

*“With autonomy we declare that no sphere is off limits.
We will send our spacecraft to search beyond the
horizon, accepting that we cannot directly control
them, and relying on them to tell the tale.”*

*– Bob Rasmussen
New Millennium Autonomy Team*



Immobile Robots

AI in the New Millennium¹

Brian C. Williams and P. Pandurang Nayak

■ A new generation of sensor-rich, massively distributed, autonomous systems are being developed that have the potential for profound social, environmental, and economic change. These systems include networked building energy systems, autonomous space probes, chemical plant control systems, satellite constellations for remote ecosystem monitoring, power grids, biospherelike life-support systems, and reconfigurable traffic systems, to highlight but a few. To achieve high performance, these immobile robots (or immobots) will need to develop sophisticated regulatory and immune systems that accurately and robustly control their complex internal functions. Thus, immobots will exploit a vast nervous system of sensors to model themselves and their environment on a grand scale. They will use these models to dramatically reconfigure themselves to survive decades of autonomous operation. Achieving these large-scale modeling and configuration tasks will require a tight coupling between the higher-level coordination function provided by symbolic reasoning and the lower-level autonomic processes of adaptive estimation and control. To be economically viable, they will need to be programmable purely through high-level compositional models. Self-modeling and self-configuration, autonomic functions coordinated through symbolic reasoning, and compositional, model-based programming are the three key elements of a model-based autonomous system architecture that is taking us into the new millennium.

The limelight has shifted dramatically in the last few years from an AI tradition of developing mobile robots to that of developing software agents (that is, *softbots*) (Etzioni and Weld 1994; Kautz et al. 1994; Maes and Kozierok 1993; Dent et al. 1992; Etzioni and Segal 1992). This shift is reflected in Hollywood, which has replaced R2D2 with cyberpunk descendants of Disney's TRON as the popular robot icon. One motivation for the shift is the difficulty and cost of developing and maintaining physical robots, which are believed to substantially impede progress to-

ward AI's central goal of developing agent architectures and a theory of machine intelligence (Etzioni and Segal 1992). As Etzioni and Segal argue, software environments, such as a UNIX shell and the World Wide Web, provide softbots with a set of ready-made sensors (for example, LS and GOPHER) and end effectors (for example, FTP and TELNET) that are easy to maintain but still provide a test bed for exploring issues of mobility and real-time constraints. At the same time, the recent Internet gold rush and the ensuing web literacy has provided an enormous textual corpus that screams for intelligent information-gathering aides (Levy, Rajaraman, and Ordille 1996; Knoblock and Levy 1995).

However, two concerns have been raised about using software agents as a research test bed and application domain: First, softbots often operate in an environment lacking the rich constraints that stem from noisy, analog sensors and complex nonlinear effectors that are so fundamental to physical environments. Can such a software environment adequately drive research on agent kernels? Second, given that much of the information on the Internet is textual, will oft-envisioned softbot applications, such as information gathering and synthesis, be viable before the hard nut of language understanding has been cracked?

In this article, we argue that the information-gathering capabilities of the Internet, corporate intranets, and smaller networked computational systems supply additional test beds for autonomous agents of a different sort. These test beds, which we call immobile robots (or *immobots*), have the richness that comes from interacting with physical environments yet promise the ready availability associated with the networked software environment of softbots. Potential immobile robots include networked building energy systems, autonomous space probes, chemical plant control systems, satellite constellations for ecosystem monitoring, power grids, biosphere

We argue that the focus of attention of immobile robots is directed inward, toward maintaining their internal structure, in contrast to the focus of traditional robots, which is toward exploring and manipulating their external environment.

life-support systems, and reconfigurable traffic systems. Conversion of these and other real-time systems to immobile robots will be a driving force for profound social, environmental, and economic change. The power of a term such as robot, mobot, or softbot is the way in which it sets our imagination free. The purpose of this article is to see where the im-mobot metaphor can take us, from science fiction to fundamental AI insights to significant applications.

We argue that the focus of attention of im-mobile robots is directed inward, toward maintaining their internal structure, in contrast to the focus of traditional robots, which is toward exploring and manipulating their external environment. This inward direction focuses the im-mobot on the control of its complex internal functions, such as sensor monitoring and goal tracking; parameter estimation and learning; failure detection and isolation; fault diagnosis and avoidance; and recovery, or moving to a safe state. Metaphorically speaking, the main functions of an im-mobot correspond to the human nervous, regulatory, and immune systems rather than the navigation and perceptual systems being mimicked in mobile robots.

Finally, we argue that these im-mobots give rise to a new family of autonomous agent architectures, called *model-based autonomous systems*. Three properties of such systems are central: First, to achieve high performance, im-mobots will need to exploit a vast nervous system of sensors to model themselves and their environment on a grand scale. They will use these models to dramatically reconfigure themselves to survive decades of autonomous operations. Hence, self-modeling and self-configuration make up an essential executive function of an im-mobot architecture. Second, to achieve these large-scale modeling and configuration functions, an im-mobot architecture will require a tight coupling between the higher-level coordination function provided by symbolic reasoning and the lower-level au-tonomic processes of adaptive estimation and control. Third, to be economically viable, im-mobots will have to be programmable purely from high-level compositional models, supporting a “plug and play” approach to software and hardware development.

These properties of a model-based au-tonomous system are embodied in two im-plemented systems: MORIARTY (Williams and Mil-lar 1996) and LIVINGSTONE (Williams and Nayak 1996). LIVINGSTONE provides a kernel for controlling the immune system of immobile robots, and MORIARTY provides part of the ker-

nel for controlling an im-mobot’s regulatory system. Our work on these systems fuses re-search from such diverse areas of AI as model-based reasoning, qualitative reasoning, plan-ning and scheduling, execution, propositional satisfiability, concurrent reactive languages, Markov processes, model-based learning, and adaptive systems. MORIARTY and LIVINGSTONE are grounded in two im-mobot test beds. MORIARTY was part of the RESPONSIVE ENVIRONMENT (Elrod et al. 1993; Zhang, Williams, and Elrod 1993), an intelligent building control system devel-oped within the Ubiquitous Computing Pro-ject at Xerox PARC. LIVINGSTONE is part of the REMOTE AGENT, a goal-directed, fully au-tonomous control architecture, which will fly the National Aeronautics and Space Admin-istration (NASA) *Deep Space One* space probe in 1998 (Pell, Bernard, et al. 1996). At least one of these im-mobots has the promise of significant impact at the beginning of the new millennium.

Examples of Immobile Robots

Hardware advances in cheap single-chip con-trol and communication processors, sensors, point actuators, analog-to-digital converters, and networking have enabled a new category of autonomous system that is sensor rich, massively distributed, and largely immobile. This technology is being quickly embedded in almost every form of real-time system, from networked building energy systems to space-craft constellations. The current generation of these hybrid hardware-software systems has created a slumbering giant whose potential has only begun to be tapped. Furthermore, they offer a diverse set of ready-made im-mobile-robot test beds just waiting to be exploit-ed by AI researchers.

The potential of this technology captured the imagination of Hollywood in the 1960s with such movies as *2001: A Space Odyssey*, *The Forbin Project*, and *The Andromeda Strain* and the *Star Trek* television episode “Spock’s Brain.” The most famous computational in-telligence, the HAL9000 computer, and the most famous biological intelligence, Spock, were both for a time immobile robots. In this article, we use HAL and Spock as starting points for two immobile robot futures.

HAL9000

In the movie *2001*, HAL is the first of a new generation of computers with unpreced-ented intelligence and flawless behavior. HAL is characterized as the single mission element with the greatest responsibility, the “brain

and central nervous system" of a spacecraft targeted for Jupiter.

It might seem puzzling to call HAL an im-mobot; after all, it travels from Earth to Jupiter in 18 months, making it the fastest manmade artifact of its time. However, traditional mobile robots typically focus on their external environment and on the tasks of navigation and obstacle avoidance. In contrast, the movie portrays HAL with an extreme sense of immobility. The camera zooms in for long periods on HAL's optical sensors, which are attached to walls throughout the spaceship. HAL's actuators are also extremely limited; examples include the opening and closing of doors and the raising and lowering of beds. The distribution of these sensors and actuators throughout the spacecraft compensates for their immobility, giving HAL a sense of omnipresence. HAL's attention is focused inward throughout the movie, highlighted by the characterization of HAL as the "brain and central nervous system" of the spacecraft as opposed to the navigator and pilot. HAL's major responsibility is to look after the health of the spacecraft and crew, including monitoring the spacecraft's health, performing fault diagnosis and repair, operating the life-support systems, and continuously monitoring the medical status of crew members in hibernation. Finally, the movie highlights the connection between HAL's higher-level symbolic reasoning and the complex, low-level, automatic processes distributed throughout the spacecraft. Hence, HAL can be thought of as the spacecraft's immune system.

The New Millennium Program

Although many aspects of *2001* now seem farfetched, the current future of space exploration is no less exciting. With the creation of the New Millennium Program in 1995, NASA put forth the challenge of establishing a virtual presence in space through an armada of intelligent space probes that autonomously explore the nooks and crannies of the solar system: "With autonomy we declare that no sphere is off limits. We will send our spacecraft to search beyond the horizon, accepting that we cannot directly control them, and relying on them to tell the tale" (Rasmussen 1996).

This presence is to be established at an *Apollo*-era pace, with software for the first probe to be delivered in mid-1997, leaving only 1-1/2 years for development, and the probe (*Deep Space One*) to be launched in mid-1998. The additional constraint of low cost is of equal magnitude. Unlike the billion-

dollar *Galileo* and *Cassini* missions with hundreds of members in their flight operations teams, New Millennium Program missions are to cost under \$100 million, with only tens of flight operators. The eventual goal is a \$50 million spacecraft operated by a mere handful of people. Achieving these goals will require autonomy and robustness on an unprecedented scale.

The final challenge, spacecraft complexity, is equally daunting. Consider *Cassini*, NASA's state-of-the-art spacecraft headed for Saturn. *Cassini's* "nervous system" includes a sophisticated networked, multiprocessor system, consisting of two flight computers that communicate over a bus to more than two dozen control units and drivers. This nervous system establishes complex sensing and control paths to an array of fixed sensors and actuators, including inertial reference units, sun sensors, pressure sensors, thrusters, main engines, reaction wheels, and heaters. The most complex is the main engine subsystem, consisting of a web of redundant pipes, valves, tanks, and engines that offer exceptional levels of reconfigurability. Coordinating these complicated, hybrid systems poses significant technical hurdles.

