through on-time delivery of goods. Raw materials (stock) are transformed into finished products via operations performed at a number of work stations. The manufacturing plan for a product constrains the path of the raw stock through the work stations, and the scheduling system must assign this stock to an actual sequence of work stations where the operations prescribed in its plan are undertaken. An example, having several dozen work cells, is a floor panels factory. A typical work cell in the factory would consist of one or more workstations where an “align-panel-with-support-frame” function would be performed. The scheduling of panel work piece components between factory cells is governed by figures of merit (see Figure 9).

The scheduler must pay attention to a number of factors:

- Achievable—Can the plan called for actually work in this factory?
- Interface between two jobs—Hundreds of parts are undergoing some work at any moment. Do they interfere with each other? Do two parts require the same tool at the same time? Is any work piece likely to be ruined because it spoiled, or because the fixture required to hold it was busy?
- Material—Is the required stock available to start this job?
- Improvements—Can the order of work be changed to eliminate the need to reconfigure a workstation?
- Safety—Does the schedule allow for required operator breaks, shift changes? Is the schedule safe?

Even without these subjective factors, the general scheduling problem cannot be solved by classical methods. To compound the difficulty of achieving a solution, schedules are rarely used to completion. A machine break; stock fails to arrive, a tool breaks—change in the schedule is required.

Before a schedule change, one must consider that many workpieces are already flowing through the factory. The cost to restart this work is usually too

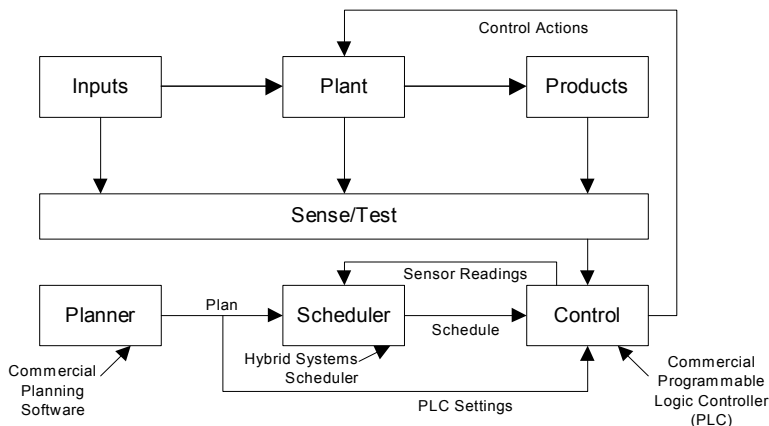


Figure 10: Hybrid systems as the “glue” between event-based systems and continuum systems

high to consider. Thus, the scheduler must produce a new schedule quickly while taking new, and possibly not previously anticipated, constraints into account.

The scheduler for a factory must produce the schedule by evaluating throughput, tardiness, work-in-progress, machine wear, and other metrics. Each of these input-output criteria can be measured as a number. These numbers are global criteria (apply to the factory as a whole), and, since the criteria are not associated with a particular step in the manufacturing process, treat the factory (the plant in Fig. 10) as a black box (i.e., to be analyzed by input-output considerations).

Contrast this with the problem of deciding what to do next at an individual work cell. The logical decisions discussed above, normally captured as events occurring at instants in time, must be compatible with the laws of physics which govern the continuous operation of motors, conveyor belts, sensors, actuators and so forth. If the factory is equipped with a single super computer and a very fast, unfailing sensor environment, then one program can be tasked with deciding for every work cell what it should do next, in complete detail. If communication lines to that computer fail, the factory stops. If its sensors lie, the factory may not stop, which can be worse. If simultaneous events overload the supercomputer, everyone waits. If small amounts of another product need to be produced between production runs of the current product, the plant must be reconfigured. Adequate solutions are available for achieving high-level plans for production. Adequate solutions are also available for achieving low-level control of individual work cells. The current challenge is in resolving logical inconsistencies when the current plan doesn’t match the throughput capability of the workstations or the continuum values of a workstation parameter don’t comply with the high-level abstraction of system status.

