Model-based Autonomous Systems in the New Millennium

Brian C. Williams
Recom Technologies
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94305 USA
E-mail: `williams@ptolemy.arc.nasa.gov`

Abstract

A new generation of sensor rich, massively distributed, autonomous systems is being developed that has the potential for unprecedented performance, such as networks of smart buildings, reconfigurable factories, spacecraft constellations and remote earth ecosystem monitoring. To achieve high performance these *immobile robots* will need to use their sensors to accurately model themselves and their environment on a grand scale. They will use these models to reconfigure themselves dramatically if they are to survive decades of autonomous operations. To be economically viable they will need to be programmable purely through high level compositional models. Self modeling, self configuration, deliberated reactions and compositional, model-based programming are the four key elements of a *model-based autonomous systems* architecture that is taking us into the *New Millenium.*

Introduction

The lime-light has shifted dramatically, in the last few years, from an AI tradition of developing mobile robots, to that of software agents (aka softbots) (Dent *et al.* 1992; Maes & Kozierok 1993; Kautz *et al.* 1994). Meanwhile the media has supplanted R2D2 with punk descendants of Tron as the popular icon. One motivation for this shift is the difficulty and cost of developing and maintaining physical robots, which are believed to substantially impede progress towards AI's central goals, such as a theory of machine intelligence and agent architectures (Etzioni & 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 (e.g., **ls** and **gopher**) and end effectors (e.g., **ftp** and **telnet**) that are easy to maintain, while still providing a testbed for exploring issues of mobility and realtime constraints. At the same time the recent Internet gold rush and the ensuing web literacy has provided an enormous textual corpus which screams for intelligent information gathering aides.

Yet two concerns have been raised. First, do these software environments provide many of the rich constraints that seem fundamental to physical environments, such as the need to interpret sensors receiving continuous, noisy signals, or the need to control end effectors with nonlinear characteristics? If not how do we drive research on agent architectures that address these key constraints? Second, will these information gathering software agents be able to significantly relieve the human information overload problem, without solving the natural language understanding problem? If not, then what will be the scope of applicability of agents on the Internet, until this hard nut is cracked?

In this paper we argue that the information gathering capabilities of the Internet, and smaller networked computational systems, supply testbeds of a very different sort. These testbeds, which we call *immobile robots*, have the richness that comes from interacting with physical environments, are becoming readily available, and complement pure software environments. Their application will be a driving force for profound social, environmental, and economic change.

Furthermore, we argue that these *immobots* give rise to a new family of agent architectures, called *model-based autonomous systems*, that fuse together research in such diverse areas as model-based reasoning, qualitative reasoning, planning, scheduling and execution, propositional satisfiability, concurrent reactive languages, and adaptive systems. This paper highlights four properties of a model-based autonomous system that are essential: self-modeling, self-configuration, deliberated reactions and model-based programming. These properties are embodied in two systems, *Moriarty* and *Livingstone,* described technically in (Williams & Millar 1996) and (Williams & Nayak 1996), respectively.

This paper grounds the model-based autonomous systems approach through two immobile robot testbeds, which the author has worked with over the last three years. The first testbed is the *Responsive Environment* (Zhang, Williams, & Elrod 1993; Elrod & et al. 1993), an intelligent building control system, which was part of the Ubiquitous Computing project at Xerox PARC. The second is the *Remote Agent*, a goal-directed, fully autonomous control architecture, which will fly the Deep Space One space probe in 1998 (Pell *et al.* 1996). At least one of these immobots has the promise of significant impact at the very beginning of the New Millennium.

Immobile Robots

Hardware advances in cheap single chip control and communication processors, sensors, point actuators, A to Ds, and networking has enabled a new category of autonomous system that is sensor rich, massively distributed, and largely immobile. These immobile robots are rapidly being deployed in the form of networked building energy systems, chemical plant control networks, the earth observing system, power grids, biosphere life support systems, satellite constellations, reconfigurable traffic systems, factories, telescope farms, networked space probes, planes, automobiles and copiers – to highlight a few.

The current generation of these hybrid hardware/software systems has created a slumbering giant whose potential has only begun to be tapped. For example, networked building energy management systems are enabling energy reductions well over 30%. For example, a building like Xerox PARC saw a one year cost reduction of a quarter million dollars, by exploiting a networked building management system. The earth observing system is moving towards a level of sensing that will enable full earth ecosystem modeling, providing insight into problems of pollution, global warming and ozone depletion. Constellations of autonomous space probes, communicating by the deep space network, will enable a virtual presence throughout the solar system and beyond. These are not far off dreams. Many government and private institutions, such as NASA, PG&E, Rockwell Automation, Echelon and Johnson Controls, are aggressively embracing these technologies as fundamental to their future in the new millennium, that is, in the immediate future.