Together, the goals of the New Millennium Program pose an extraordinary opportunity and challenge for AI. As with HAL, our challenge is to provide the immune system for this type of immobile robot over the lifetime of its mission.

"Spock's Brain"

A second example of an immobile robot from 1960's Hollywood comes from a little-known episode of the original *Star Trek* series called "Spock's Brain." In this episode, Spock's body is found robbed of its brain by an unknown alien race, and the crew of the *Enterprise* embarks on a search of the galaxy to reunite Spock's brain and body. Spock detects that he has a new body that stretches into infinity and that appears to be breathing, pumping blood, and maintaining physiological temperature. When discovered, Spock's brain is found to be within a black box, tied in by light rays to a complex control panel. Instead of breathing, maintaining temperature, and pumping blood, he is recirculating air, running heating plants, and recirculating water. That is, the function that requires this supreme intelligence is the regulation of a planetwide heating and ventilation system.

As with HAL, Spock's body in this case is extremely immobile. The episode portrays an immobile robot as a massively distributed be-

*... in 1995,
NASA put
forth the
challenge of
establishing
a virtual
presence in
space through
an armada of
intelligent
space probes
that au-
tonomously
explore the
nooks and
crannies of
the solar
system*

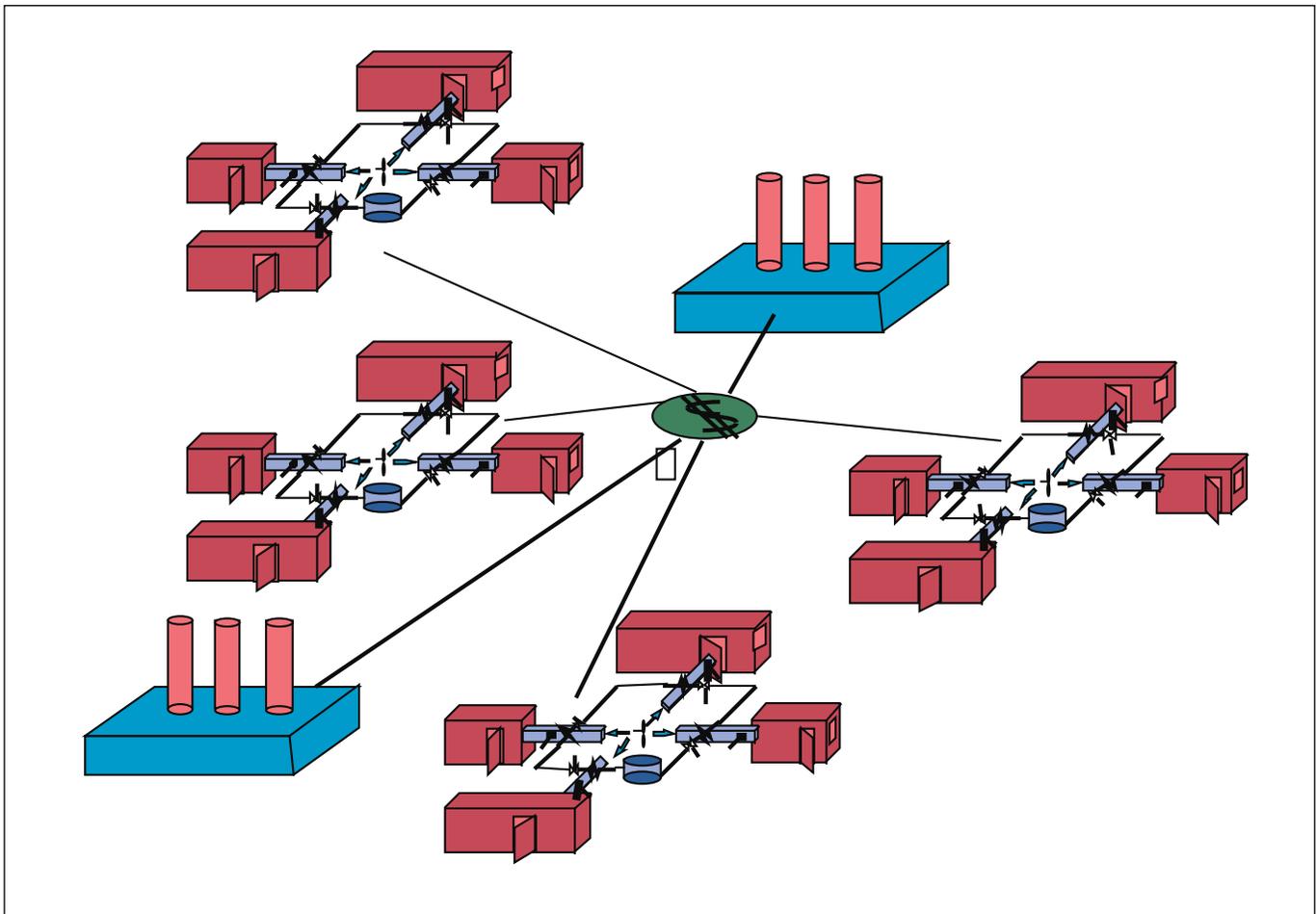


Figure 1. *Immobots of Unprecedented Size Are Being Developed in the Area of Building Energy Management.*

Key elements include networked building control systems, optical networks that connect buildings on a power grid to multiple power-production facilities, deregulation of the power industry, and the establishment of a dynamic energy market.

hemoth, sufficient to encircle a globe. Again, the crucial link between high-level reasoning (that is, Spock's brain) and autonomic processes is highlighted. Finally, although *2001* highlights HAL's function as an immune system that maintains the health of the immobot, this episode highlights Spock's function as a regulatory system, performing high-fidelity control of the immobot's internal organs. A regulatory system provides an efficient metabolism, supplying central resources efficiently, equitably distributing these resources throughout the system, consuming these resources efficiently, and attenuating the effects of disturbances.

The Responsive Environment

The concept in "Spock's Brain" is not as implausible as it might seem. In fact, it is quickly becoming a reality through a confluence of forces (figure 1). First is the broad installation of the new generation of networked building

management systems. These are networked multiprocessor systems containing hundreds to thousands of processors that allow high-fidelity sensing and control throughout a building. They will enable systems that are far more energy efficient and at the same time improve indoor air quality. Second is the interconnection of building and home utilities through optical fiber networks, which has led to several commercial alliances between members of the utility and telecommunications industries. Third is the deregulation of the utilities and the ensuing establishment of computer-based energy markets. These new energy markets will allow peak and average electric rates to be fluidly adjusted, enabling more even and stable balancing of the energy load. Technologies for storing energy within buildings during off-peak hours will enable even greater stability and efficiency. The rapid deployment of these networked control technologies has already generated a multibillion

dollar growth in the service industries for remote building monitoring and maintenance. However, the potential of these immobile robots for optimal energy-efficient control has yet to be tapped.

These are only two examples of an ever-increasing collection of immobile robot test beds being constructed. NASA's Earth-observing system is moving toward a level of sensing that will enable full Earth ecosystem modeling, providing insight into problems of pollution, global warming, and ozone depletion. Vast networked control systems are generating a revolution in chemical processing, factory automation, drug manufacturing, and semiconductor fabrication, producing immobile robots that enable substantial improvements in quality, efficiency, and safety.

These are not far-off dreams. Many government and private institutions, such as NASA, Pacific Gas and Electric, Rockwell Automation, Echelon, Hewlett Packard, and Johnson Controls, are aggressively embracing these technologies as fundamental to their future in the new millennium, that is, in the immediate future. The two immobot test beds discussed in this article represent first steps toward these future visions.

Immobot Characteristics

From these examples, we can extract a number of properties that distinguish immobots from their mobile robot and softbot siblings, both in terms of their physical structure and their most salient functions. Structurally, immobots are massively distributed, physically embedded, autonomous systems with a large array of simple, fixed-location sensors and actuators. Functionally, an immobot's primary task is to control its massive regulatory, immune, and nervous systems through a coupling of high-level reasoning and adaptive autonomic processes. More specifically, an immobot has the following distinctive features:

Physically embedded: An immobot's sensors and actuators operate on the physical world, many of which operate in noisy environments. Immobots need to react in real time; however, the time scale of the reaction time can vary dramatically. For example, a spacecraft might have to act instantaneously to close off a leaking thruster but then might have weeks during its cruise to perform a self-diagnosis.

Immobile: An immobot's sensors and actuators reside at fixed locations and are largely limited to one-dimensional signals and one-

degree-of-freedom movement. Examples include light sensors, window-blind actuators, gyroscopes, valves, and reaction wheels. Although an immobot's hardware is often redundant and highly reconfigurable, the primitive elements of interaction that are used to build a configuration are simple and fixed. In contrast, a traditional robot typically has a set of complex mobile sensors and actuators, such as three-dimensional vision systems and articulated hands, arms, and legs.

Omnipresent: The lack of sensor and actuator mobility is compensated for by the sheer number of sensors and actuators. A spacecraft has dozens; a building can have on the order of thousands.

Sensors and actuators must reside a priori in all locations that the immobot wants to model or affect, giving the immobot the potential for being continuously aware of all aspects of its environment. In contrast, a mobile robot's few precious sensors and actuators must be navigated to the point where they are needed. The omnipresence of the immobot shifts the bottleneck from the expense of gathering each piece of information to the integration of a continuous stream of data.

Massively distributed and tightly coupled: Computation is distributed along lengthy communication paths, with increasingly complex computation being performed at sites near sensors and actuators. For example, in *Cassini*, a complex set of physical and software processes work together to fire an engine: The flight computer turns on an engine by sending a command over a bus to a communication processor, which tells a driver to open two valves, which causes fuel to flow into the engine, where combustion occurs. These properties lead to distributed sensing and control problems of high dimensionality as a result of tight couplings between internal components. This high dimensionality makes these problems extremely difficult to solve.