These considerations argue for a multiple-agent approach (Figs. 4, 5, and 6 with coordination among work cells to achieve reasonable performance. While a high degree of autonomy among agents is needed to support a “divide-and-conquer” approach to meeting the complexity of producing workable plant schedules, it is possible for one controller to degrade factory performance by shuffling parts between two orders in a looping fashion, which, in turn, can cause downstream workcells undue overhead in reconfiguring. Distribution of control is needed, so that work cells can make autonomous decisions, using advice from neighboring work cells, and performance criteria (goals) from a factory-wide goal-setting agent. Factory plans are made by considering certain constraints, normally in the context of a nominal manufacturing scenario. However, factory execution occurs in the context of actual decisions and events, which can deviate from the nominal scenario. Logical inconsistencies occurring at the higher levels in the nominal hierarchy (such as a new priority for scheduling that conflicts with the priority being executed) and continuum inconsistencies occurring at the lower levels in the hierarchy (such as violation of safety, quality, or performance constraints) lead to failure of being able to execute the *current* plan and the need to *reactively* create a consistent plan that complies with current logical and continuum constraints. In a general sense this is true for any application that spans the boundary between planning, where limited experimentation can be conducted, and situated activity, which has a much larger set of possible outcomes. The concept of an agent must appear on both sides of this boundary. Multiple-agent, declarative control supports off-line analysis and design and on-line generation of planning, scheduling, and control software that acts as the “glue” between commercially available high-level enterprise planning software and commercially available low-level control systems (see Fig. 10).

The factory model

We model a factory as a network of work cells. The kind of work cells and their connections can be defined using physical, process, and/or policy considerations. Typically several work cells will be connected via conveyers and communications. The collection of cell models (agents) and network connections comprise the factory model.

We consider each work cell as a physical plant and as an elemental component of the factory. Each cell has an agent controller that directs its activities (see Fig. 4 and 11).

Work cell attributes are:

- *Configurations*: Work cells take different states that define feasible operations. Each state is a configuration which defines the fixtures, cutters, and numerical control (nc) media needed for its operations.
- *Sensors*: Sensors, which measure plant performance parameters, are attached to each physical plant. The measurements periodically update the controller’s knowledge base. Measurements include velocity of a conveyer, rotational speed of a drill, feed rate of a cutter, pressure of a vacuum pad,

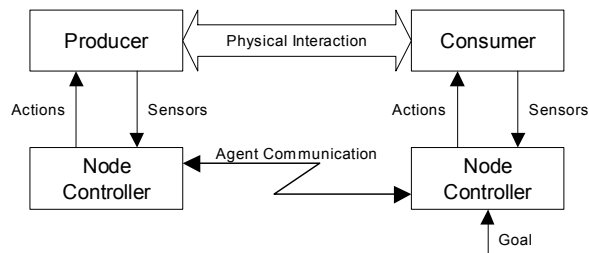


Figure 11: Experimental evaluation of a factory cell

number of items in a holding area, accuracy of a cut, and wear on the equipment.

- *Actuators*: Actuators implement the commands that the controller sends to the physical plant to alter its behavior. The controller periodically computes new actuator commands to satisfy the conditions prescribed by its planning component. Examples of commands are conveyor velocity = 2.3 rpm, drill Speed = 2100 rpm, next work piece = PIN6XXXX, route current item to work cell X.
- *Operations*: Each work cell is equipped to perform a set of operations. These correspond to factory work codes that define individual operations in the part's process plan. Each operation has a setup configuration, a setup time, an operation time, and a tear down time.
- *Constraints*: Work cells are characterized by the kinds of constraints they must satisfy. Some of these are the timing and configuration constraints mentioned above. Others involve the relationships between two work cells. Typical constraints are queue timing constraints, shared resource constraints, material handling constraints, and communication constraints.