There are several distinctive properties of these testbeds that make them a rich set of testbeds for autonomous architectures:

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 timescale of the reaction time can vary dramatically, depending on the context, from microseconds to days. For example, a spacecraft may have to act instantaneously to close off a leaking thruster, but then may have weeks during its cruise to perform a self diagnosis. On the other hand it may miss its orbital insertion if it can't execute a complex recovery within minutes.

Immobile An immobot's sensors and actuators are largely limited to one dimensional signals and one degree of freedom movement, for example, light sensors, window blind actuators and engine thrusters. While an immobot's hardware is often highly reconfigurable, the primitive elements of interaction that are used to build a configuration are fixed. Some immobile robots, like spacecraft, do move in three space; however, issues, central to mobots and robots, of navigating around complex obstacles or recognizing 3D objects are not prevalent.

(Massively) Distributed What is lacking in the sophistication of the individual sensors, is made up for by their sheer numbers. A spacecraft has dozens of sensors and actuators, a building can have on the order of thousands, and if intelligently coordinated on a power grid, the scale is millions. Computation is distributed along the communication paths out to the end effectors, and incorporates complex computation at the sites of the sensors and actuators. This leads to distributed sensing and control problems of extremely high dimension.

Self-Absorbed Robots, mobots and softbots focus largely on what is occurring in the world outside the "bot," navigating around obstacles, moving through networks to databases, and changing navigation paths due to external failures. Because an immobile robot's computation is distributed out towards its end effectors, establishing an optimal path of observation and control involves a complex reasoning task. Hence an immobile robot's attention is largely inward directed, monitoring the internal health of its network, and reconfiguring its components or control policies to achieve robust performance. While some reasoning is directed outward, the external world is not fluid in the same sense as for classical robots.

Heterogenous Ironically what binds together immobile robots is that no two are alike. Drug assembly lines, Disney's space mountain, car factories, mars rovers, earth orbiting satellites, deep space probes, and Antartic biospheres, are distinctively different, yet the first three are products of one corporation and the rest are of one agency. The dilemma is how to cost effectively build these one of a kinds, while exploiting their potential.

Hybrid Discrete/Continuous Heterogeneity carries over into the forms of computation involved, mixing discrete concurrent transition systems with continuous adaptive estimation and control methods, not to mention ladder logic, petri nets, and a myriad of other methods. The hybrid of discrete and continuous methods is shared with most robots and mobots, but is further accentuated due to the extreme size and heterogeneity.

While these massively distributed control systems are just beginning to capture AI's imagination as worthy of "bot" status, they were very much a part of the media's vision of the robotic future back in the 60's. This includes such movies as *2001: A Space Odyssey*, *The Forbin Project*, *The Andromeda Strain* and, of course, the classic Star Trek episode, *Spock's Brain*. The most famous computational intelligence, HAL 9000 was clearly an immobile robot, controlling all aspects of the spacecraft's control and life support. His cleverness at protecting his critical resources came across clearly in the scene outside the spacecraft, where Dave says "Open the pod bay door, Hal," and Hal, pausing a moment in careful thought, says "Sorry Dave, but I can't do that." The most famous biological intelligence, Spock, also became an immobile robot. In the episode "Spock's Brain," Spock's body is found robbed of his brain by unknown aliens. His brain wakes up only to find that his new body is pumping fluids that coarse through veins and arteries encompassing a planet. That is, the function that requires this supreme intelligence is the control of a planetary heating and ventilation system. Our two testbeds for model-based autonomy are the first steps towards these two future visions from the recent past.

Model-based Autonomous Systems

We propose that the unique characteristics of immobile robots requires a significant departure from past robotic architectures:

Self Configuration Immobile robots, such as interplanetary explorers, must operate robustly in harsh environments over decades, while others, such as buildings and closed-loop martian habitats, must achieve extraordinary levels of efficiency. To accomplish this they must be *self-configuring*, dynamically engaging and disengaging both components and adaptive control policies.

Self-modeling The massive size of some immobots and the corresponding high dimensionality of the control space dictates a model-based control approach. For example, models are necessary to achieve quick convergence, and first time correct behaviors. Discrete and continuous models (e.g., failure modes, model parameters and novel behaviors) must be adapted and sometimes acquired from sensor information, but the high dimensionality and the paucity of data in some cases requires a strong bias from a prior model.