Self-absorbed: Mobile robots and softbots focus largely on what is occurring in the world outside the "bot," including navigating around obstacles, moving through networks to databases, and changing navigation paths because of external failures. However, an immobile robot's attention is largely directed inward toward monitoring the internal health of its network (immunology) and reconfiguring its components or control policies to achieve robust performance (regulation). Although some reasoning is directed outward, the external world is not fluid in

Structurally, immobots are massively distributed, physically embedded, autonomous systems with a large array of simple, fixed-location sensors and actuators. Functionally, an immobot's primary task is to control its massive regulatory, immune, and nervous systems through a coupling of high-level reasoning and adaptive autonomic processes.

Parameter estimation	Adaptive control
Monitoring	Control policy coordination
Mode confirmation	Hardware reconfiguration
Goal tracking	Fault recovery
Detection of anomalies	Standby
Isolation of faults	Safing
Diagnosis of causes	Fault avoidance
Calibration	

Figure 2. Tasks Performed to Maintain the Immobiles Regulatory and Immune Systems.

the same sense that it is for classical robots.

One of a kind: Ironically, what binds together immobile robots is that no two are alike. Drug manufacturing lines, Disney's space mountain, car factories, Mars rovers, Earth-orbiting satellites, deep-space probes, and Antarctic biospheres are each one-of-a-kind devices, making it difficult to amortize their development costs. The dilemma is how to cost effectively build these one of a kinds yet provide high performance and reliability.

Controlling Immobiles

In the preceding sections, we argued that the dominant function of an immobile is to control its massive regulatory, immune, and nervous systems through a coupling of high-level reasoning with adaptive autonomic processes. Providing a regulatory and immune system involves a broad set of tasks, including those listed in figure 2.

Writing software to control the immune and regulatory systems of an immobile is difficult for a variety of reasons: First, it often requires the programmer to reason through systemwide interactions to perform the appropriate function. For example, diagnosing a failed thruster requires reasoning about the interactions between the thrusters, the attitude controller, the star tracker, the bus controller, and the thruster valve electronics. The complexity of these interactions can lead to cognitive overload, causing suboptimal decisions and even outright errors. Furthermore, the one-of-a-kind nature of immobiles means that the cost of reasoning through systemwide interactions cannot be amortized and must be paid over again for each new immobile.

Second, the need for fast reactions in anomalous situations has led to a tradition of precomputing all responses. However, immobiles often operate in harsh environments over decades, so that explicitly enumerating

responses to all possible situations quickly becomes intractable. Tractability is typically restored with the use of simplifying assumptions such as using local suboptimal control laws, assuming single faults, ignoring sensor information, and ignoring subsystem interactions. Unfortunately, this results in systems that are either brittle or grossly inefficient.

Finally, the autonomic processes of an immobile involve a broad range of discrete, continuous, and software behaviors, including control laws, Kalman filters, digital hardware commands, and software drivers. This range of behaviors needed by an immobile makes it difficult and expensive to both manually synthesize high-fidelity autonomic processes and couple these autonomic processes to high-level symbolic reasoning.

Model-Based Autonomous Systems

The goal of our research program is to solve these difficulties by developing a model-based autonomous system kernel for maintaining the regulatory and immune systems of immobiles. The kernel that we are driving toward is defined by three desiderata: (1) model-based programming, (2) model-based reactive execution, and (3) model-based hybrid systems.

Our work on immobiles originated with work on *interaction-based design* (Williams 1990), which explored the coordination and construction of continuous interactions, as a design task that involves the addition of novel hardware-device topologies. In contrast, *model-based autonomy* explores the coordination of hardware and software interactions using a digital controller.

Model-Based Programming

A model-based autonomous system addresses the difficulty of reasoning about systemwide interactions using model-based programming. Model-based programming is based on the idea that the most effective way to amortize software development cost is to make the software plug and play. To support plug and play, immobiles are programmed by specifying component models of hardware and software behaviors. A model-based autonomous system combines component models to automate all reasoning about systemwide interactions necessary to synthesize real-time behaviors such as the ones in figure 2. The development of model libraries is used to reduce design time, facilitate reuse, and amortize modeling costs.

The application of a stringent qualitative

modeling methodology is used to reduce both modeling time and the sensitivity to model inaccuracies and hardware changes. Our experience and those of others (Malik and Struss 1996; Hamscher 1991) has shown that extremely weak qualitative representations, for example, representing only deviations from nominal behavior, are sufficient for many model-based autonomy tasks. Counter to the folklore of the field (Sacks and Doyle 1992), ambiguity and intractable branching has not proven to be a significant practical issue.

Model-based programming on a large scale is supported by developing languages, compilers, debuggers, and visualization tools that incorporate classical concepts of object-oriented, procedural, and hierarchical abstractions into the modeling language.

Model-Based Reactive Execution

The difficulty of precomputing all responses means that a model-based autonomous system must use its models to synthesize timely responses to anomalous and unexpected situations at execution time. Furthermore, the need to respond correctly in time-critical and novel situations means that it must perform deliberative reasoning about the model within the reactive control loop. Although the list of tasks that the model-based execution system must support is seemingly diverse (figure 2), they divide into two basic functions: (1) self-modeling and (2) self-configuring.

Self-modeling: Identifying anomalous and unexpected situations and synthesizing correct responses in a timely manner require an imrobot to be self-modeling. Although parts of its model are provided a priori using model-based programming, other parts need to be adapted or elaborated using sensor information. Self-modeling tasks include tracking model parameters over time (for example, base-line voltages), tracking changes in component behavioral modes (for example, a valve going from open to closed and then to stuck closed), and elaborating quantitative details of qualitative models (for example, learning a quantitative pipe model given that flow is proportional to the pressure drop). Self-modeling tasks are listed in the left-hand column of figure 2; all but estimation, monitoring, and calibration involve identifying discrete behavioral modes of the imrobot's components.

Self-configuring: To provide immune and regulatory systems, an imrobot must be self-configuring; it must dynamically engage and disengage component operating modes and

adaptive control policies in response to changes in goals, the imrobot's internal structure, and the external environment. The right-hand column of figure 2 lists tasks that involve self-configuration. The first two items involve coordinating or adjusting continuous control policies, and the remainder involve changes in component behavioral modes.

Model-Based Hybrid Systems

Given the wide range of digital, analog, and software behaviors exhibited by an imrobot, developing a model-based approach for coordinating the imrobot's autonomic processes requires a rich modeling language and reasoning methods that go well beyond those traditionally used in qualitative and model-based diagnosis. More specifically, a model-based autonomous system must be able to represent and reason about the following:

Concurrent software: Coordinating, invoking, and monitoring real-time software requires formal specifications of their behavior. These behaviors are modeled by incorporating formal specifications of concurrent transition systems into the model-based programming language. Concurrent transition systems provide an adequate formal semantics for most concurrent real-time languages (Manna and Pnueli 1992).

Continuous adaptive processes: Achieving high fidelity requires the merging of symbolic model-based methods with novel adaptive estimation and control techniques (for example, neural nets). Specifications of adaptive processes must be combined with specifications of discrete concurrent behavior within the models. For high-level reasoning systems that coordinate adaptive processes, the most suitable specification of the adaptive processes are often qualitative.

Stochastic processes: Inherent to supporting an imrobot's regulatory and immune functions is the modeling and control of stochastic events that occur within an imrobot's components. This stochastic behavior is modeled by extending the concurrent transition-system model, mentioned previously, to modeling concurrent, partially observable Markov processes.

The previous three subsections outlined the key requirements for model-based autonomy. The essential property that makes these requirements manageable is the relative immobility of our robots. The system interactions are relatively fixed and known a priori through the component models and their interconnection. The flexibility within the system is largely limited to changes in compo-

The goal of our research program is ... developing a model-based autonomous system kernel for maintaining the regulatory and immune systems of imrobots.

ment modes and control policies and adjustments to parameter values. In the rest of this article, we consider two implemented systems that exploit immobility to achieve these desiderata. They form two major components of a kernel we envision for maintaining the regulatory and immune systems of immobots.

Responsive Environments

The scenario in “Spock’s Brain” of a heating and cooling system on a planetary scale highlighted three aspects of immobile robots that we examine technically in this section: First is the task of maintaining a massive, high-performance regulatory system. Second is the need by this system to acquire and adapt accurate models of itself and its environment to achieve high performance. Third is the need for high-level reasoning to coordinate a vast set of autonomic processes during adaptive modeling and regulation.

As a stepping stone toward this ambitious scenario, we developed a test bed for fine-grained sensing and control of a suite of offices at Xerox PARC called the *responsive environment* (Elrod et al. 1993). This test bed includes a networked control system for the complete building plus 15 model offices, each of which has been enhanced with a networked microprocessor that controls an array of sensors and actuators. These include sensors for occupancy, light level, temperature, pressure, and airflow and actuators for regulating air temperature and airflow.

The heart of the building is the central plant, which generates airflow and cold and hot water through a fan, chiller, and boiler, respectively. The extremities of the building are the hundred or more offices, whose temperature must be regulated carefully. The veins and arteries of the building are the pipes and duct work that deliver air and water to the extremities, where they are used to regulate temperature. The connection between the central plant and a single office, used for cooling, is shown in figure 3. Office temperature is controlled through heat flow into, or out of, the office, which includes heat flowing from the sun and equipment, through the walls and doorways, and through the air duct. Heat flow into an office is controlled by the air duct in two ways: First, the amount of airflow is controlled by a damper that partially blocks airflow. Second, the temperature of the airflow is changed by blowing the air over a radiatorlike device, called a *reheat*, that contains hot or cold wa-

ter, where the flow rate of the water is controlled by the reheat-valve position.

Regulating a building is difficult because the number of control variables is overwhelming, and the control variables are highly coupled. For example, the energy consumption of the fan and the chiller are nonlinear functions of the fan speed and the change from outside to inside temperature. The optimal settings of the damper and reheat valve then depend on the outside temperature and the demands that other offices are placing on the chiller and the fan. Hence, the performance of all the offices is coupled, which is exacerbated by the slow response time of office temperature change, making a trial-and-error approach to global control extremely unstable.

We solve this problem with a gradient-descent controller that uses a global model to adaptively predict where the optimum lies. An informal evaluation using the responsive environment test bed suggests that energy savings in excess of 30 percent are conceivable with help from model-based control (Zhang, Williams, and Elrod 1993).