An experiment with two work cells

We have conducted an experimental evaluation of the architecture in terms of a reactive Planning, Scheduling and Control system for a computer-simulated shop floor consisting of two workcells with an agent controlling each cell (see Fig. 11). The closed-loop simulation controls the flow of products and partial products through the shop and also controls each of the workcell production rates, according to a user provided figure of merit, encoded production requirements, and encoded planning, scheduling, and control strategies.

The simulation and control is implemented on a Unix workstation using three Quintus Prolog processes: one for the producer controller agent, one for the consumer controller agent, and a third for the simulation. The simulation includes a generator of orders of products the shop can produce. At any time, the list of outstanding orders constitutes the Goal of the system. Each order

is represented by a data structure whose tokens include order number, delivery time, quantity, priority, and process and material information (see Fig. 12). Each agent has the capability of constructing a variational schema on-line, formulated as a theorem, for characterizing the desired behavior that will drive the system toward the desired goal while satisfying the encoded requirements. The central function of each agent at each update interval is to prove that its theorem logically follows from the current status of its encoded knowledge base and satisfies the constraints imposed by the other agent. If the proof is successful, as a side effect, command actions to the cells are instantiated otherwise, i.e., if the theorem does not logically follow from the current status of the knowledge base an adaptation procedure is activated by the agent to generate a corrected statement of the theorem. This failure-driven adaptation is the essence of the reactive strategy in our architecture.

```

order(order27,12,[pop2(cut2),cop2(nc2)],mat1,76,3).
order(order28,14,[pop2(cut4),cop1(nc1)],mat1,101,4).
order(order29,13,[pop3(cut4),cop3(nc3)],mat2,105,3).
order(order30,7,[pop1(cut3),cop1(nc1)],mat1,71,1).
order(order31,5,[pop2(cut3),cop2(nc2)],mat2,75,1).
order(order32,5,[pop2(cut3),cop1(nc2)],mat1,76,4).
order(order33,11,[pop2(cut2),cop1(nc1)],mat2,119,4).
order(order34,5,[pop2(cut2),cop1,(cut4)],mat1,102,1).
order(order35,7,[pop1(cut3),cop2(nc2)],mat1,105,1).
order(order36,14,[pop2(cut2),cop3(nc3)],mat1,73,1).
order(order37,7,[pop3(cut4),cop3(nc3)],mat1,50,1).
order(order38,10,[pop1(cut1),cop1(cut4)],mat2,60,500). % AOG
order(order39,19,[pop1(cut1),cop1(nc1)],mat2,111,1).
order(order40,7,[pop1(cut3),cop1(nc2)],mat1,56,3).
order(order41,6,[pop1(cut4),cop1(nc1)],mat1,80,1).
order(order42,16,[pop2(cut3),cop3(nc3)],mat1,112,3).
order(order43,18,[pop2(cut3),cop3(nc3)],mat1,85,5).
order(order44,5,[pop1(cut3),cop2(nc2)],mat1,112,3).
order(order45,7,[pop2(cut3),cop1(nc2)],mat2,85,4).
order(order46,11,[pop1(cut2),cop3(nc3)],mat1,51,1).
order(order47,9,[pop2(cut4),cop1(nc1)],mat2,73,3).

```

Figure 12: Output of multiple-agent scheduler.

To study and tune this adaptation schema, the simulation has capabilities for the generation of external events to force the controlling agents to react to unexpected changes. Examples of these events are changes in inventory, goal orders, failures, etc. The central activity provided in the architecture for adapting reactively to those changes is to tune the processing rate of the Producer and the Consumer. Fig. 12 provides the result of applying the conservation principle of (8) in a multiple-agent architecture controlled by the near-optimal solution of (9) to reactively tune different operations on different materials by different numerically controlled machines.

