NISTIR 6083

Behavior Generation in Intelligent Systems

Dr. James S. Albus
Intelligent Systems Division

Dr. Alexander Meystel
Intelligent Systems Division
and
Drexel University
Department of Electrical & Computer Engineering
Philadelphia, PA 19104

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

October 1997

U.S. DEPARTMENT OF COMMERCE
William M. Daley, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Robert E. Hebner, Acting Director

Summary

This book is about architectures for behavior generation. One should distinguish the concept of "architecture" from the concept of "algorithm". The same architecture can employ different algorithms while the same algorithm can be applied in many architectures. Architecture studies how various functional components should be put together to provide for the desirable functioning of a system. Architecture contemplates the need in partucular components and their interrelatedness if a concrete functioning is required. In other words, architecture can be considered a "metalogic" of the system, or its "meta-algorithm".

This book was stimulated by the problem of developing Intelligent Systems in the variety of application domains such as robotics, integrated manufacturing, large complex systems in costruction. The techniques we describe are to be applied for design and control of all complex integrated technologies where NIST-RCS Architecture is appropriate. It should be considered further development of the general NIST-RCS concept. The thrust of this book is the concept of a recursive BG-module and the concept of multiresolutional, nested Loops of Functioning.

The material contains main statements and positions of the 1992-1994 discussions between J. Albus and A. Meystel. The materials of these discussions have been developed and revised during the period from November 1994 through December 1995. NIST-RCS** has several versions created resulting from gradual evolution of the initial RCS concept within the NASREM* architecture to the current version. Although the NIST-RCS architecture has many successful applications, they are only particular solutions. This book elaborates particularly upon one of the NIST-RCS subsystems: the subsystem of "Behavior Generation" (BG).

The BG subsystem is the driving force for the overall functioning of the NIST-RCS architecture. It propagates Goals top down and requests for the goal corrections from the bottom up. This provides for the transformations from the general Goals, Subgoals, Assignment, and Tasks into plans. These include Job Assignments and Schedules, feedforward and feedback control laws as well as the sequences of control commands. The system of Behavior Generation is

---

* "NASREM" is the NASA/NBS Standard Reference Model for the Space Station Telerobotics [14]. This was one of the first descriptions of the mutiresolutional hierarchy of organization proposed for analysis and design of complex systems.

**The term "RCS" is partially misleading because of the bulk of literature dedicated to a variety of real-time operating computer-based controllers (see, for example, Eds. J. Stankovic, K. Ramamritham, Advances in Real-Time Systems, IEEE CS Press, 1993). In the meantime, in the last meeting on Hybrid Controllers, most of the authors spoke about the results obtained at NIST referring to them as to "NASREM" hierarchical controller. This is why the term NIST-RCS seems to reflect the entire situation in the area.

the hub for the NIST-RCS architecture functioning, and for using NIST-RCS architecture practically. The BG-problem should be resolved in its entirety.

But Behavior Generation is more than Planning/Control activities. It would be impossible without constant use of the World Model — a representation of the World which is valid for the interval of time which can be associated with the word "Now" at each of the hierarchical levels. Actually, Behavior Generation is the result of interacting of the Planning/Control system with the World Model and exploring how and what can be anticipated by the World Model concerned with the decisions the system is about to make.

In their discussions, J. Albus and A. Meystel addressed a refined structure of the "Behavior Generation" subsystem within a conceptual paradigm which can reduce difficulties of task decomposition including planning and control. This material also contains the results of discussions with members of the Architecture Group of Intelligent Systems Division (ISD) and Manufacturing Systems Integration Division (MSID) of NIST, as well as discussions with Ed Barkmeyer and Neil Christopher.

The results from S. Uzzaman's research have been used*. This research was conducted under joint supervision of J. Albus and A. Meystel.

The first version of the report on "Behavior Generation" was issued at NIST, Intelligent Systems Division in November 1994. Then during 1995—three revisions were discussed at ISD. This material includes the fifth revised version of the original report on "Behavior Generation". Parts of this report were reported in papers** and included into NIST documents on ISAM and ISAV.