Generic thermal models are available for complete buildings (for example, the DOE2 simulator models) that suffice for this type of control. These models are extremely rich, including not only the building’s hardware and nonlinear thermal characteristics but also sunlight and shading, external weather, and the occupants’ desires. They have on the order of thousands of equations for buildings with 100 or more offices.

The labor-intensive task is tailoring this model to the specifics of a building, which requires estimating numeric values for parameters such as thermal conductance through the walls, heat output of the equipment, and thermal capacity of the office airspace. An im-mobot can estimate its parameters by adjusting their values until the model best fits the sensor data. More precisely, given sensed variables y and x (a vector) and a vector of parameters \mathbf{p} , we first construct an estimator f from the model that predicts y given x and \mathbf{p} :

$$y = f(x; \mathbf{p}) .$$

Next, given a set of (x,y) data D , estimating \mathbf{p} using least squares fit of f to y involves solving the optimization problem

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \sum_{\langle y_i, \mathbf{x}_i \rangle \in D} (y_i - f(\mathbf{x}_i; \mathbf{p}))^2 .$$

Such an estimate is performed by an adaptive numeric algorithm that can be viewed as one of the im-mobot’s autonomic processes.

There are far too many parameters in a

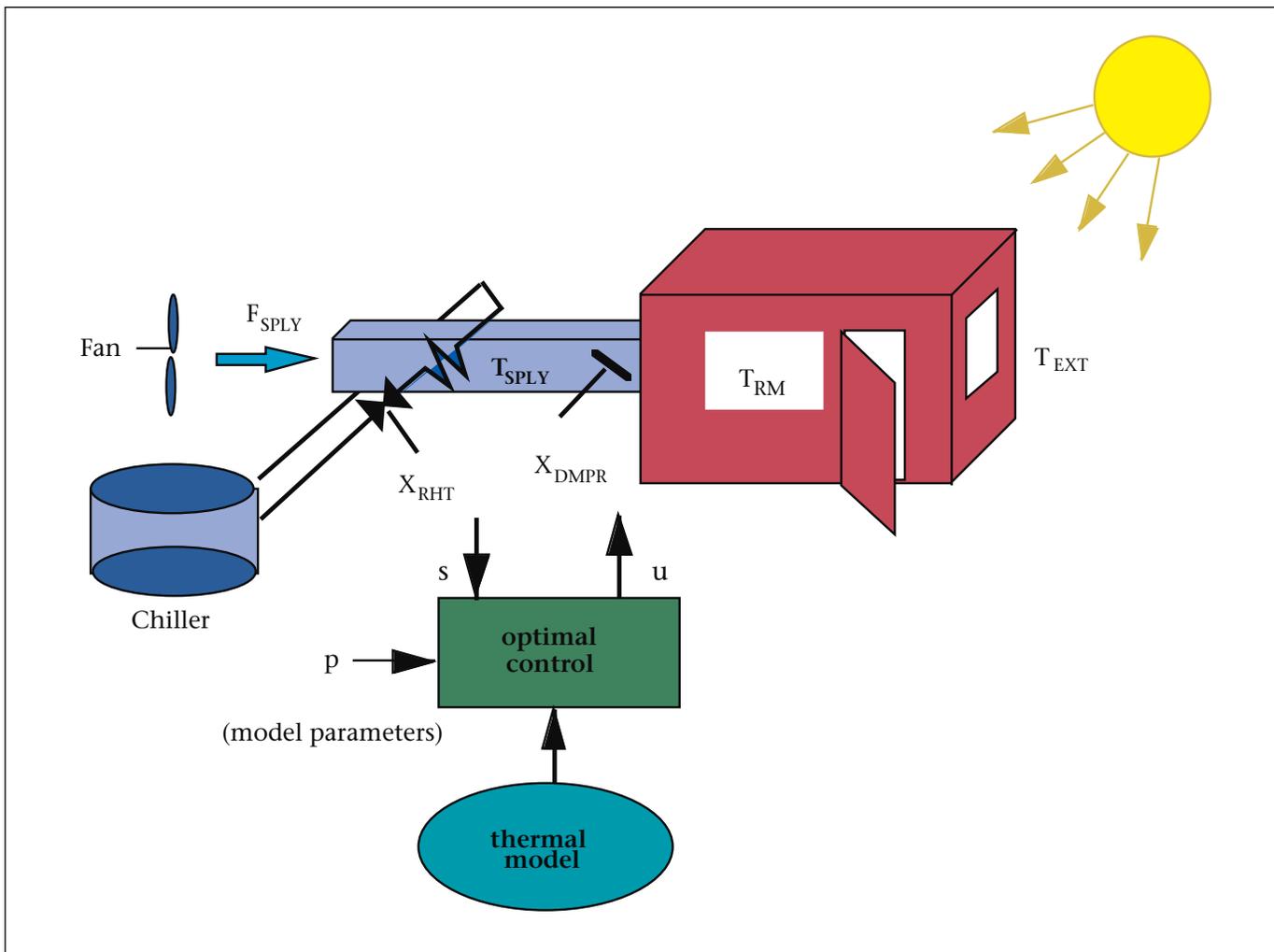


Figure 3. Model-Based Air Conditioning of a Single Office, Using Air Blown over Pipes That Contain a Flow of Chilled Water.

building model to estimate all of them at once. Instead, the immobile robot must automate a control engineer or modeler's expertise at breaking a model into a set of tractable parameter-estimation tasks and then coordinating and combining their results.² This coordination represents the link between high-level reasoning and autonomic processes in our self-modeling immobot. An army of modelers is used for large-scale tasks, such as earth ecosystem modeling, at great cost. This army must be automated for many immobile robot tasks if high performance and robustness are to be achieved. MORIARTY automates key aspects of how a community of modelers decompose, simplify, plan, and coordinate large-scale model-estimation tasks through a technique called *decompositional, model-based learning* (DML).

Coordinating Adaptive Model Estimation

When decomposing a model into a set of estimation tasks, there is often a large set of possible estimators ($y = f(x; p)$) to choose from, and the number of parameters contained in each estimator varies widely. MORIARTY decomposes a model into a set of simplest estimators that minimize the dimensionality of the search space and the number of local minima, hence improving learning rate and accuracy. Each estimator, together with the appropriate subset of sensor data, forms a primitive estimation action. MORIARTY then plans the ordering and the coordination of information flow between them.

To estimate the parameters for a single office, MORIARTY starts with a thermal model

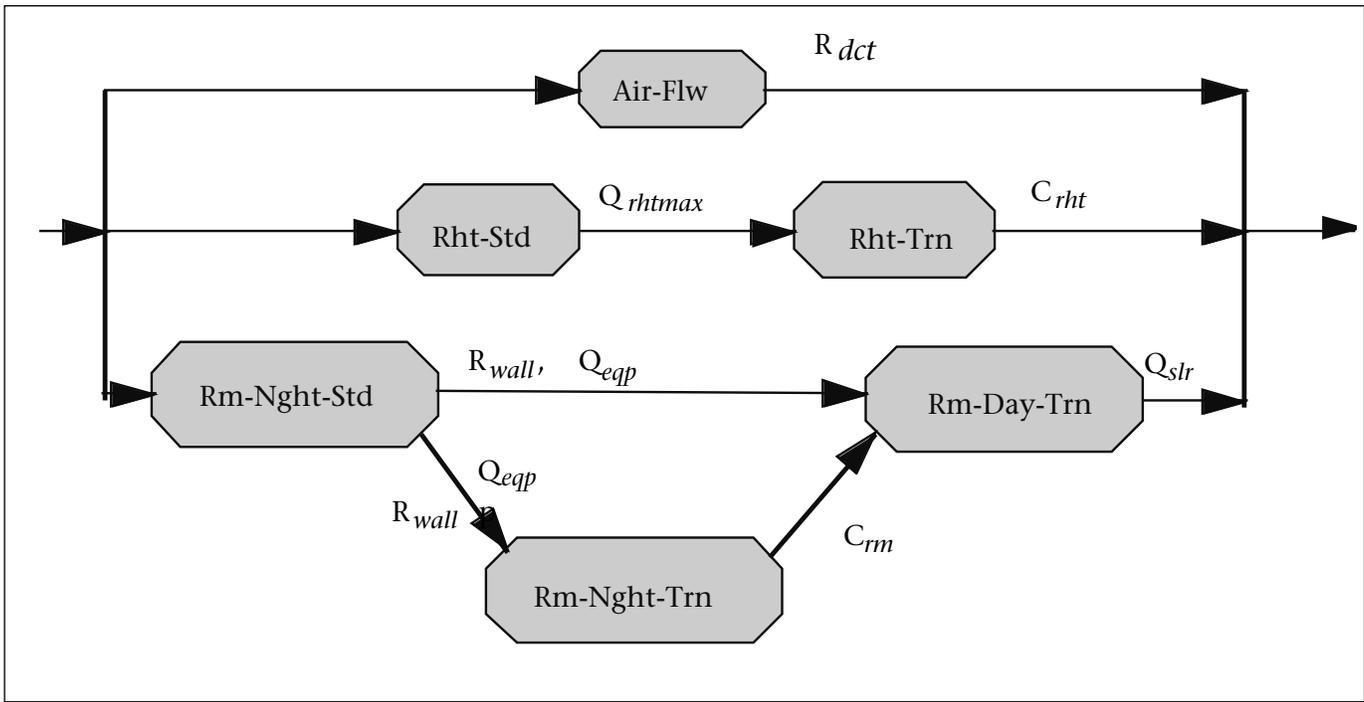


Figure 4. Model-Estimation Plan for a Single Office.

consisting of 14 equations that involve 17 state variables and 11 parameters. About one-third of the equations are nonlinear, such as

$$F_{dmp} = \left(\frac{\rho_{dmp}(X_{dmp})}{R_{dct}} \right) \sqrt{P_{dct}} ,$$

which relates airflow through the damper to duct pressure and duct air resistance as a function of damper position. Nine of the state variables are sensed, including temperature T , flow rate F , air pressure P , and damper and reheat valve position X . Seven of the 11 parameter values are unknown and must be estimated by MORIARTY.

