ber software-naming conventions and insight into the architecture of the system), programmers working in pairs, adherence to a set of coding standards, collaboration of customers and programmers, frequent verbal communication, frequent releases of software in small increments of development, repeated testing of the developmental software by both programmers and customers, and continuous interaction between the team and the customers.

The environment in which the Maestro team works requires the team to quickly adapt to changing needs of its customers. In addition, the team cannot afford to accept unnecessary development risk. Extreme programming enables the Maestro team to remain agile and provide high-quality software and service to its customers. However, several factors in the Maestro environment have made it necessary to modify some of the conventional extreme-programming practices. The single most influential of these factors is that continuous interaction between customers and programmers is not feasible. The major resulting differences between the Maestro and conventional versions of extreme programming are the following:

- Because customers are not always available for planning sessions, members of the team act on behalf of customers during these sessions.
- In an elaboration of the frequent-planning and incremental-release concept, releases and planning meetings are synchronized with a fixed one-week iteration cycle that facilitates maintenance of focus on the development task.
- Metaphors are occasionally used as needed in specific instances, but the conventional extreme-programming concept of a system metaphor is abandoned as not being helpful.
- In a departure from the simplest-design rule, the team sometimes develops software infrastructure that affords capabilities, beyond those required in the current iteration, that may be useful later in the development process.
- In the absence of continuous involvement of customers and of frequent testing of software by customers, there is heavy reliance on automated testing.

This work was done by Jeffrey Norris, Jason Fox, Kenneth Rabe, I-Hsiang Shu, and Mark Powell of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

The software used in this innovation is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-41811.

Adaptive Behavior for Mobile Robots

A robotic system attempts to both preserve itself and progress toward a goal.

NASA's Jet Propulsion Laboratory, Pasadena, California

The term "System for Mobility and Access to Rough Terrain" (SMART) denotes a theoretical framework, a control architecture, and an algorithm that implements the framework and architecture, for enabling a land-mobile robot to adapt to changing conditions. SMART is intended to enable the robot to recognize adverse terrain conditions beyond its optimal operational envelope, and, in response, to intelligently reconfigure itself (e.g., adjust suspension heights or baseline distances between suspension points) or adapt its driving techniques (e.g., engage in a crabbing motion as a switchback technique for ascending steep terrain). Conceived for original application aboard Mars rovers and similar autonomous or semi-autonomous mobile robots used in exploration of remote planets, SMART could also be applied to autonomous terrestrial vehicles to be used for search, rescue, and/or exploration on rough terrain.

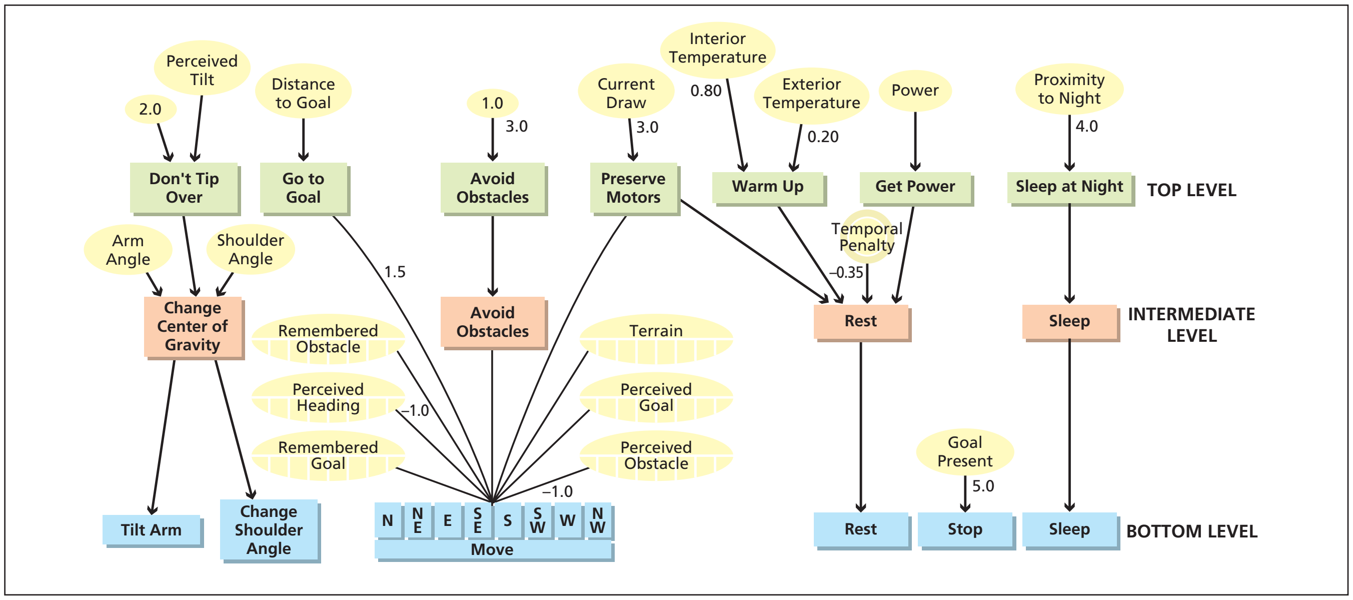
In SMART, controlling the motion of the robot, managing the "health" of the robot, and managing resources are considered as parts of a free-flow behavior hierarchy that autonomously adapts to changing conditions. Tasks that must be performed in the continuing development of SMART are to provide for safe, adaptive mobility on highly sloped terrain include:

- Determination of strategies for adaptive reconfiguration and driving that are nearly optimal with respect to safety and are computationally feasible for on-board implementation,
- Determination of a representation for uncertainty in sensing and prediction of the state of the robot and its environment, and
- Determination of resource-management strategies that mitigate such risks as those of the loss of battery power and/or drive motors.