---

*J. Albus, A. Meystel, S. Uzzaman, "Nested Motion Planning for an Autonomous Robot", Proc. of the IEEE Regional Conference on Aerospace Control Systems, Westlake Village, CA, May 1993

**J. Albus, A. Meystel, "A Reference Model Architecture for Design and Implementation of Semiotic Control in Large and Complex Systems", in Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems, Proc. of the 1995 ISIC Workshop, Monterey, CA 1995, pp. 33-45

J. Albus, A. Meystel, "Intelligent Systems: Beyond Neural Networks and Fuzzy Control", Proc. of the 13th World Congress of IFAC, Plenary Volume, Pergamon, 1996, pp. 107-112

Table of contents

1. Intelligence in Natural and Constructed Systems

1.1 Introduction

Much is unknown about intelligence, and much will remain beyond human comprehension for a very long time. The fundamental nature of intelligence is only dimly understood. The elements of awareness, consciousness, perception, reason, emotion, and intuition are cloaked in mystery that shrouds the human psyche and fades into the religious discourse. Even the definition of intelligence remains a subject of controversy, and so must any theory which attempts to explain what intelligence is, how it originated, or what are the fundamental processes by which it functions.

Yet, much is known, both about the mechanisms and function of intelligence. The study of intelligent machines and neuroscience are both extremely active fields. Many millions of dollars per year are now being spent in Europe, Japan, and the United States on computer integrated manufacturing, robotics, and intelligent machines for a wide variety of military and commercial applications. International researchers in neuroscience search for the anatomical, physiological, and chemical basis of behavior.

Neuroanatomy has produced extensive maps of the interconnecting pathways making up the structure of the brain. Neurophysiology demonstrate how neurons compute functions and communicate information. Neuropharmacology is discovering many of the transmitter substances that modify value judgments, compute reward and punishment, activate behavior, and produce learning. Psychophysics provides many clues about how individuals perceive objects, events, time, and space, and how they reason about relationships between themselves and the external world. Behavioral psychology adds information about mental development, emotions, and behavior.

Research in learning automata, neural nets, and brain modeling provides insight into learning and the similarities and differences between neuronal and electronic computing processes. Computer science and artificial intelligence probe the nature of language and image understanding, and have made significant progress in rule based reasoning, planning, and problem solving. Game theory and operations research have developed methods for decision making in the face of uncertainty. Robotics and autonomous vehicle research has produced advances in real-time sensory processing, world modeling, navigation, trajectory generation, and obstacle avoidance. Research in automated manufacturing and process control has produced intelligent hierarchical controls, distributed databases, representations of object geometry and material properties, data-driven task sequencing, network communications, and multiprocessor operating systems. Modern control theory has developed precise understanding of stability, adaptability, and controllability under various conditions of feedback and noise. Research in sonar, radar, and optical signal processing has developed methods to fuse sensory input from multiple sources, and assess the believability of noisy data.

Progress is rapid, and there exists an enormous and rapidly growing literature in each of the above fields. What is lacking is a general theoretical model of intelligence which ties all these separate areas of knowledge into a unified framework. This Chapter is an attempt to formulate at least the broad outlines of such a model.

Our model is to be used for analysis of all kinds of intelligent systems including such examples as human population, economical system of a country, integrated manufacturing system, etc. We believe that the functioning of most of the large and complex systems can be interpreted in such a way to make them objects in our theory of intelligent systems. For simplicity though, we allude primarily to living creatures or robots in our examples. We expect that active readers will make all necessary implications and evolve the theory into the direction of their interests.