The on-line reactive matching of current logic and continuum constraints from plant sensors with user-defined logic and continuum models is the basis for generating procedures to interface heterogeneous components of the manufacturing process, and for our belief that the resulting architecture will support incremental expansion of new components with greatly reduced requirements

for expensive experimentation validation. We do not expect to fully eliminate the need for experimentation because the degree of “trust” in the newly composed architecture will depend on the rules for composition of the components. However, to the degree that the composition rules are correct, the methodology will be a formally correct composition of the components, the focus of the verification and validation effort will be raised to the component level, and the results will be reusable across the confederation of components.

Maruti Implementation

In addition to the manufacturing simulation, we have built a single-agent demonstration that relies on use of National Instruments LabView for analog-to-digital and digital-to-analog transformations and has a PC processor running a single process to evaluate the logic Lagrangian of the relaxed optimization problem. Similarly, a multiple-agent demonstration [42] relies on single processes in the PC for each agent and a low-data-rate, higher-level process in the coordinating agent processor. In general, a distributed, real-time operating system, such as envisioned for the Signal system [26] or the CAMEL system [25] is needed to implement the reactive control programs. Our implementation approach envisions use of the Maruti real-time operating system [43] to provide an open-systems solution. The Maruti system is being considered by the Open Software Foundation as the standard system for implementing real-time, distributed systems. Maruti currently runs as an adjunct to the MACH operating system, thus providing access to the power of that distributed version of the Unix operating system when it is needed for more complex tasks such as user interface. However, the Maruti system can also be ported to other general-purpose computers, which may be a lower-cost alternative for implementing computer-controlled systems for simpler tasks such as lower-level servo control. An outline of the technical approach for implementation is described below.

If we consider a manufacturing example, the implementation of the control automata constructed as described above requires that the hybrid control agents interface with the cell-level controllers which interact with the sensors and actuators of the actual machine tools. The implementation architecture that would support the interface between the factory floor level, the work-cell level, and the sensor/actuator level is shown in Fig. 13. The commands generated by the agent automata have to be sent to the controllers and the actuators in a timely manner, and the data collected by the sensors have to be transferred to the agent automata so that they can take that information into account in the generation of the next set of proof automata. Further, in order to be able to scale this system, the low-level controls exercised by each cell controller receive command control laws from a corresponding agent automata. The operation of the system has to be able to support real-time operations in which temporal guarantees can be met. In addition, the system has to be reactive in that it can change schedules during execution. The Maruti system developed at the University of Maryland has the capabilities necessary (especially the ability to

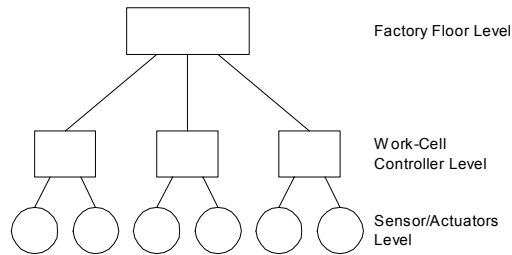


Figure 13: Implementation architecture

implement on-line changes in the “calendar” for scheduling of processes) to provide this support. This system has been developed with partial support from ARPA.

Fig. 14 shows a simplified version of Maruti environment and how the development of applications proceeds in this environment. The current version of Maruti kernel is being implemented to run on top of Mach microkernel. It supports the management of resources using calendars while maintaining a support for the execution of hard, soft, and non-real-time applications. The schedule of execution is maintained in calendars, which are used at the run time to control the dispatching of tasks. Therefore, the run-time environment for the system is very simple.

The current implementation supports reactive operation in that the calendar entries can be changed during execution. This reactive operation is essential to support the changes to be made to the operations of a cell controller in response to the command control laws it receives from the agent automata.

Monitoring and fault handling capabilities are built as an integral part of applications in the Maruti system. Monitoring of operations is carried out at a detailed level so that any improper operations can be detected quickly so that such information can be sent to agent automata allowing them to take the necessary corrective action.