SMART is based largely on a prior architecture denoted Biologically Inspired System for Map-based Autonomous Rover Control (BISMARC), which, in turn is based on a modified free-flow hierarchy. BISMARC has been used with success in a number of different simulated mission scenarios, wherein it has been demonstrated to afford capabilities for retrieving objects cached at multiple locations, fault tolerance on missions of long duration, and preparing terrain sites for habitation by humans. BISMARC includes provisions for all aspects of safety, self-maintenance, and achievement of goals, as needed to support a sustained presence on the surface of a remote planet.

BISMARC is organized as a two-level system. From stereoscopic images acquired by cameras aboard the robot, the first level generates hypotheses of motor actions. The second level processes these hypotheses, coupled with external and internal inputs, to generate control signals to drive the actuators on the robot.

The figure illustrates the free-flow action-selection hierarchy of BISMARC and SMART. The rectangular boxes represent behaviors, while the ovals represent sensory inputs (either fixed, direct, or derived). At the top are the high-level behaviors, including Don't Tip Over, Go to Goal, Avoid Obstacles, Preserve Motors, Warm Up, Get Power, and Sleep at Night. The intermediate-level behaviors (Change Center of Gravity, Avoid Obstacles, Rest, and Sleep) are designed to interact with both the short-term memory (which corresponds to perceived sensory stimuli), and the long-term memory (which encodes remembered sensory information). Control loops are prevented by use of temporal penalties, which constrain the system to repeat a given behavior no more than a predetermined number of times. The bottom-level behaviors (Tilt Arm, Change Shoulder Angles, Move, Rest, Stop, Sleep) fuse the sensory inputs and the activations of the higher-level behaviors in order to select appropriate actions for safety and achieving goals.



The **Free-Flow Action-Selection Hierarchy** includes multiple behaviors at different levels. The numerical values shown at several places are examples of weights assigned to inputs of behavioral modes. In general, such weights are changed as needed to adapt to changing or previously unknown environmental conditions.

Inputs to the behavioral nodes are calculated as weighted sums. In BISMARC, the weights are fixed; consequently, BISMARC is not capable of adaptation to changing conditions or to environments outside an original world model. In contrast, SMART includes a learning mechanism that adapts the weights to changing and previously unanticipated conditions: An algorithm, known in the art as the maximize collective happiness (MCH) algorithm, adjusts the weights in such a manner as to maintain the health of the robot while ensuring progress toward the goal.

This work was done by Terrance Huntsberger of Caltech for NASA's Jet Propulsion Laboratory.

The software used in this innovation is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-40899.

Protocol for Communication Networking for Formation Flying

This protocol provides for adaptation to changing formation geometry and communication requirements.

NASA's Jet Propulsion Laboratory, Pasadena, California

An application-layer protocol and a network architecture have been proposed for data communications among multiple autonomous spacecraft that are required to fly in a precise formation in order to perform scientific observations. The protocol could also be applied to other autonomous vehicles operating in formation, including robotic aircraft, robotic land vehicles, and robotic underwater vehicles.

A group of spacecraft or other vehicles to which the protocol applies could be characterized as a precision-formation-flying (PFF) network, and each vehicle could be characterized as a node in the PFF network. In order to support precise formation flying, it would be necessary to establish a corresponding communication network, through which the vehicles could exchange position and orientation data and formation-control commands. The communication network must enable communication during early phases of a mission, when little positional knowledge is available. Particularly during early mission phases, the distances among vehicles may be so large that communication could be achieved only by relaying across multiple links. The large distances and need for omnidirectional coverage would limit communication links to operation at low bandwidth during these mission phases. Once the vehicles were in formation and distances were shorter, the communication network would be required to provide high-bandwidth, low-jitter service to support tight formation-control loops.

The proposed protocol and architecture, intended to satisfy the aforementioned and other requirements, are based on a standard layered-reference-model concept. The proposed application protocol would be used in conjunction with conventional network, data-link, and physical-layer protocols. The proposed protocol includes the ubiquitous Institute of Electrical and Electronics Engineers (IEEE) 802.11 medium access control (MAC) protocol to be used in the data-link layer. In addition to its widespread and proven use in diverse local-area networks, this protocol offers both (1) a random-access mode needed for the early PFF deployment phase and (2) a time-bounded-services mode needed during PFF-maintenance operations. Switching between these two modes could be controlled by upper-layer entities using standard link-management mechanisms.

Because the early deployment phase of a PFF mission can be expected to involve multihop relaying to achieve network connectivity (see figure), the proposed protocol includes the open shortest path