The ultimate goal is a general theory of intelligence that encompasses all possible instantiations: biological, machine, societal, and others. The model presented in this Chapter incorporates knowledge gained from many different sources and the discussion frequently shifts between natural and artificial systems, and both elements elements are discussed. For example, the definition of intelligence addresses both natural and artificial systems. The origin and function of intelligence is treated from the standpoint of biological evolution. The system architecture is strongly influenced by our knowledge of the brain, although as far as its engineering is concerned it is derived almost entirely from research in robotics and control theory. It has been applied to devices ranging from undersea vehicles to automatic factories. The material contains numerous references to neurophysiological, psychological, and psychophysical phenomena that support the model, and frequent analogies are drawn between biological and artificial systems. The value judgments are based mostly on the neurophysiology of the limbic system and the psychology of emotion. Results on neural computation and learning derive mostly from neural net research.

The model is described in terms of definitions, axioms, assertions, theorems, hypotheses, conjectures, and arguments supporting them. Axioms are statements that are assumed to be obvious without proof. Assertions are statements that are considered true by the authors. These statements can or cannot be proven and might be rejected by other researchers. Theorems are statements that the authors feel could be demonstrated true by existing logical methods or empirical evidence. Few of the theorems are proven, but each is followed by informal discussions supporting the theorem and suggesting arguments upon which a formal proof might be constructed. Hypotheses are statements that the authors feel probably could be demonstrated through future research. Conjectures are statements that the authors feel might be demonstrable.

The definition of intelligence includes both biological and machine embodiments. These span an intellectual range from that of an insect to that of an Einstein, from thermostats to the most sophisticated computer systems. In order to be useful in the quest for a general theory, the definition of intelligence must not be limited to behavior that is not understood. The definition of intelligence should include the ability of a robot to spotweld an automobile body, the ability of a

bee to navigate in a field of wild flowers, a squirrel to jump from limb to limb, a duck to land in a high wind, and a swallow to work a field of insects. It should include the ability of blue jays to battle in the branches for a nesting site, a pride of lions to attack a wildebeest, and a flock of geese to migrate. It should include a human's ability to bake a cake, play the violin, read a book, write a poem, fight a war, or invent a computer.

At a minimum, intelligence requires the ability to sense the environment, to make decisions, and to control action. Higher levels of intelligence may include the ability to recognize objects and events, to represent knowledge in a world model, and to reason about and plan for the future. In advanced forms, intelligence provides the capacity to perceive and understand, to choose wisely, and to act successfully under a large variety of circumstances so as to survive, prosper, and reproduce in a complex and often hostile environment.

From the viewpoint of control theory, intelligence might be defined as a knowledgeable "helmsman of behavior." Intelligence is a new phenomenon which emerges as a result of the integration of knowledge and feedback into a sensory-interactive, goal-directed control system that can make plans and generate effective, purposeful action directed toward achieving them.

From the viewpoint of both psychology and biology, intelligence might be defined as a behavioral strategy that gives each individual a means for maximizing the likelihood of propagating its own genes. Intelligence is the integration of perception, reason, emotion, and behavior in a sensing, perceiving, knowing, caring, planning, and acting system that can succeed in achieving its goals in the world.

For the purposes of this book, intelligence will be defined as the ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioral subgoals that support the system's ultimate goal.

Both the criteria of success and the system's ultimate goal are defined external to the intelligent system. For an intelligent machine system, the goals and success criteria are typically defined by designers, programmers, and operators. For intelligent biological creatures, the ultimate goal is gene propagation, and success criteria are defined by the processes of natural selection.

Assertion: There exist degrees, or levels, of intelligence, which are determined by the following features of the system: 1) the computational power of the system's brain (or computer), 2) the sophistication of algorithms the system uses for sensory processing, world modeling, behavior generation, value judgment, and global communication, 3) the information and values the system has stored in its memory, and 4) the sophistication of the processes of the system functioning. These levels of intelligence are different in the probability of the success of decisions that is measured by various criteria of performance (including time, accuracy, and others.)