Model-based Programming An immobot's ability to reconfigure in response to global goals, and to self model from limited observables, requires extensive reasoning about system wide interactions. Given the heterogeneity and size of these immobots, this reasoning cannot be performed apriori and hand coded through the control software. Instead, the immobots should be quickly assembled together from components, and the control software automatically generated from component-based models. This requires a new paradigm in compositional, model-based programming.

Deliberated Reactions The observation that "if you see Pluto, you've gone to far," while correct is not particularly useful in spacecraft autonomy. The need to respond correctly in time critical, novel situations requires deliberative reasoning within the reactive control loops of immobile robots. The novelty of the situations and the size of the possible events encountered frequently prevents performing this reasoning before the event occurs.

These are the four key requirements for model-based autonomy. The essential property that makes them manageable is the relative immobility of our robots. In the rest of this paper we consider two systems that exploit immobility to achieve these desiderata.

Responsive Environments

The concept in "Spock's brain" of heating and cooling systems on a vast scale is not as implausible as it might seem. In fact it is quickly becoming a reality through a confluence of four forces: The broad installation of the new generation of networked building management systems, the interconnection of building and home utilities through optical fibers, the deregulation of the utlilities, and the ensuing establishment of computer-based energy markets. These forces are assembling an immobile robot on the same scale as the world wide web.

Based on Xerox PARC studies it is by no means inconceivable that, with the help of model-based adaptive methods, these systems will see energy reductions of over 60%, plus improvements in air quality and individual comfort. But to achieve high performance these massive systems will need to accurately model themselves and their environment from sensor information. The modeling task is extremely rich, since the environment includes not only the hardware and the building's nonlinear thermal characteristics, but sunlight and shading, external weather, and the occupants desires and habits.

Despite their potential, the labor and skill involved in applying model-based adaptive methods is economically infeasible for most large scale modeling, learning and control problems. One of our primary goals in model-based autonomy is to automate the expertise embodied by a skilled community of modelers at decomposing, planning and coordinating large scale model estimation or learning tasks, and to embody them within model-based autonomous systems. To this end we are developing a large scale automated modeling approach we call *decompositional, model-based learning (DML)*, and its implementation, called *Moriarty*. Moriarty is the continuous, self-modeling component of our model-based autonomy architecture, complementing *Livingstone's*, discrete self-modeling and self-configuration (Williams & Nayak 1996) capabilities, discussed later.

Our work on DML was developed in the context of synthesizing an optimal heating and cooling control system for a smart, self-modeling building. To study this synthesis process we built a testbed for fine grained sensing and control of a building, the responsive environment (Elrod & et al. 1993), and used this testbed to study the manual art of our control engineers at decomposing the model estimation and optimal control components of the overall control problem (Zhang, Williams, & Elrod 1993).

For example, looking at a fragment of the modeling problem for the responsive environment, an office's energy and mass flow (heat, air and water) is modeled by a vector $\mathbf{e}(\mathbf{c};\mathbf{v})$ of 14 equations involving seventeen state variables $\mathbf{v}$:

$$\begin{aligned}
F_{ext} &= F_{sply} && (1)\\
F_{sply} &= F_{rtrn} && (2)\\
Q_{ext} &= C_0 F_{ext} T_{ext} && (3)\\
Q_{sply} &= C_0 F_{sply} T_{sply} && (4)\\
Q_{rtrn} &= C_0 F_{rtrn} T_{rm} && (5)\\
Q_{rhtcap} &= Q_{ext} - Q_{sply} + Q_{rht} && (6)\\
Q_{rhtcap} &= C_{rht} \frac{dT_{sply}}{dt} && (7)\\
Q_{rht} &= \left(\frac{Q_{rhtmax}}{X_{rhtmax}}\right) X_{rht} && (8)\\
F_{ext} &= F_{lkg} + F_{dmpr} && (9)\\
F_{lkg} &= \left(\frac{\rho_{lkg}}{R_{dct}}\right)\sqrt{P_{dct}} && (10)\\
F_{dmpr} &= \left(\frac{\rho_{dmpr}(X_{dmpr})}{R_{dct}}\right)\sqrt{P_{dct}} && (11)\\
Q_{rm} &= Q_{sply} + Q_{eqp} + Q_{slr}(t) - Q_{wall} - Q_{rtrn} && (12)\\
Q_{rm} &= C_{rm} \frac{dT_{rm}}{dt} && (13)\\
Q_{wall} &= \sigma_{wall}(T_{rm} - T_{ext}) && (14)
\end{aligned}$$