MORIARTY's task is to generate an estimation plan from the thermal model. MORIARTY, when brought to full capability, will be able to generate the plan in figure 4 (it currently generates less efficient plans). In terms of the imobot metaphor, each octagon in the figure represents an adaptive, autonomic estimation process, and the plan graph represents higher-level coordination. Each octagon in the diagram is an estimation action that is defined by an estimator $y = f(x,p)$ applied to a subset of the sensor data. The estimator is constructed from a subset of the model. The arc going into the action specifies additional parameters whose values are already known, and the arc leaving the action specifies parameters whose values have been determined by the estimation. For example, the top octagon, la-

beled *airflow*, produces an estimate for parameter R_{dct} , the air resistance in the duct. It performs this estimate using the following estimator for F_{ext}

$$F_{ext} = (\rho_{lkg} + \rho_{dmp}(X_{dmp})) \sqrt{P_{dct}/R_{dct}} ,$$

which is derived using three equations in the office model pertaining to airflow. This action does not take any parameter values as input (ρ_{lkg} and $\rho_{dmp}(X_{dmp})$ are known a priori as constants).

Consider a commonsense account of how the estimation plan is generated. MORIARTY constructs this plan by generating a set of possible estimation actions from the model. It then selects and sequences a subset of the actions sufficient to cover all parameters. Actions are generated in two steps: In the first step, subsets of the model and sensors are identified that are sufficiently constrained to form an estimation problem. For example, the airflow action described earlier was generated from the pressure P_{dct} and airflow sensors F , together with the following airflow equations:

$$F_{ext} = F_{lkg} + F_{dmp} .$$

$$F_{lkg} = \left(\frac{\rho_{lkg}}{R_{dct}} \right) \sqrt{P_{dct}} .$$

$$F_{dmp} = \left(\frac{\rho_{dmp}(X_{dmp})}{R_{dct}} \right) \sqrt{P_{dct}} .$$

Two additional actions, Rht and Rm, are created from the set of temperature and heat-flow equations by using the duct air-temperature sensor to split the set into reheat and room submodels. The first action, Air-Flw, involves the single parameter R_{dct} , action Rht involves the two parameters Q_{rhtmax} and C_{rht} , and Rm involves the four parameters R_{wall} , Q_{eqp} , C_{rm} , and $Q_{str}(t)$. MORIARTY's first step generates eight possible estimation actions in total, each containing between one and seven parameters. Air-Flw, Rht, and Rm are three actions that cover all seven model parameters and contain the fewest parameters individually. The fact that the sets of parameters in these actions are disjoint is purely coincidental.

An additional step, under development, produces one or more simplified versions of each estimation action by identifying conditions on the data such that influences by one or more parameters become negligible. For example, consider the estimator for action Rm:

$$\frac{dT_{rm}}{dt} = \frac{C_0 F_{sply} (T_{sply} - T_{rm})}{C_{rm}} + \frac{Q_{eqp} + Q_{str}(t) + R_{wall}(T_{ext} - T_{rm})}{C_{rm}} .$$

This estimator can be simplified by noticing that solar effect $Q_{str}(t)$ is negligible at night-time when the sun is down (action Rm-Nght), but it is significant during the day (action Rm-Day-Trn). Action Rm-Nght is generated from Rm by restricting the data set to data taken at night, which allows Q_{str} to be eliminated, reducing the number of parameters in the estimator from four to three: R_{wall} , Q_{eqp} , and C_{rm} .

Furthermore, the effect of the room heat capacity C_{rm} is negligible when the room temperature is in steady state (Rm-Nght-Std), but it becomes particularly significant during room-temperature transients (Rm-Nght-Trn). Hence, action Rm-Nght-Std is generated from the simplified action Rm-Nght by further restricting the data set to data where dT_{rm}/dt is small. Thus, the term $C_{rm}dT_{rm}/dt$ can be eliminated from the estimator, and the number of parameters can be reduced from three to two. Applying this simplification process to the first three of the eight actions generated in the first step results in the six primitive estimation actions shown in figure 4.

Having generated these estimation actions, sequencing exploits the fact that some of these estimation actions share parameters to further reduce the dimensionality of the search performed by each action. In particular, the output of an estimation action with

fewer unknown parameters, such as Rm-Nght-Std, is passed to an overlapping action with more parameters, such as Rm-Nght. Sequencing reduces the number of unknown parameters estimated in the second action; for example, Rm-Nght is left only with C_{rm} as an unknown. This approach produces the sequence shown in figure 4. Each action estimates at most two unknown parameters, a dramatic reduction from the seven unknown parameters in the original problem.

Technically, in the first step, MORIARTY decomposes a model into an initial set of estimation actions by exploiting an analogy to the way in which model-based diagnosis decomposes and solves large-scale multiple-fault problems. The decomposition of a diagnostic problem is based on the concept of a *conflict*—a minimal subset of a model (typically in propositional or first-order logic) that is inconsistent with the set of observations (de Kleer and Williams 1987; Reiter 1987). MORIARTY's decompositional learning method is based on the analogous concept of a *dissent*—a minimal subset of an algebraic model that is overdetermined given a set of sensed variables (that is, a dissent is just sufficient to induce an error function). The set of three flow equations used earlier to construct the Air-Flw estimator is an example of a dissent. Following this analogy, MORIARTY uses a dissent-generation algorithm (Williams and Millar 1996) that parallels the conflict-recognition phase of model-based diagnosis.

MORIARTY's simplification step (currently under development) is based on an order-of-magnitude simplification method called *caricatural modeling* (Williams and Raiman 1994). Caricatural modeling partitions a model into a set of operating regions in which some behaviors dominate, and others become negligible. MORIARTY applies caricatures to each estimation action generated in the first step, using the action's estimator as the model. Each operating region generated corresponds to a simplified estimation action; the operating region's model is the simplified estimator, and the operating region's boundary description provides a filter on data points for the estimator. In the thermal example, one of many simplifications to our estimator for Rm is to minimize $Q_{str}(t)$. Because $Q_{str}(t) \approx 0$ for all data at night, this simplification can be used. In the final step, sequencing selects and orders primitive estimation actions using an algorithm that greedily minimizes the unknown parameters in each estimator, as described in Williams and Millar (1996).

Consider MORIARTY's performance on the ex-

MORIARTY's task is to generate an estimation plan from the thermal model.

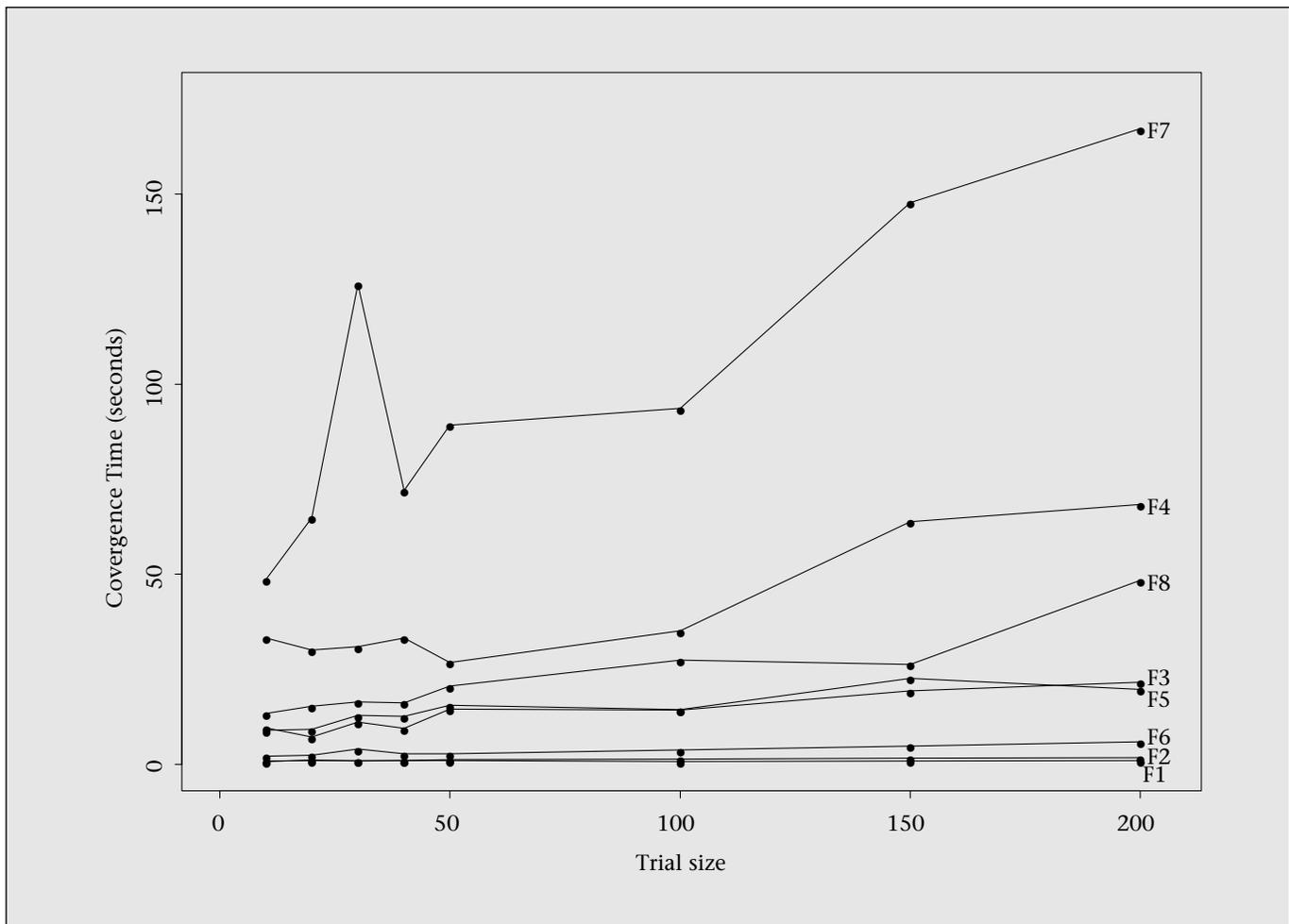


Figure 5. Convergence Rate versus Data Size for the Generated Estimators F1 to F8.

ample, where the simplification step hasn't been performed (figure 5). MORIARTY generates the eight estimators, F1 to F8, in the first step and immediately sequences them to produce $\langle F2, F1, F6 \rangle$, which corresponds to the flow reheat and room-estimation actions, given earlier. We compare the performance of the 8 estimators by running them against sensor data sets ranging in size from 10 to 200, shown previously. The y axis denotes the time required to converge on a final estimate of parameters involved in the dissent. The plot labeled F7 is for the original seven-dimensional estimator, and plots F2, F1, and F6 are the three estimators in MORIARTY's sequence. Higher-dimensional estimators, such as F7, tend to fail to converge given arbitrary initial conditions; hence, ball-park initial parameter estimates were supplied to allow convergence in the higher-dimensional cases. Decomposition leads to significant speedup even when good initial estimates were avail-

able. For example, at trial size 200, the original estimator requires 166 seconds, but the total time to estimate all parameters using F2, F1, and F6 is under 9 seconds, representing a speedup by a factor of 14.