Summary

Our formulation supports multiple views and incremental development of a precise statement of control problems in terms of multiple-agent hybrid declarative control. Our approach characterizes the statement of control problems via a knowledge base of equational rules that describes the dynamics, constraints, and requirements of the system being controlled (manufacturing workstation, conveyor, processes, manpower, scheduling and planning systems, etc.). The proposed CACE environment will support an end-to-end methodology for requirements formulation and analysis, modeling, design, simulation, performance analysis, and implementation.

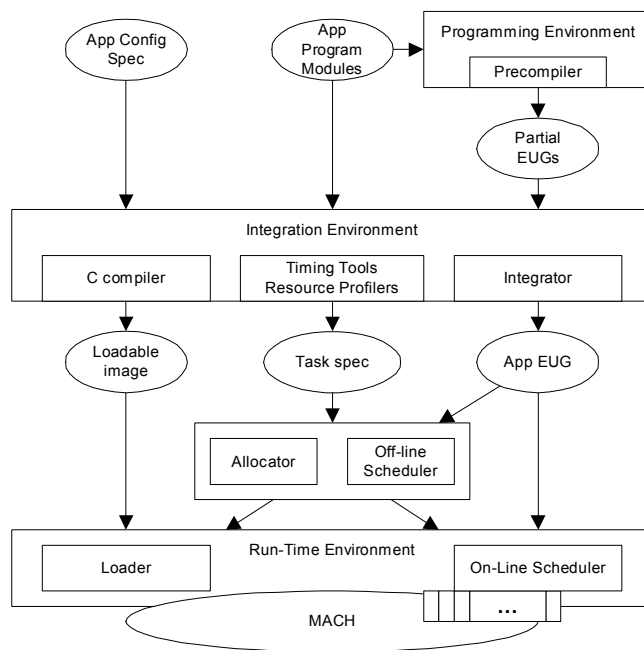


Figure 14: Maruti environment

The CACE environment discussed here does not exist. However, we emphasize that the major pieces of our proposed architecture for CACE environments do exist in isolation from each other. Tools, applications, experiments, theoretical foundations, and initial results are available for scenario-based requirements analysis, declarative hybrid control, and distributed, real-time operating systems. We are convinced that a formal methods approach is required to link the major components of a CACE architecture. We are also convinced that a rigorous and systematic approach to integration of set-based models and continuous-time models of complex systems is required to support the control engineering process. We intend to pursue hybrid systems as the means of realizing the required integration.

References

- [1] A. Benveniste and K. Astrom, "Meeting the Challenge of Computer Science in the Industrial Application of Control: An Introductory Discussion to the Special Issue," *IEEE Transactions on Automatic Control*, vol. 38, pp. 1004–1010, 1993.
- [2] W. Kohn, "A Declarative Theory for Rational Controllers," *Proceedings of the 27th IEEE CDC*, vol. 1, pp. 131–136, Austin, TX, Dec. 7–9, 1988.
- [3] W. Kohn, "Declarative Control Architecture," *CACM*, Aug. 1991, vol. 34, no. 8.
- [4] W. Kohn and A. Nerode, "Multiple-Agent Hybrid Control Systems," *Proceedings of IEEE CDC '92*, Tucson AZ, Dec 16–19, 1992.
- [5] A. Nerode and W. Kohn, "Multiple Agent Declarative Control Architecture," *Hybrid Systems*, Springer-Verlag, 1993, vol. 736, R. Grossman, A. Nerode, T. Rischel, A. Ravn, eds.
- [6] W. Kohn and A. Nerode, "Models for Hybrid Systems: Automaton, Topologies, Control, Controllability, Observability," *Hybrid Systems*, Springer-Verlag, 1993, vol. 736, R. Grossman, A. Nerode, T. Rischel, A. Ravn, eds.
- [7] A. Nerode and W. Kohn, "Multiple Agent Autonomous Control: A Hybrid Systems Architecture," *Logical Methods In Honor of Anil Nerode's Sixtieth Birthday*, N.C. Crossley, J.B. Remmel, M.E. Sweedler, eds., Birkhauser, Boston, MA, 1993.
- [8] W. Kohn, "Distributed Hybrid Controller Architecture," *Proceedings of the Workshop on Hybrid Systems and Autonomous Control*, Mathematical Sciences Institute, Cornell University, Ithaca, NY, Oct. 28–30, 1994. To be published in a Springer-Verlag volume 1995.
- [9] J. James, W. Kohn, and A. Nerode, "A Hybrid System Architecture and Epsilon-Optimal Solution of the Non-linear Hamilton-Jacobi-Bellman