Nine of the state variables are sensed $\mathbf{s}$, and the constants $\mathbf{c}$ consist of seven unknown parameters $\mathbf{p}$, and four known constants $\mathbf{c}'$:

$$\begin{aligned}
\mathbf{s} &= \langle T_{ext}, T_{sply}, T_{rm}, \frac{dT_{sply}}{dt}, \frac{dT_{rm}}{dt}, X_{rht}, X_{dmpr}, F_{sply}, P_{dct}\rangle^T\\
\mathbf{p} &= \langle R_{dct}, C_{rht}, Q_{rhtmax}, C_{rm}, \sigma_{wall}, Q_{eqp}, Q_{slr}(t)\rangle^T\\
\mathbf{c}' &= \langle \rho_{lkg}, \rho_{dmpr}(X_{dmpr}), C_0, X_{rhtmax}\rangle^T
\end{aligned}$$

Estimation involves adjusting the set of model parameters to maximize the agreement between a specified model and the sensor data using, for example, a least-squares criteria with a Gaussian noise model:

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

where $y \in \mathbf{s}$, $\mathbf{x} \subset \mathbf{s}$, $\mathbf{p}' \subset \mathbf{p}$, $\mathbf{c}' \subset \mathbf{c}$, and y_i and the $\mathbf{x}_i$ are in the ith sampling of sensor values D for $\mathbf{s}$. Applying least-squares involves selecting one of the sensed variables y from $\mathbf{s}$, and manipulating equations $\mathbf{e}(\mathbf{c};\mathbf{v})$ to construct an estimator $y = f(\mathbf{x};\mathbf{p}';\mathbf{c}')$ that predicts y.

The art of model decomposition

It is typically infeasible to estimate the parameters of a large nonlinear model using a single estimator that covers all parameters. This includes our 7 parameter thermal model. However, there is often a large set of possible estimators to choose from, and the number of parameters contained in each estimator varies widely. The art of modeling for data analysis (and DML) involves decomposing a task 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. The process of large scale modeling, embodied in a community of modelers, involves identifying these primitive estimation actions, and planning the ordering and coordination of information flow between them.

A key insight offered by our study is that the process of decomposing a large model estimation problem into primitive estimation actions is analogous to that used in model-based diagnosis to solve large scale multiple fault diagnosis 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 & 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. Following this analogy Moriarty uses a dissent generation algorithm DG1(Williams & Millar 1996) that parallels the conflict recognition phase of model-based diagnosis. DG1 is sound and complete with respect to generating a restricted form of dissent, which is simultaneity free. The analogy also suggests a much more rich space of connections between continuous model-based learning and discrete mode identification (a generalization of model-based diagnosis), as described in the next section.

The thermal estimation problem has eight minimal dissents, a fraction of the overdetermined subsystems, which number in the tens of thousands. Moriarty generates these in 10 seconds on a Sparc 2. The number of parameters mentioned vary from 1 to 7 per dissent. The dissent with the fewest equations and sensed variables is:

$$\langle\langle E9 - 11\rangle^T \langle F_{ext}, P_{dct}, X_{dmpr}\rangle^T\rangle$$

where v^T denotes vector transpose, and a dissent is a pair consisting of vectors of equations and sensed variables. This dissent involves one unknown parameter, R_{dct}, hence a one dimensional search. The error in the parameter estimation is influenced by noise in only three sensors. In contrast, the largest dissent is:

$$\langle\langle E2 - 14\rangle^T, \langle \frac{dT_{rm}}{dt}, \frac{dT_{sply}}{dt}, F_{sply}, P_{dct}, T_{rm}, T_{sply}, X_{rht}, X_{dmpr}\rangle^T\rangle$$

This dissent involves a 7-dimensional search, and accuracy is influenced by noise from eight sensors.

Moriarty generates an estimator from each dissent by simply solving for one of the dissent's sensed variables. It then plans a "simplest" sequence of estimations using a greedy algorithm. This algorithm selects successive estimators in the sequence such that at least one new parameter is being estimated that wasn't estimated at an earlier step in the sequence, while minimizing the number of new parameters being estimated at each step. When two estimators have equal numbers of new parameters, the estimator with the fewest sensed variables is selected. The first condition drives convergence time downwards by reducing the dimensionality of the search space, while the second condition improves accuracy, by reducing the combined sensor error introduced.