Intelligence can be observed to grow and evolve, both through growth in computational

power, and through accumulation of knowledge of how to sense, decide, and act in a complex and changing world. In artificial systems, growth in computational power and accumulation of knowledge derives mostly from human hardware engineers and software programmers. In natural systems, intelligence grows, over the lifetime of an individual, through maturation and learning, and over intervals spanning generations, through evolution.

Thus, our idea of intelligence goes beyond the concept of simple *adaptation*. It includes adaptation at different time scales and becomes closer rather to the concept of *learning*. Note that learning is not required in order to be intelligent, only to become more intelligent as a result of experience. Learning is defined as consolidating short-term memory into long-term memory, and exhibiting altered behavior because of that memory. We discuss learning as a mechanism for storing knowledge about the external world, and for acquiring skills and knowledge of how to act. It is, however, assumed that many creatures can exhibit intelligent behavior using instinct, without having learned anything.

1.2 The Origin and Function of Intelligence

Generation of advantageous behavior

Assertion: Natural intelligence, like the brain in which it appears, is a result of the natural selection processes.

The brain is first and foremost a control system. Its primary function is to produce successful goal-seeking behavior in finding food, avoiding danger, competing for territory, attracting sexual partners, and caring for offspring. All brains, even those of the smallest insects, generate and control behavior. Some brains produce only simple forms of behavior, while others produce very complex behaviors. Only the most recent (on the scale of evolution) and highly developed brains show any evidence of abstract thought.

Assertion: For each intelligent system (natural, or constructed), intelligence provides a mechanism to generate advantageous behavior.

Intelligence improves an individual's ability to act effectively and choose wisely between alternative behaviors. A more intelligent animal has many advantages over less intelligent rivals in acquiring choice territory, gaining access to food, and attracting more desirable mates. The intelligent use of aggression improves an individual's position in the social world. Intelligent predation improves success in capturing prey. Intelligent exploration improves success in hunting and establishing territory. Intelligent use of stealth gives a predator the advantage of surprise. Intelligent use of deception improves the prey's chances of escaping from danger.

Higher levels of intelligence allow the system to think ahead, plan before acting, and reason about the probable results of alternative actions. These abilities give the more intelligent individual a competitive advantage over the less intelligent in the competition for survival and gene

propagation. Intellectual capacities and behavioral skills that produce successful hunting and gathering of food, acquisition and defense of territory, avoidance and escape from danger, and bearing and raising offspring tend to be passed on to succeeding generations. Intellectual capabilities that produce less successful behaviors reduce the survival probability of the brains that generate them. Competition between individuals drives the evolution of intelligence within a species.

Assertion: For groups of individuals, intelligence provides a group mechanism for cooperatively generating advantageous behavior.

The intellectual capacity to simply congregate into flocks, herds, schools, and packs increases the number of sensors watching for danger. The ability to communicate danger signals improves the survival probability of all individuals in the group. Communication is most advantageous to the quickest individuals, who recognize danger messages, and effectively respond. The intelligence to cooperate in mutually beneficial activities, such as hunting and group defense, increases the probability of gene propagation for all members of the group.

The most intelligent individuals and groups within a species will tend to occupy the best territory, be the most successful in social competition, and have the best chances of their offsprings' survival. More intelligent individuals and groups will dominate in serious competition.

Biological intelligence is the product of continuous competitive struggle for survival and gene propagation which has taken place between billions of brains, over millions of years. The results of those struggles have been determined in large measure by the intelligence of the competitors.

Intelligence of constructed systems should be designed so that similar properties could be achieved by *intelligent systems*.

Communication and Language

The ability to transform "reality" into "representation of reality" can be considered to be one of the most important phenomena linked with intelligence. This representation of reality is done in signs and is necessary for storage, communication, and behavior generation. The signs contain data, information, and knowledge which reflect different levels of representation.

Definition: Communication is the transmission of data, information, and knowledge between intelligent systems, or among subsystems of an intelligent system.

Definition: Language is the means by which data, information, and knowledge are encoded for purposes of communication and storage.