In addition, the sequence requires less data to converge, important for self-modeling immobile robots, which use online estimation to quickly track time-varying parameters. Based on the rule of thumb that the data-set size should roughly be tenfold the dimension of the parameter space, F7 would require around 70 data points, but the F2, F1, F6 sequence requires only 40. The anomalous slowdown in the convergence rate of F7 at 25 data points is attributed to insufficient data. Finally, although not included, parameter accuracy, measured by the confidence interval of each parameter, is also improved using the generated sequence.

To summarize, a model-based approach is essential for regulating systems of the size of

most immobile robots. Embodying an immobile robot with self-modeling capabilities requires the use of symbolic reasoning to coordinate a large set of autonomic estimation processes. This coordination mimics the way in which a community of modelers decomposes, simplifies, and coordinates modeling problems on a grand-challenge scale. DML automates one aspect of this rich model decomposition and analysis planning process.

The New Millennium Program

HAL in 2001 highlights four aspects of the immune system of an immobile robot that we examine technically in this section: First is the ability to monitor internal health, detecting and isolating failing functions, processes, and components. Second is the ability to continually reconfigure the low-level autonomic processes of an imrobot, establishing intended functions and working around failures in a cost-efficient and reliable manner. Third is the ability to reason extensively through reconfiguration options yet ensure reactivity. Fourth is the ability to reason about hybrid systems.

Livingstone

We have developed a system called LIVINGSTONE³ to investigate these issues in the context of NASA's New Millennium Program. LIVINGSTONE is a fast, reactive, model-based configuration manager. Using a hierarchical control metaphor, LIVINGSTONE sits at the nexus between the high-level feed-forward reasoning of classical planning-scheduling systems and the low-level feedback response of continuous adaptive control methods, providing a kernel for model-based autonomy. LIVINGSTONE is distinguished from more traditional robotic executives through the use of deliberative reasoning in the reactive feedback loop. This deliberative reasoning is compositional and model based, can entertain an enormous search space of feasible solutions, yet is extremely efficient because of the ability to quickly focus on the few solutions that are near optimal.

Three technical features of LIVINGSTONE are particularly worth highlighting: First, the approach unifies the dichotomy within AI between deduction and reactivity (Brooks 1991; Agre and Chapman 1987). We achieve a reactive system that performs significant deduction in the sense-response loop by drawing on our past experience at building fast propositional conflict-based algorithms for model-based diagnosis and framing a model-based configuration manager as a propositional

feedback controller that generates focused, optimal responses. Second, LIVINGSTONE's representation formalism achieves broad coverage of hybrid discrete-continuous, software-hardware systems by coupling the concurrent transition-system models underlying concurrent reactive languages (Manna and Pnueli 1992) with the qualitative representations developed in model-based reasoning. Third, the long-held vision of model-based reasoning has been to use a single central model to support a diversity of engineering tasks. For model-based autonomous systems, it means using a single model to support a variety of execution tasks, including tracking planner goals, confirming hardware modes, reconfiguring hardware, detecting anomalies, isolating faults, diagnosing, recovering from faults, and safing. LIVINGSTONE automates all these tasks using a single model and a single-core algorithm, thus making significant progress toward achieving the model-based vision.

Configuration Management

To understand the role of a configuration manager, consider figure 6. It shows a simplified schematic of the main engine subsystem of *Cassini*, the most complex spacecraft built to date. It consists of a helium tank, a fuel tank, an oxidizer tank, a pair of main engines, regulators, latch valves, pyro valves, and pipes. The helium tank pressurizes the two propellant tanks, with the regulators acting to reduce the high helium pressure to a lower working pressure. When propellant paths to a main engine are open, the pressurized tanks force fuel and oxidizer into the main engine, where they combine and spontaneously ignite, producing thrust. The pyro valves can be fired exactly once; that is, they can change state exactly once, either from open to closed or vice versa. Their function is to isolate parts of the main engine subsystem until needed or isolate failed parts. The latch valves are controlled using valve drivers (not shown), and an accelerometer (not shown) senses the thrust generated by the main engines.

To start with the configuration shown in the figure, the high-level goal of producing thrust can be achieved using a variety of different configurations: Thrust can be provided by either main engine, and there are a number of different ways of opening propellant paths to either main engine. For example, thrust can be provided by opening the latch valves leading to the engine on the left or firing a pair of pyros and opening a set of latch valves leading to the engine on the

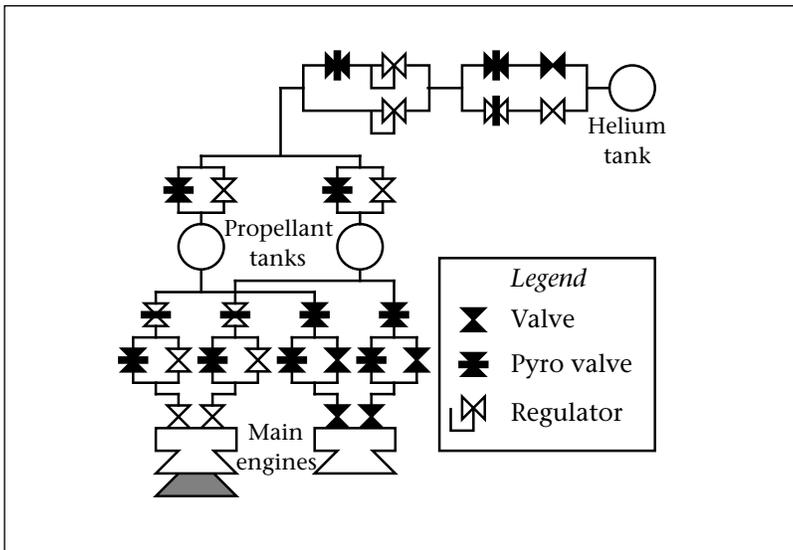


Figure 6. Schematic of the Cassini Main-Engine Subsystem.

In the valve configuration shown, the engine on the left is firing.

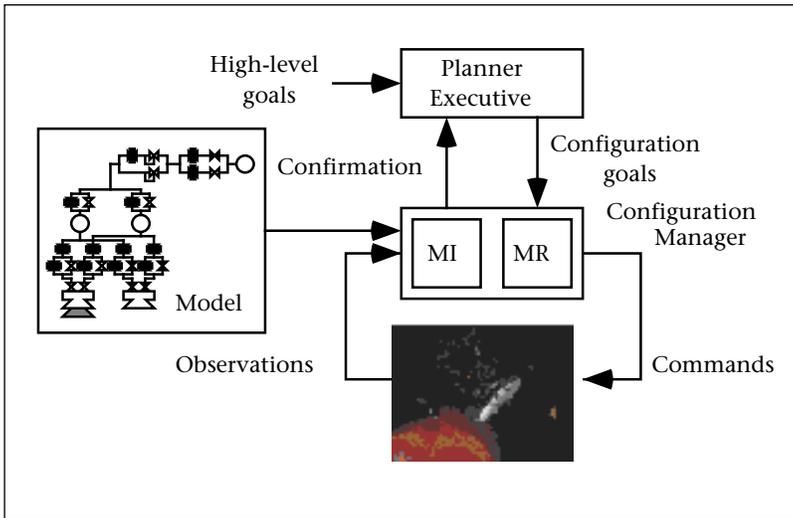


Figure 7. Model-Based Reactive Configuration Management.

right. Other configurations correspond to various combinations of pyro firings. The different configurations have different characteristics because pyro firings are irreversible actions and because firing pyro valves requires significantly more power than opening or closing latch valves.

Suppose that the main-engine subsystem has been configured to provide thrust from the left main engine by opening the

latch valves leading to it. Suppose that this engine fails, for example, by overheating, so that it fails to provide the desired thrust. To ensure that the desired thrust is provided even in this situation, the spacecraft must be transitioned to a new configuration in which thrust is now provided by the main engine on the right. Ideally, thrust is provided by firing the two pyro valves leading to the right side and opening the remaining latch

valves (rather than firing additional pyro valves).

A configuration manager constantly attempts to move the spacecraft into lowest-cost configurations that achieve a set of high-level dynamically changing goals, such as the goal of providing nominal thrust. When the spacecraft strays from the chosen configuration because of failures, the configuration manager analyzes sensor data to identify the current configuration of the spacecraft and then moves the spacecraft to a new configuration, which, once again, achieves the desired configuration goals. In this sense, a configuration manager such as LIVINGSTONE is a discrete control system that is strategically situated between high-level planning and low-level control; it ensures that the spacecraft's configuration always achieves the set point defined by the configuration goals.

Model-Based Configuration Management

LIVINGSTONE is a reactive configuration manager that uses a compositional, component-based model of the spacecraft to determine configuration actions (figure 7). Each component is modeled as a transition system that specifies the behaviors of operating and failure modes of the component, nominal and failure transitions between modes, and the costs and likelihoods of transitions (figure 8). Mode behaviors are specified using formulas in propositional logic, but transitions between modes are specified using formulas in a restricted temporal, propositional logic. The restricted propositional, temporal logic is adequate for modeling digital hardware, analog hardware using qualitative abstractions (de Kleer and Williams 1991; Weld and de Kleer 1990), and real-time software using the models of concurrent reactive systems in Manna and Pnueli (1992). The spacecraft transition-system model is a composition of its component transition systems in which the set of configurations of the spacecraft is the cross-product of the sets of component modes. We assume that the component transition systems operate synchronously; that is, for each spacecraft transition, every component performs a transition.