Applied to the example this algorithm generates sequence $\langle f_2, f_1, f_6 \rangle$, corresponding to dissents D2, D1 and D6. These contain one, two and four parameters, respectively. f_2, is:

$$F_{ext} = \left(\rho_{lkg} + \rho_{dmpr}(X_{dmpr})\right) \frac{\sqrt{P_{dct}}}{R_{dct}}$$

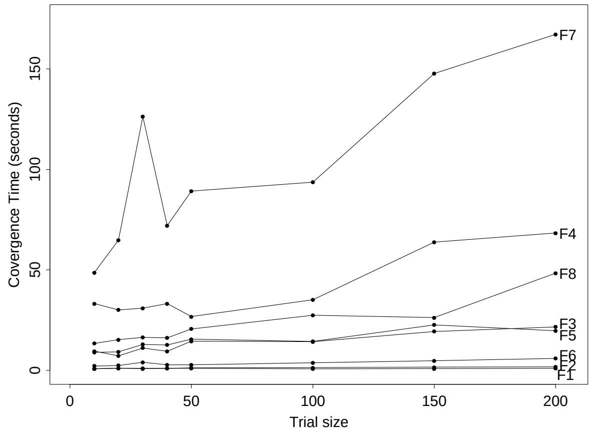
where $y_2 = F_{ext}$, $\mathbf{x}_2 = \langle P_{dct}, X_{dmpr} \rangle^T$, $\mathbf{c}'_2 = \langle \rho_{lkg}, \rho_{dmpr}(X_{dmpr}) \rangle^T$ and $\mathbf{p}_2 = \langle R_{dct} \rangle^T$. f_1, is:

$$\frac{dT_{sply}}{dt} = \frac{C_0 F_{sply}\left(T_{ext} - T_{sply}\right) + \left(\frac{Q_{rhtmax}}{X_{rhtmax}}\right) X_{rht}}{C_{rht}}$$

where $y_1 = \frac{dT_{sply}}{dt}$, $\mathbf{x}_1 = \langle T_{sply}, F_{sply}, T_{ext} X_{rht} \rangle^T$, $\mathbf{c}'_1 = \langle X_{rhtmax}, C_0 \rangle^T$ and $\mathbf{p}_1 = \langle C_{rht}, Q_{rhtmax} \rangle^T$. Finally, f_6 is:

$$\frac{dT_{rm}}{dt} = \frac{C_0 F_{sply}\left(T_{sply} - T_{rm}\right) + Q_{eqp} + Q_{slr}(t) + \sigma_{wall}(T_{ext} - T_{rm})}{C_{rm}}$$

where $y_6 = \frac{dT_{rm}}{dt}$, $\mathbf{x}_6 = \langle T_{rm}, F_{sply}, T_{sply}, T_{ext} \rangle^T$, $\mathbf{c}'_6 = \langle C_0 \rangle^T$ and $\mathbf{p}_6 = \langle C_{rm}, Q_{eqp}, \sigma_{wall}, Q_{slr}(t) \rangle^T$.



To characterize performance the estimators for the 8 dissents were run against data sets ranging in size from 10 to 200, shown above. The plot labeled F7 is for the original 7-dimensional estimator, while plots F2, F1, and F6 are for the estimators selected by Moriarty in its sequence. Estimators were provided with ball-park initial parameter estimates to allow convergence in the higher dimensional cases, and decomposition lead to significant speed-up even when good initial estimates were available. For example, at trial size 200, the original estimator requires 166 seconds, while the total time to estimate all parameters using F2, F1, and F6 is under 9 seconds.

DML automates just one aspect of a rich model decomposition and analysis planning process. For example, in an additional stage not presented here (see (Zhang, Williams, & Elrod 1993)), the responsive environment modelers used dominance arguments (in the spirit of (Williams & Raiman 1994)) to decompose these three estimators into a set of six estimators, one with two unknown parameters, and the remainder with only one unknown. The next real challenge is scaling up to the complete building models developed by DOE, or, for example, NASA's proposed Antartic habitat, which incorporates the control of high density gardens and fish ponds in a closed biosphere.