- Equation: A Step on the Road to High-Safety, High-Assurance Systems,” Proceedings of the Workshop on Hybrid Systems and Autonomous Control, Mathematical Sciences Institute, Cornell University, Ithaca, NY, 28–30 Oct. 1994. To be published in a Springer-Verlag volume 1995.
- [10] L.C. Young, *Optimal Control Theory*, Chelsea Publishing Co., NY, 1980.
 - [11] J.R. James, D.K. Frederick, and J.H. Taylor, “Use of Expert Systems Programming Techniques for the Design of Lead-Lag Compensators,” *IEEE Proceedings*, vol. 134, Pt. D, no. 3, May 1987.
 - [12] B.W. Boehm and W.L. Scherlis, “Megaprogramming,” *Proceedings of the DARPA Software Technology Conference*, Los Angeles, CA, April 1992.
 - [13] E. Mettala and M.H. Graham, “The Domain-Specific Software Architecture Program,” *Proceedings of the DARPA Software Technology Conference*, Los Angeles, CA, April 1992.
 - [14] C. Landauer and K.L. Bellman, “New Mathematics for Computing (A New Initiative),” Proceedings of Hybrid Systems and Autonomous Control, Mathematical Sciences Institute, Cornell University, Ithaca, NY, Oct. 28–30, 1994.
 - [15] W. Kohn, J. James, and A. Nerode, “Multiple Agent Hybrid Control Architecture: A Generic, Open Architecture for Incremental Construction of Reactive Planning, Scheduling, and Control Systems for Manufacturing,” a white paper developed in September 1994.
 - [16] J.S. Albus, C. McLean, A. Barbera, M. Fitzgerald, “An Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Environment,” *4th IFAC/IFIP Symposium on Information Control of Robots in Manufacturing Technology*, Gaithersburg, MD, Oct. 1982.
 - [17] A.J. Barbera, J.S. Albus, M.L. Fitzgerald, and L.S. Haynes, “RCS: The NBS Real-Time Control System,” *Robots 8 Conference and Exhibition*, Detroit, MI, June 1984.
 - [18] J.S. Albus, H.G. McCain, and R. Lumia, “NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM),” *NIST (formerly NBS) Technical Note 1235*, April 1989 edition.
 - [19] J.S. Albus and R. Quintero, “Toward a Reference Model Architecture for Real-Time Intelligent Control Systems (ARTICS),” *Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems*, Charles J. Herget, ed., September 1990.
 - [20] J.S. Albus, R. Quintero, R. Lumia, M. Herman, R. Kilmer, and K. Goodwin, “Concepts for a Reference Model Architecture for Real-Time Intelligent Control Systems (ARTICS),” *NIST Technical Note 1277*, April 1990.