Language has three basic components: vocabulary, syntax, and semantics. Vocabulary is the set of words in the language. Words may be represented by symbols. Syntax, or grammar, is

the set of rules for generating strings of symbols that form sentences. Semantics is the encoding of information into meaningful patterns, or messages. Messages are sentences that convey useful information.

Communication requires that information be: 1) encoded, 2) transmitted, 3) received, 4) decoded, 5) interpreted, 6) understood. Understanding implies that the information in the message has been decoded correctly, incorporated into the world model of the receiver, and problems in it are explicated. Understanding of problems presumes subsequent search for their solution and generation of the required actions.

Communication may be either intentional or unintentional. Intentional communication occurs as the result of a sender executing a task whose goal it is to alter the knowledge or behavior of the receiver to the benefit of the system's goal. Unintentional communication occurs when a message is unintentionally sent, or when an intended message is received and understood by someone other than the intended receiver. Preventing an enemy from receiving and understanding communication between friendly agents can often be crucial to survival.

Communication and language are not unique to human beings. Virtually all creatures, even insects, communicate in some way, and hence have some form of language. For example, many insects transmit messages announcing their identity and position. This may be done acoustically, by smell, or by some visually detectable display. The goal may be to attract a mate or to facilitate recognition and/or location by other members of a group. Species of lower intelligence, such as insects, have very little information to communicate, and hence have languages with only a few of what might be called words, with little or no grammar. In many cases, language vocabularies include motions and gestures (i.e. body or sign language) as well as acoustic signals generated by variety of mechanisms from stamping feet, to snorting, squealing, chirping, crying, and shouting.

Theorem: In any species, language evolves to provide for the balance between the adequate content and the complexity of messages that can be generated by the intelligence of that species.

Depending on its complexity, a language may be capable of communicating many sophisticated messages, or only a few simple ones. More intelligent individuals have a larger vocabulary and more complex syntax, and are quicker to understand and act on the meaning of messages.

Hypothesis: To the receiver, the benefit, or value, of communication is roughly proportional to the product of the amount of information contained in the message, multiplied by the ability of the receiver to understand and act on that information, multiplied by the importance of the act to survival and gene propagation of the receiver. This statement can be extended by a

further hypothesis that the increment of the value per unit of information (a sign, or a label) can serve as a measure of increase in knowledge and/or action resulting in a better state for success of functioning.

Based upon similar premises, similar evaluations can be made for many other important parameters of the situation. For example, to the sender, the benefit is the value of the receiver's action to the sender, minus the danger incurred by transmitting a message that may be intercepted by, and give advantage to, an enemy.

Greater intelligence enhances both the individual's and the group's ability to analyze the environment, to encode and transmit information about it, to detect messages, to recognize their significance, and act effectively on information received. Greater intelligence produces more complex languages capable of expressing more information, i.e. more messages with more shades of meaning.

In social species, communication also provides the basis for societal organization. Communication of threats that warn of aggression can help to establish the social dominance hierarchy, and reduce the incidence of physical harm from fights over food, territory, and sexual partners. Communication of alarm signals indicates the presence of danger, and in some cases, identifies its type and location. Communication of pleas for help enables group members to solicit assistance from one another. Communication between members of a hunting pack enables them to remain in formation while spread far apart, and hence to hunt more effectively by cooperating as a team in the the tracking and killing of prey.

Among humans, primitive forms of communication include facial expressions, cries, gestures, body language, and pantomime. The human brain is, however, capable of generating ideas of much greater complexity and subtlety than can be expressed through cries and gestures. To transmit messages commensurate with the complexity of human thought, human languages have evolved with grammatical and semantic rules capable of stringing words from vocabularies consisting of thousands of entries into sentences which express ideas and concepts with exquisitely subtle nuances of meaning. To support this process, the human vocal apparatus has evolved complex mechanisms for making a large variety of sounds.