The New Millennium

NASA has put forth the challenge of establishing a "virtual presence" in space through a fleet of intelligent space probes that autonomously explore the nooks and crannies of the solar system:

> With autonomy we declare that from now on, no sphere is to be off limits. We will send our spacecraft where we cannot see, let them search beyond the horizon, accept that we cannot control them while they are there, and rely on them to tell the tale. – *Guy Man, Autonomy co-Lead of the New Millennium Program.*

This "presence" is to be established at an Apollo-era pace, with software for the first probe to be completed late 1996 and the probe (Deep Space One) to be launched in early 1998. The final pressure, low cost, is of an equal magnitude. Together this poses an extraordinary opportunity and challenge for AI. To achieve robustness during years in the harsh environs of space the spacecraft will need to radically reconfigure itself in response to failures, and then navigate around these failures during its remaining days. To achieve low cost and fast deployment, one-of-a-kind space probes will need to be plugged together quickly, using component-based models wherever possible to automatically generate the control software that coordinates interactions between subsystems. Finally, the space of failure scenarios a spacecraft will need to entertain over its lifespan will be far too large to generate software before flight that explicitly enumerates all contingencies. Hence the spacecraft will need to think through the consequences of reconfiguration options on the fly while ensuring reactivity.

Livingstone is a fast, reactive, model-based autonomous system that is self-modeling and self-configuring. Using a hierarchical control metaphor, Livingstone sits at the nexus between the high-level feedforward reasoning of classical planning/scheduling systems, and the low-level feedback response of continuous adaptive control methods (figure 1), providing a kernel for model-based autonomy. Livingstone is

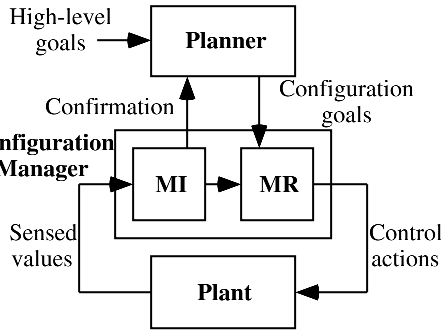


Figure 1: Livingstone: Model-based self-configuration

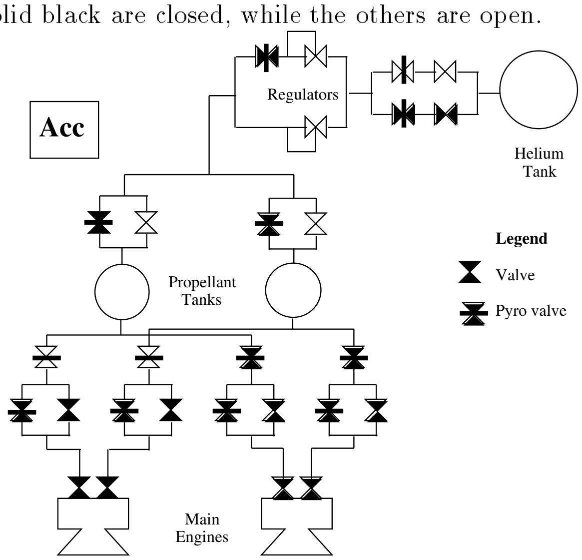
distinguished from more traditional robotic executives through the use of deliberative reasoning in the reactive feedback loop. This deliberative reasoning is compositional, model-based, can entertain an enormous search space of feasible solutions, yet is extremely efficient, due to the ability to quickly focus on the few solutions that are near optimal.

More specifically, three technical features of Livingstone are particularly worth highlighting: First 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 & Pnueli 1992) with the discrete qualitative representations developed in model-based reasoning. Reactivity is respected by restricting the model to concurrent propositional transition systems that are synchronous. Second, this approach helps to unify the classical dichotomy within AI between deduction and reactivity. We achieve a reactive system that performs significant deductions in the sense/response loop by drawing on our past experience at building fast propositional conflict-based algorithms for model-based diagnosis, and by framing a model-based configuration manager as a propositional, conflict-based feedback controller that generates focussed, optimal responses. Third, the long held vision of the model-based reasoning community has been to use a single central model to support a diversity of engineering tasks. For model-based autonomous systems this means using a single model to support tasks including: monitoring, tracking planner goal activations, confirming hardware modes, reconfiguring hardware, detecting anomalies, isolating faults, diagnosis, fault recovery, safing and fault avoidance. Livingstone automates all these tasks using a single model and a single core deductive engine, thus making significant progress towards achieving the model-based vision.