- [21] R. Quintero, "Toward an Intelligent Control Systems Development Methodology," *Proceedings of the JSGCC Software Initiative Workshop*, December 1–4, 1992. Guidance and Control Information Analysis Center (GACIAC) PR92-03.
- [22] S. Hufnagel, K. Harbison, and D. Goldstein, "Scenario-Driven Requirements Analysis Method," *Proceedings of the Joint Services Guidance and Control Workshop*, December 1992.
- [23] B.A. Ogunnaike, "Problems and Challenges of Industrial Process Control: A Commercial Polymerization Reactor Case Study," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [24] G. Grubel, "The ANDECS CACE Framework," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [25] R. Rutz and J. Richert, "CAMEL—An Open CACSD Environment," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [26] A. Benveniste and P. Le Guernic, "Hybrid Dynamical Systems Theory and the SIGNAL Language," *IEEE Transactions on Automatic Control*, 35(5), May 1990, pp. 535–546.
- [27] Pang, G. K. H., H. Bacakoglu, M. F. Ho, Y. Hwu, B. J. Robertson, and B. Shahrrava, A Knowledge-Based System for Control Design Using MEDAL, *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [28] C.M. Rinvall, H.A. Spang III, J. Farrell, M. Radecki, and M. Idelchik, "An Open Architecture for Automatic Code Generation Using the BEACON CACE Environment," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [29] M. Englehart and M. Jackson, "ControlH: A Fourth-Generation Language for Real-Time GN&C Application," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [30] J.R. James, J.H. Taylor, and D.K. Frederick, "An Expert System Architecture for Coping With Complexity in Computer-Aided Control Engineering," *Proceedings of the 3rd IFAC/IFIP International Symposium on Computer-Aided Design in Control and Engineering Systems*, Syngby, Denmark, July 31–Aug. 2, 1985.
- [31] J.R. James, "Expert System Shells for Combining Symbolic and Numeric Processing in CADCS," *Proceedings of the 4th IFAC/IFIP International Symposium on Computer-Aided Design in Control and Engineering Systems*, Beijing, PRC, 1988.

- [32] D.K. Frederick, J.R. James, A. Antonetti, and Hiroyuki Nitta, "A Second-Generation Expert System for Computer-Aided Control System Design," *Proceedings of the 45th IFAC/IFIP International Symposium on Computer-Aided Design in Control and Engineering Systems*, Swansea, U.K., July 15–17, 1991.
- [33] W. Kohn and J. James, "Declarative Control Architecture for a Laboratory Test Fixture," Intermetrics Report to the Army Armaments Research, Development and Engineering Center, Nov. 1994.
- [34] W. Kohn, J. James, A. Nerode, and J. Lu, "Multiple-Agent Hybrid Control Architecture for the Target Engagement Process (MAHCA-TEP)," version 0.2 of *MAHCA-TEP: Technical Background, Simulation Requirements, and Engagement Model*, Intermetrics Technical Report, Aug. 25, 1994.
- [35] S. Sastry, "Verification Problems in Hierarchical Hybrid Control System for Intelligent Highways," *Proceedings of the Workshop on Hybrid Systems and Autonomous Control*, Mathematical Sciences Institute, Cornell University, Ithaca, NY, Oct. 28–30, 1994. To be published in a Springer-Verlag volume, 1995.
- [36] A.R. Deshpande and P. Varaiya, "Viable Control of Hybrid Systems," *Proceedings of the Workshop on Hybrid Systems and Autonomous Control*, Mathematical Sciences Institute, Cornell University, Ithaca, NY, Oct. 28–30, 1994. To be published in a Springer-Verlag volume, 1995.
- [37] W.J. Bencze and G. F. Franklin, "A Separation Principle for Hybrid Control System Design," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994.
- [38] R.W. Brockett, "Language-Driven Hybrid Systems," *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, Dec. 14–16, 1994. pp. 4210–4214.
- [39] J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon, "Digital Control From a Hybrid Perspective," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994, pp. 4241–4246.
- [40] M.S. Branicky, V.S. Borkar, and S.K. Mitter, "A Unified Framework for Hybrid Control," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design*, March 1994, pp. 4228–4234.
- [41] W. Kohn, A. Nerode, and J. Remmel, "Multiple-Agent Control of Hybrid Systems," March 1994.
- [42] N. Coleman, S. Banks, J. James, A. Nerode, and W. Kohn, "Hybrid Systems Models, Simulation, and Testing," *Proceedings of the 5th IEEE Conference on AI Simulation and Planning in High Autonomy System*, Gainesville, FL, December 1994.

- [43] M. Saksena, J. da Silva, and A. Agrawala, "Design and Implementation of Maruti-II," *Advances in Real-Time Systems*, Sang Son, ed., Prentice-Hall, 1994, pp. 73–99.