Human Intelligence and Technology

Superior intelligence alone made humans successful hunters. The intellectual capacity to make and use tools, weapons, and spoken language made humans the most successful of all predators. In recent millennia, humans' levels of intelligence have led to the use of fire, the domestication of animals, the development of agriculture, the rise of civilization, the invention of writing, the building of cities, the practice of war, the emergence of science, and the growth of industry. These capabilities have extremely high gene propagation value for the individuals and

societies that possess them relative to those who do not. Intelligence has thus made modern civilized humans the dominant species on the planet Earth.

Conjecture: Any intelligent system can be decomposed into four elements (subsystems) of intelligence: sensory processing, world modeling, behavior generation, and value judgment. Input to and output from intelligent systems are realized via sensors and actuators that can be external agents. However, they become components of the loop of functioning of the intelligent system (see ELF in Chapter 2).

ACTUATORS -- Output from an intelligent system is produced by actuators which move, exert forces, and position arms, legs, hands, and eyes. Actuators generate forces to point sensors, excite transducers, move manipulators, handle tools, steer and propel locomotion. An intelligent system may have a few or thousands of actuators, all of which must be coordinated to perform tasks and accomplish goals. Natural actuators are muscles and glands. Machine actuators are motors, pistons, valves, solenoids, and transducers. In organizational (and/or social) systems, actuators can be individuals, or groups of people.

The concept of actuation can be extended for any process that produces an action. Speech generation can be considered a process that happens via actuation. A "move" on a stock market can be considered actuation for an appropriate system. In our further examples, we will discuss only relatively straightforward cases related to living creatures.

SENSORS -- Input to an intelligent system is produced by sensors, which may include visual brightness and color sensors; tactile, force, torque, position detectors; velocity, vibration, acoustic, range, smell, taste, pressure, and temperature measuring devices. Sensors may be used to monitor both the state of the external world and the internal state of the intelligent system itself. Sensors provide input to a sensory processing system. Similar to actuators, sensors are not limited to technical or biological devices. In organizational (and/or social) systems, both sensors and actuators can be individuals, or groups of people.

SENSORY PROCESSING -- Perception occurs in a sensory processing system element that compares sensory observations with expectations generated by an internal world model. Sensory processing algorithms integrate similarities and differences between observations and expectations over time and space so as to detect events and recognize features, objects, and relationships in the world. Sensory input data, from a wide variety of sensors, over extended periods of time, are fused into a consistent unified perception of the state of the world. Sensory processing algorithms compute distance, shape, orientation, surface characteristics, physical and dynamical attributes of objects and regions of space. Sensory processing is equivalent to the subsystem of Perception in living creatures. Sensory processing may include elements of speech recognition and language and music interpretation.

WORLD MODEL -- The world model is the intelligent system's best estimate of the state of the world. The world model includes a database of knowledge about the world, plus a database management system that stores and retrieves information. The world model also contains a simulation capability which generates expectations and predictions. The world model provides answers to requests for information about the present, past, and probable future states of the world. The world model provides this information service to the behavior generation system element to make intelligent plans and behavioral choices. It provides information to the sensory processing system element to perform correlation, model matching, and model-based recognition of states, objects, and events. It provides information to the value judgment system element to compute values such as cost, benefit, risk, uncertainty, importance, attractiveness, etc. The world model is kept up-to-date by the sensory processing system element.

VALUE JUDGMENT -- The value judgment system element determines good and bad, rewards and punishments, important and trivial, certain and improbable. The value judgment system evaluates both the observed state of the world and the predicted results of hypothesized plans. It computes costs, risks, and benefits both of observed situations and of planned activities. It computes the probability of correctness and assigns believability and uncertainty parameters to state variables. It also assigns attractiveness, or repulsiveness to objects, events, regions of space, and other creatures. The value judgment system provides the basis for decision making, or choosing one action instead of another. Without value judgments, any biological creature would soon be destroyed, and any artificially intelligent system would soon be disabled by its own inappropriate actions.