A quick trip to Saturn

In the second half of 1995, Livingstone was part of a 6 month rapid prototyping demonstration of a fully autonomous architecture for spacecraft control, called the Remote Agent. Coincidentally, HAL9000 must have been completed in 1995 for it to complete the 6 year trip to Jupiter, hence a possible explanation for the urgency of our task. To challenge and evaluate the architecture, spacecraft engineers at JPL defined the Newmaap spacecraft and scenario based on Cassini, NASA's most complex spacecraft to date. 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 schematic below depicts Newmaap's idealization of the Cassini main engine subsystem. The main engine subsystem consists of a helium tank, fuel tank, oxidizer tank, 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 tank pressure to a lower working pressure. When propellant paths to a main engine are open, the pressurization of the propellant tanks forces fuel and oxidizer into the main engine, where they combine and spontaneously ignite, producing thrust. The pyro valves can be fired exactly once, changing state from open to closed or vice versa. Their function is to isolate parts of the main engine subsystem until needed, or to isolate failed parts. The latch valve states are controlled using valve drivers (not shown), and the accelerometer (Acc) senses the thrust generated by the main engines. In the figure, valves in solid black are closed, while the others are open.



The following table provides summary information about Livingstone's model of the Newmaap spacecraft, demonstrating its complexity.

Number of components	80
Average modes/component	3.5
Number of propositions	3424
Number of clauses	11101

Starting from 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 by firing a pair of pyros and opening a set of latch valves leading to the engine on the right, to name a few. Different configurations have different associated costs since pyro firings are irreversible actions and require significantly more power than changing the state of 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 now that this engine fails, e.g., by overheating, so that its thrust is low. To ensure that the desired thrust is provided, the spacecraft must be transitioned to a new configuration in which thrust is provided by the main engine on the right. Ideally, this is achieved by firing the two pyro valves leading to the right side, and opening the remaining latch valves (rather than firing additional pyro valves).

Livingstone's reactive, self-configuration capability constantly attempts to move the spacecraft into lowest cost configurations that achieve the desired high-level goals. When the spacecraft strays from the chosen configuration due to failures, Livingstone 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 goals. In this sense Livingstone is a discrete, model-based control system that is self-modeling, self-configuring, goal-directed and reactive.

Model-based configuration

Given a physical plant, modeled as a transition system, and an initial state, *configuration management* involves evolving the transition system along a desired trajectory. The combination of a transition system and a configuration manager is called a *configuration system*.

A plant transition system is composed of a set of concurrent component transition systems that communicate through shared variables. In the Newmaap example, each hardware component in the schematic has a corresponding transition system, and each wire between components denotes communication through variables shared by their corresponding transition systems. The component transition systems of a plant operate synchronously, that is, at each plant transition every component performs a state transition. A component's nominal transitions correspond to correct functioning, while failure transitions move to a set of failure states. In response to a successful repair action, the nominal transition will move the component from a failure state to a nominal state.

We presume that the state of the plant is partially observable, and that the results of control actions on the plant influence the observables and internal state variables through a set of physical processes. Our focus is on reactive configuration management systems that use a model to infer a plant's current state and to select optimal control actions to meet configuration goals. The use of a model is essential to ensure that the goal is likely achieved in the next state without mistake.

More specifically, a *model-based* configuration manager (figure 1) uses a plant transition model $\mathcal{M}$ to determine the desired control sequence in two stages—*mode identification* (MI) and *mode reconfiguration* (MR). MI incrementally generates the set of all plant trajectories consistent with the plant transition model and the sequence of plant control and sensed values. MR uses a plant transition model and the partial trajectories generated by MI up to the current state to determine a set of control values such that all predicted trajectories that don't traverse a failure transition achieve the configuration goal in the next state.

Both MI and MR are reactive. MI infers the current state from knowledge of the previous state and observations within the current state. MR only considers actions that achieve the configuration goal within the next state. Given these commitments, the decision to model component transitions as synchronous is key. An alternative is to model multiple concurrent transitions through interleaving. This, however, would place an arbitrary distance between the current state and the state in which the goal is achieved, defeating a desire to limit inference to a small fixed number of states. Hence we use an abstraction in which multiple commands are given synchronously. In the Newmaap spacecraft demonstration, for example, these synchronous commands were sent by a single processor through interleaving, and the next state sensor information returned to Livingstone is the state following the execution of all these commands.

In practice, not all trajectories and control actions need to be generated. Rather, just the likely trajectories and an optimal control action is required. We efficiently generate these by recasting MI and MR as *combinatorial optimization problems*. In keeping with the reactive nature of configuration management, MI incrementally tracks the likely trajectories by always extending the set of trajectories leading to the current state by the likely transitions to the next state. MR then identifies the lowest cost control action that transitions from any of the likely current states to a next state that achieves the goal configuration. We develop a formal characterization of MI and MR in (Williams & Nayak 1996).