Wolf Kohn received his M.S. and Ph.D. degrees in electrical engineering from MIT in 1976 and 1978, respectively. Kohn has played a pioneering role in the creation of hybrid systems theory and technology. He is currently chief scientist of Intermetrics in the area of formal methods and intelligent control systems. From 1987 to 1991, he was chief researcher of AI at Boeing Computer Services Research & Technology Branch, where he pioneered the development of Declarative Control Theory and technology with applications to avionics, air traffic control, robotics, and intelligent manufacturing. He built prototype controllers in each of these application areas. For the past three years he has worked with Anil Nerode to develop mathematical foundations for multiple-agent hybrid control theory and with John James to apply these results to large-scale systems.

John James received the B.S. degree from the United States Military Academy in 1967, the M.S.E.E. degree from the University of California, Berkeley, in 1973 and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute in 1986. He is a past chairman of the IEEE Technical Committee on Computer-Aided Control System Design (CACSD) and was general chair of the 1992 IEEE Symposium on CACSD. His current research interests are in the area of hybrid systems and, with Robert Grossman of the University of Illinois at Chicago, he organized invited sessions on hybrid systems at the 1992 and 1993 IEEE Conferences on Decision and Control. He has been engineering manager for Intermetrics efforts in applying hybrid systems, including conducting demonstrations of single-agent and multiple-agent control architectures.

Anil Nerode is Goldwin Smith Professor of Mathematical and Computer Science and director of the Mathematical Sciences Institute at Cornell University. He received his Ph.D. in mathematics at the University of Chicago in 1956 under Saunders MacLane and was an NSF postdoctoral fellow with Kurt Godel at the Institute for Advanced Study in Princeton and with Alfred Tarski at the University of California at Berkeley. He is the author of 150 papers and books in mathematical logic, algebra, recursive functions, automata, recursive and polynomial time algebra, concurrency and distributed systems, non-monotonic reasoning systems, logic programming, and hybrid systems. He was an editor of the *Journal of Symbolic Logic* for 16 years, has been an editor of *Annals of Pure and Applied Logic* for 15 years, and is an editor for other journals in mathematics, applied mathematics, computer science, and modeling and simulation. He has been a military consultant in design of weapon systems for 41 years. He is currently a vice president of the American Mathematical Society.

Karan Harbison received an M.S. from the University of Texas at Austin and an M.A. and Ph.D. from the University of Texas at Arlington. She was manager the UTA Automation and Robotics Research Institute (ARRI) Artificial Intelligence for Manufacturing (AIM) Laboratory for its first four years. Her major research interests are in the areas of artificial intelligence, control sys-

tems, and systems/software engineering. She leads the Program for Research in Intelligent Systems and Machines (PRISM) Lab. The PRISM Lab has ongoing projects in intelligent control and analysis and design methodologies for intelligent systems. Her group is a member of the Air Force ManTech's Next Generation Controller Program and ARPA's Semi-Autonomous Surrogate Vehicle Program. Harbison is studying the semantics of intelligent, hybrid control and specification of control/planning languages for manufacturing control systems. Her group is also designing a scenario-based object-oriented engineering process for autonomous vehicles, including aspects of real-time control and complex planning subsystems.

Ashok Agrawala received his Ph.D. in applied mathematics from Harvard University in 1970. He joined the faculty of the Department of Computer Science at the University of Maryland in 1971 and has been a professor since 1982. He has also been a professor at the University of Maryland Institute of Advanced Computer Studies since 1985. He established the Systems Design and Analysis Group in the Department of Computer Science in 1977. This group has conducted research on system design and performance issues for the last 15 years. Agrawala's current research interests are in issues associated with creation of a hard real-time operating system that supports distributed, heterogeneous operation, as well as multiple degrees of fault tolerance. Agrawala has led the creation of the Maruti operating system, which has these attributes. Agrawala has applied the temporal analysis techniques to the flow and congestion control problems of computer networks.