A model-based configuration manager uses its transition-system model to both identify the current configuration of the spacecraft, called *mode identification* (MI), and move the spacecraft into a new configuration that achieves the desired configuration goals, called *mode reconfiguration* (MR). MI incrementally generates all spacecraft transitions

from the previous configuration such that the models of the resulting configurations are consistent with the current observations (figure 9). MR determines the commands to be sent to the spacecraft such that the resulting transitions put the spacecraft into a configuration that achieves the configuration goal in the next state (figure 10). The use of a spacecraft model in both MI and MR ensures that configuration goals are achieved correctly.

Both MI and MR are reactive. MI infers the current configuration from knowledge of the previous configuration and current observations. MR only considers commands that achieve the configuration goal in the next state. Given these commitments, the decision to model component transitions as synchronous is key. An alternative is to model multiple component transitions through interleaving. However, such interleaving can place an arbitrary distance between the current configuration and a goal configuration, defeating the desire to limit inference to a small fixed number of states. Hence, we model component transitions as being synchronous. If component transitions in the underlying hardware-software are not synchronous, our modeling assumption is still correct as long as some interleaving of transitions achieves the desired configuration.

In practice, MI and MR need not generate all transitions and control commands, respectively. Rather, just the most likely transitions and an optimal control command are required. We efficiently generate these by recasting MI and MR as combinatorial optimization problems. In this reformulation, MI incrementally tracks the likely spacecraft trajectories by always extending the trajectories leading to the current configurations by the most likely transitions. MR then identifies the command with the lowest expected cost that transitions from the likely current configurations to a configuration that achieves the desired goal. We efficiently solve these combinatorial optimization problems using a conflict-directed best-first search algorithm. See Williams and Nayak (1996) for a formal characterization of MI and MR and a description of the search algorithm.

A Quick Trip to Saturn

Following the announcement of the New Millennium Program in early 1995, spacecraft engineers from the Jet Propulsion Laboratory (JPL) challenged a group of AI researchers at NASA Ames and JPL to demonstrate, within the short span of five

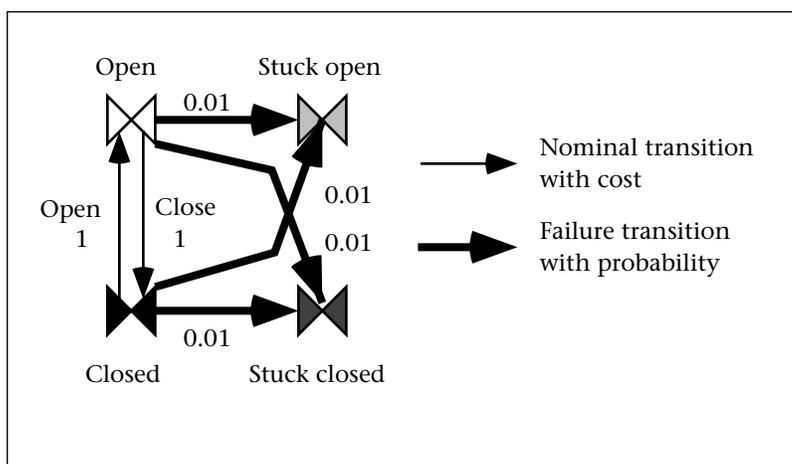


Figure 8. Transition-System Model of a Valve.

Open and closed are normal operating modes, but stuck open and stuck closed are failure modes. The Open command has unit cost and causes a mode transition from closed to open, similarly for the Close command. Failure transitions move the valve from the normal operating modes to one of the failure modes with probability 0.01.

months, a fully autonomous architecture for spacecraft control. To evaluate the architecture, the JPL engineers defined the NEWMAAP spacecraft and scenario based on *Cassini*. The NEWMAAP spacecraft is a scaled-down version of *Cassini* that retains the most challenging aspects of spacecraft control. The NEWMAAP scenario is based on the most complex mission phase of *Cassini*—successful insertion into Saturn’s orbit even in the event of any single point of failure.

The AI researchers, working closely with the spacecraft engineers, developed an autonomous agent architecture that integrates LIVINGSTONE with the HSTS planning and scheduling system (Mussettola 1994) and a multithreaded smart executive (Pell, Gat, et al. 1996) based on RAPS (Firby 1995). In this architecture (see Pell, Bernard, et al. [1996] for details), HSTS translates high-level goals into partially ordered tokens on resource time lines. The executive executes planner tokens by translating them into low-level spacecraft commands while enforcing temporal constraints between tokens. LIVINGSTONE tracks spacecraft state and planner tokens and reconfigures for failed tokens. This autonomous agent architecture was demonstrated to successfully navigate the simulated NEWMAAP spacecraft into Saturn orbit during its one-hour insertion window, despite about a half-dozen failures. Consequently, LIVINGSTONE, HSTS, and the smart executive have been selected to fly *Deep Space One*, forming the core autonomy architecture of NASA’s New Millennium Program.

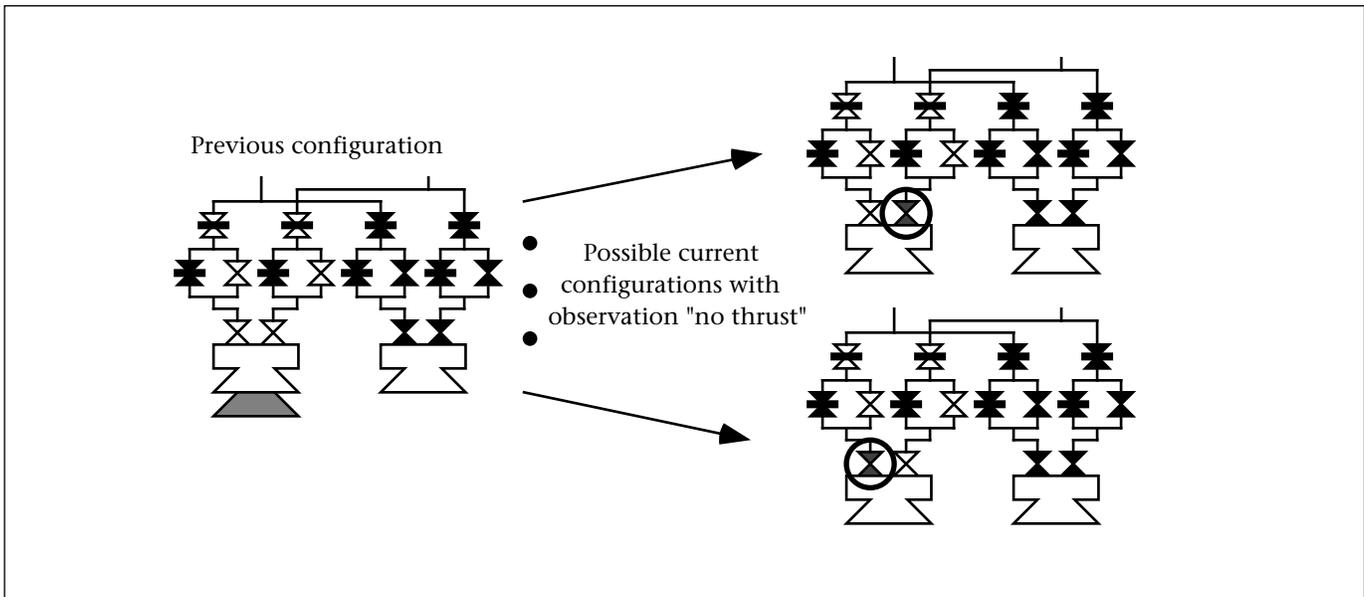


Figure 9. Mode Identification (MI).

The figure shows a situation in which the left engine is firing normally in the previous state, but no thrust is observed in the current state. MI's task is to identify the configurations into which the spacecraft has transitioned that account for this observation. The figure shows two possible transitions, corresponding to one of the main-engine valves failing "stuck closed" (failed valves are circled). Many other transitions, including more unlikely double faults, can also account for the observations.

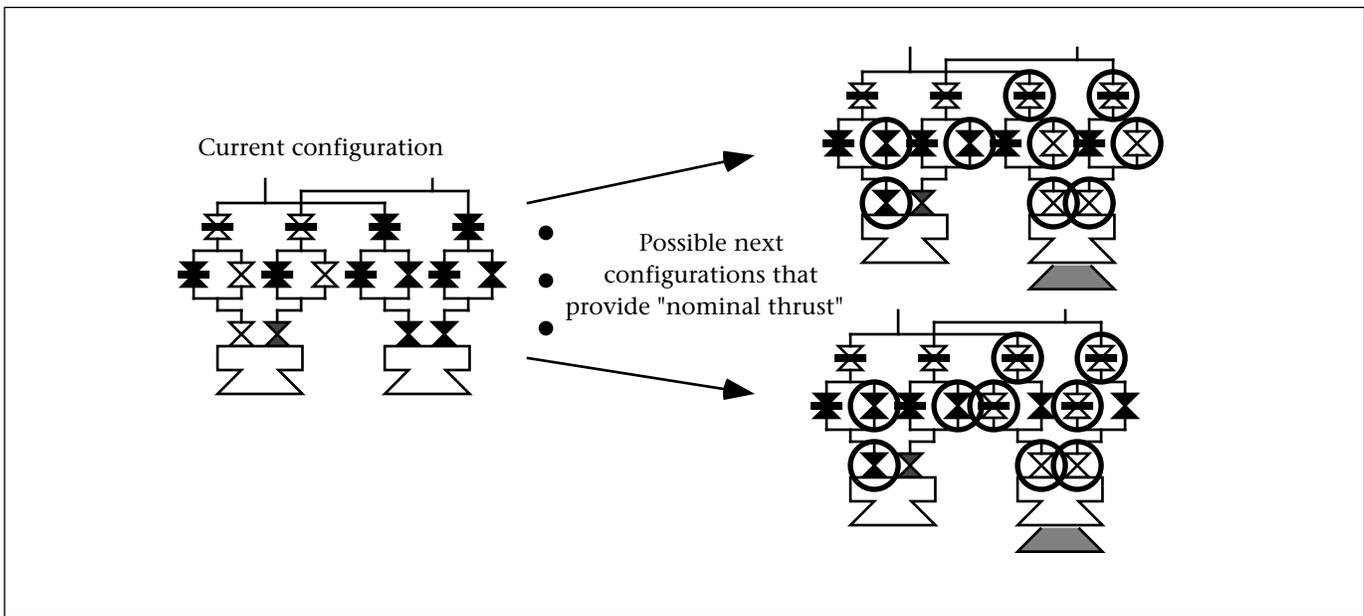


Figure 10. Mode Reconfiguration (MR).