From Newmaap to Deep Space One

The Newmaap scenario included seven failure scenarios. From Livingstone's viewpoint, each scenario required identifying the failure transitions using MI and deciding on a set of control actions to recover from the failure using MR. The following table shows the results of running Livingstone on these scenarios.

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

The first column names each of the scenarios; a discussion of the details of these scenarios is beyond the scope of this paper. The second and fifth columns show the number of solutions checked by MI and MR, respectively. On can see that even though the spacecraft model is large, the use of conflicts 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. This is generally true on spacecraft, since adding sensors is undesirable because it increases spacecraft weight. The fourth and sixth columns show the time in seconds on a Sparc 5 spent by MI and MR on each scenario, once again demonstrating the efficiency of our approach.

Livingstone, tightly integrated with the HSTS planning/scheduling system (Muscettola 1994) and the RAPS executive (Firby 1995), was demonstrated to successfully navigate the simulated NewMaap spacecraft into Saturn orbit during its one hour insertion window, despite half a dozen or more failures, including unanticipated bugs in the simulator. The success of the Newmaap demonstration has launched Livingstone to new heights: Livingstone will be part of the flight software of the first New Millennium mission, called Deep Space One, to be launched in early 1998, with final delivery of Livingstone in late 1996.

The new millenium

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 fundamental advances in model-based autonomous system architectures that are self-modeling, self-configuring, model-based programmable, and support deliberated reactions. This can only be accomplished through a coupling of the diverse set of symbolic and adaptive methods available to AI. While a mere glimmer of Spock and HAL, our two model-based immobots, Livingstone and Moriarty, provide seeds for a new millennium that is most exciting.

Acknowledgements I would particularly like to thank Pandu Nayak, my co-lead in the NASA model-based autonomy project, and Oren Etzioni, whose provocative 1992 talk on softbots provided a catalyst. Thanks also to fellow immobile roboticists Joseph O'Sullivan, Ying Zhang and Bill Millar for their enthusiasm and excellent insights.

References

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130. Reprinted in (Hamscher, Console, & de Kleer 1992).

Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zabowski, D. 1992. A personal learning apprentice. In *Proceedings of AAAI-92*.

Elrod, S., and et al. 1993. Responsive office environments. *Communications of the ACM* 36(7):84–85.

Etzioni, O., and Segal, R. 1992. Softbots as testbeds for machine learning. In *AAAI Spring Symposium on Knowledge Assimilation*.

Firby, R. J. 1995. The RAP language manual. Animate Agent Project Working Note AAP-6, University of Chicago.

Hamscher, W.; Console, L.; and de Kleer, J. 1992. *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann.

Kautz, H.; Selman, B.; Coen, M.; Ketchpel, S.; and Ramming, C. 1994. An experiment in the design of software agents. In *Proceedings of AAAI-94*.

Maes, P., and Kozierok, R. 1993. Learning interface agents. In *Proceedings of AAAI-93*.

Manna, Z., and Pnueli, A. 1992. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag.

Muscettola, N. 1994. HSTS: Integrating planning and scheduling. In Fox, M., and Zweben, M., eds., *Intelligent Scheduling*. Morgan Kaufmann.

Pell, B.; Bernard, D.; Gat, E.; Chien, S.; Muscettola, N.; Nayak, P.; Wagner, M.; and Williams, B. 1996. An implemented architecture integrating onboard planning, scheduling, execution, diagnosis, monitoring and control for autonomous spacecraft. Technical report, NASA Ames. submitted to AAAI96.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–96. Reprinted in (Hamscher, Console, & de Kleer 1992).

Williams, B. C., and Millar, B. 1996. Automated decomposition of model-based learning problems. Technical report, NASA Ames. submitted to AAAI96.

Williams, B. C., and Nayak, P. 1996. Self-configuring, reactive systems: A kernel for model-based autonomy. Technical report, NASA Ames. submitted to AAAI96.

Williams, B. C., and Raiman, O. 1994. Decompositional modeling through caricatural reasoning. In *Proceedings of AAAI-94*, 1199–1204.

Zhang, Y.; Williams, B. C.; and Elrod, S. 1993. Model estimation and energy-efficient control for building management systems. Technical report, Xerox PARC.