The figure shows a situation in which mode identification (MI) has identified a failed main-engine valve leading to the left main engine. MR reasons that normal thrust can be restored in the next state if an appropriate set of valves leading to the right engine are opened. The figure shows two of the many configurations that achieve the desired goal (circled valves are commanded to change state). Transitioning to the configuration at the top has lower cost because only necessary pyro valves are fired. The valves leading to the left engine are turned off to satisfy a constraint that, at most, one engine can fire at one time.

Table 1 provides summary information about LIVINGSTONE's model of the NEWMAAP spacecraft, demonstrating its complexity. The NEWMAAP demonstration included seven failure scenarios. From LIVINGSTONE's viewpoint, each scenario required identifying likely failure transitions using MI and deciding on a set of control commands to recover from the failure using MR. Table 2 shows the results of running LIVINGSTONE on these scenarios.

The first column in table 2 names each of the scenarios, but a discussion of the details of these scenarios is beyond the scope of this article. The second and fifth columns show the number of solutions checked by MI and MR, respectively. One can see that even though the spacecraft model is large, the use of conflict-directed search dramatically focuses the search. The third column shows the number of leading trajectory extensions identified by MI. The limited sensing available on the NEWMAAP spacecraft often makes it impossible to identify unique trajectories. The fourth and sixth columns show the time spent by MI and MR on each scenario, once again demonstrating the efficiency of our approach.

The New Millennium

We are only now becoming aware of the rapid construction of a ubiquitous, immobile robot infrastructure that rivals the construction of the World Wide Web and has the potential for profound social, economic, and environmental change. Tapping into this potential will require embodying immobots with sophisticated regulatory and immune systems that accurately and robustly control their complex internal functions. Developing these systems requires fundamental advances in model-based autonomous system architectures that are self-modeling, self-configuring, and model based programmable and support deliberated reactions. This development can only be accomplished through a coupling of the diverse set of high-level, symbolic methods and adaptive autonomic methods offered by AI. Although a mere glimmer of Spock and HAL, our two model-based immobots, LIVINGSTONE and MORIARTY, provide seeds for an exciting new millennium.

Acknowledgments

We would like to thank the many individuals who have influenced this work. Oren Etzioni's provocative 1992 AAAI Spring Symposium talk on softbots provided a catalyst. Thanks to the responsive environment team, particularly Bill Millar, Joseph O'Sullivan, and

Table 1. NEWMAAP Spacecraft Model Properties.

Number of components	80
Average number of modes per component	3.5
Number of propositions	3,424
Number of clauses	11,101

Table 2. Results from the Seven NEWMAAP Failure-Recovery Scenarios.

Time is in seconds on a SPARC 5.

Scenario	MI			MR	
	Check	Accept	Time	Check	Time
EGA preaim failure	7	2	2.2	4	1.7
BPLVD failed	5	2	2.7	8	2.9
IRU failed	4	2	1.5	4	1.6
EGA burn failure	7	2	2.2	11	3.6
Acc failed	4	2	2.5	5	1.9
ME too hot	6	2	2.4	13	3.8
Acc low	16	3	5.5	20	6.1

Ying Zhang. Thanks to our remote-agent collaborators, including James Kurien, Nicola Muscettola, Barney Pell, Greg Swietek, Douglas Bernard, Steve Chien, Erann Gat, and Reid Simmons. Finally, thanks to Yvonne Clearwater for help with the graphics.

Notes

1. This article is based on an invited talk given at the Third International Conference on AI Planning Systems (Williams 1996).
2. To achieve tractability, a modeler of nonlinear physical systems typically searches for estimators containing four or less parameters.
3. LIVINGSTONE the program is named after David Livingstone (1813–1873), the nineteenth-century medical missionary and explorer. Like David Livingstone, LIVINGSTONE the program is concerned with exploration and the health of explorers.

References

- Agre, P., and Chapman, D. 1987. PENG: An Implementation of a Theory of Activity. In Proceedings of the Sixth National Conference on Artificial Intelligence, 268–272. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Brooks, R. A. 1991. Intelligence without Reason. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 569–595. Menlo

- Park, Calif.: International Joint Conferences on Artificial Intelligence.
- de Kleer, J., and Williams, B. C., eds. 1991. *Artificial Intelligence*, Volume 51. New York: Elsevier.
- de Kleer, J., and Williams, B. C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence* 32(1): 97–130.
- Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zabowski, D. 1992. A Personal Learning Apprentice. In Proceedings of the Tenth National Conference on Artificial Intelligence, 96–103. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Elrod, S.; Hall, G.; Costana, R.; Dixon, M.; and des Rivieres, J. 1993. Responsive Office Environments. *Communications of the ACM* 36(7): 84–85.
- Etzioni, O., and Segal, R. 1992. Softbots as Testbeds for Machine Learning. Paper Presented at 1992 AAAI Spring Symposium on Knowledge Assimilation, 25–27 March, Stanford, California.
- Etzioni, O., and Weld, D. 1994. A Softbot-Based Interface to the Internet. *Communications of the ACM* 37(7): 72–79.
- Firby, R. J. 1995. The RAP Language Manual. Animate Agent Project Working Note, AAP-6, University of Chicago.
- Hamscher, W. C. 1991. Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence* 51:223–271.
- Kautz, H.; Selman, B.; Coen, M.; Ketchpel, S.; and Ramming, C. 1994. An Experiment in the Design of Software Agents. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 438–443. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Knoblock, C. A., and Levy, A. Y., eds. 1995. Paper presented at the AAAI Spring Symposium Series on Information Gathering in Distributed and Heterogeneous Environments, 27–29 March, Stanford, California.
- Levy, A. Y.; Rajaraman, A.; and Ordille, J. J. 1996. Query-Answering Algorithms for Information Agents. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 40–47. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Maes, P., and Kozierok, R. 1993. Learning Interface Agents. In Proceedings of the Eleventh National Conference on Artificial Intelligence, 459–465. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Malik, A., and Struss, P. 1996. Diagnosis of Dynamic Systems Does Not Necessarily Require Simulation. In Proceedings of the Tenth International Workshop on Qualitative Reasoning, 127–136. AAAI Technical Report WS-96-01. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Manna, Z., and Pnueli, A. 1992. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. New York: Springer-Verlag.
- Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In *Intelligent Scheduling*, eds. M. Fox and M. Zweben, 169–212. San Francisco, Calif.: Morgan Kaufmann.
- Pell, B.; Gat, E.; Keesing, R.; and Muscettola, N. 1996. Plan Execution for Autonomous Spacecraft. Paper presented at the AAAI Fall Symposium on Plan Execution, 9–11 November, Cambridge, Massachusetts.
- Pell, B.; Bernard, D. E.; Chien, S. A.; Gat, E.; Muscettola, N.; Nayak, P. P.; Wagner, M. D.; and Williams, B. C. 1996. A Remote-Agent Prototype for Spacecraft Autonomy. In Proceedings of the SPIE Conference on Optical Science, Engineering, and Instrumentation, Volume on Space Spacecraft Control and Tracking in the New Millennium. Bellingham, Wash.: Society of Professional Image Engineers.
- Rasmussen, R. 1996. Personal communication, National Aeronautics and Space Administration, Washington, D.C.
- Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artificial Intelligence* 32(1): 57–96.
- Sacks, E. P., and Doyle, J. 1992. Prolegomena to Any Future Qualitative Physics. *Computational Intelligence* 8(2): 187–209.
- Weld, D. S., and de Kleer, J., eds. 1990. *Readings in Qualitative Reasoning about Physical Systems*. San Francisco, Calif.: Morgan Kaufmann.
- Williams, B. C. 1996. Model-Based Autonomous Systems in the New Millennium. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, 275–282. Menlo Park, Calif.: AAAI Press.
- Williams, B. C. 1990. Interaction-Based Invention: Designing Novel Devices from First Principles. In Proceedings of the Eighth National Conference on Artificial Intelligence, 349–356. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Williams, B. C., and Millar, B. 1996. Automated Decomposition of Model-Based Learning Problems. In Proceedings of the Tenth International Workshop on Qualitative Reasoning, 265–273. AAAI Technical Report WS-96-01. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Williams, B. C., and Nayak, P. P. 1996. A Model-Based Approach to Reactive Self-Configuring Systems. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 971–978. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Williams, B. C., and Raiman, O. 1994. Compositional Modeling through Caricatural Reasoning. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 1199–1204. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Zhang, Y.; Williams, B. C.; and Elrod, S. 1994. Model Estimation and Energy-Efficient Control for Building Management Systems. SPL Technical Report, Xerox PARC, Palo Alto, California.

JAI Press Ad



Brian Williams is a research scientist in the Computational Sciences Division of the NASA Ames Research Center. He is co-lead of the Model-Based Autonomous Systems Project and a flight lead for the New Millennium Deep-Space One spacecraft. He received his professional degrees from the

Massachusetts Institute of Technology—an SB in electrical engineering in 1981, and SM in computer science in 1984, and a Ph.D. in 1989. He was a member of the technical staff at Xerox PARC until 1994. His current research concentrates on the development and formalization of model-based autonomous systems for controlling immobile robots, including deep-space probes and smart buildings. Research interests include model-based information gathering, ranging from diagnosis to large-scale exploratory modeling and data analysis, hybrid algorithms that use symbolic reasoning to coordinate adaptive methods, and formal theories of abstraction and approximation.



Pandurang Nayak is a research scientist in the Computational Sciences Division of the NASA Ames Research Center. He is co-lead of the Model-Based Autonomous Systems Project and a member of the New Millennium Deep-Space One flight software team. He received a B.Tech. in

computer science and engineering from the Indian Institute of Technology, Bombay, in 1985, and a Ph.D. in computer science from Stanford University in 1992. His research interests include abstractions and approximations in knowledge representation and reasoning, the building of model-based autonomous systems, diagnosis and recovery, and qualitative and causal reasoning.