BEHAVIOR GENERATION -- Behavior results from a behavior generating system element that selects goals and plans and executes tasks. Tasks are recursively decomposed into subtasks, and subtasks are sequenced to achieve goals. Goals are selected and plans generated by a looping interaction between behavior generation, world modeling, and value judgment elements. The behavior generating system hypothesizes plans. The world model predicts the results of those plans, and the value judgment element evaluates those results. The behavior generating system then selects the plans with the highest evaluations for execution. The behavior generating system element also monitors the execution of plans and modifies existing plans when the situation requires.

Each of the system elements of intelligence are reasonably well understood. The phenomena of intelligence, however, requires more than a set of disconnected elements. Intelligence requires an interconnecting system architecture that enables the various system elements to interact and communicate in intimate and sophisticated ways.

A system architecture partitions the system elements of intelligence into computational modules, and interconnects the modules in networks and hierarchies. It is what enables the behavior generation elements to direct sensors and to focus sensory processing algorithms on

objects and events worthy of attention, ignoring things that are not important to current goals and task priorities. The system architecture enables the world model to answer queries from behavior generating modules and make predictions and receives updates from sensory processing modules. It is what communicates from the value judgment element to the goal selection subsystem. The values of state-variables describe the success of behavior and the desirability of states of the world.

1.3 An Architecture for Intelligent Systems

A number of system architectures for intelligent machine systems have been conceived, and a few implemented. [1-14, 20] The architecture for intelligent systems that will be proposed here is largely based on the Real-time Control System (RCS) that has been implemented in a number of versions over the past 13 years at the National Institute for Standards and Technology (NIST formerly NBS). NIST-RCS was first implemented by Barbera for laboratory robotics in the mid 1970's [7] and adapted by Albus, Barbera, and others for manufacturing control in the NIST Automated Manufacturing Research Facility (AMRF) during the early 1980's [11,12]. Since 1986, RCS has been implemented for a number of additional applications, including the NBS/DARPA Multiple Autonomous Undersea Vehicle (MAUV) project [13], the Army Field Material Handling Robot, and the Army TMAP and TEAM semi-autonomous land vehicle projects. RCS is the basis of the NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM) being used on the space station Flight Telerobotic Servicer [14] and the Air Force Next Generation Controller. Recent RCS applications include the submarine system of control [108], machine controller EMC [109], NGIS [110], ADACS [111], and others.

Other groups also introduced architectures that are equivalent to the NIST-RCS architecture. We mention here only results obtained by the Drexel University researchers who implemented this architecture for the Autonomous Mobile Robot "Dune-Buggy" [15-17], in the machine for automated spray casting "OSPREY" [18], and for the power plant control system [20]. A number of RCS applications is done by ATR (see [112].)

The proposed system architecture organizes the elements of intelligence so as to create the functional relationships and information flows shown in Figure 1-1. In all intelligent systems, a sensory processing system processes sensory information to acquire and maintain an internal model of the external world. In all systems, a behavior generating system controls actuators to pursue behavioral goals in the context of the perceived world model. In systems of higher intelligence, the behavior generating system element may interact with the world model and value judgment system to reason about space and time, geometry and dynamics, and to formulate or select plans based on values such as cost, risk, utility, and goal priorities. The sensory processing system element may interact with the world model and value judgment system to assign

values to perceived entities, events, and situations.

Figure 1-2 shows how the proposed system architecture replicates and distributes the relationships of Figure 1-1 over a hierarchical computing structure. All logical and temporal properties illustrated in Figure 1-1 for a single level are kept. An organizational hierarchy where computational nodes are arranged in layers like command posts in a military organization appears on the left. Each node in the organizational hierarchy contains four types of computing modules: behavior generating (BG), world modeling (WM), sensory processing (SP), and value judgment (VJ) modules. Each chain of command in the organizational hierarchy, from each actuator and each sensor to the highest level of control, can be represented by a computational hierarchy, such as what is illustrated in the center of Figure 1-2

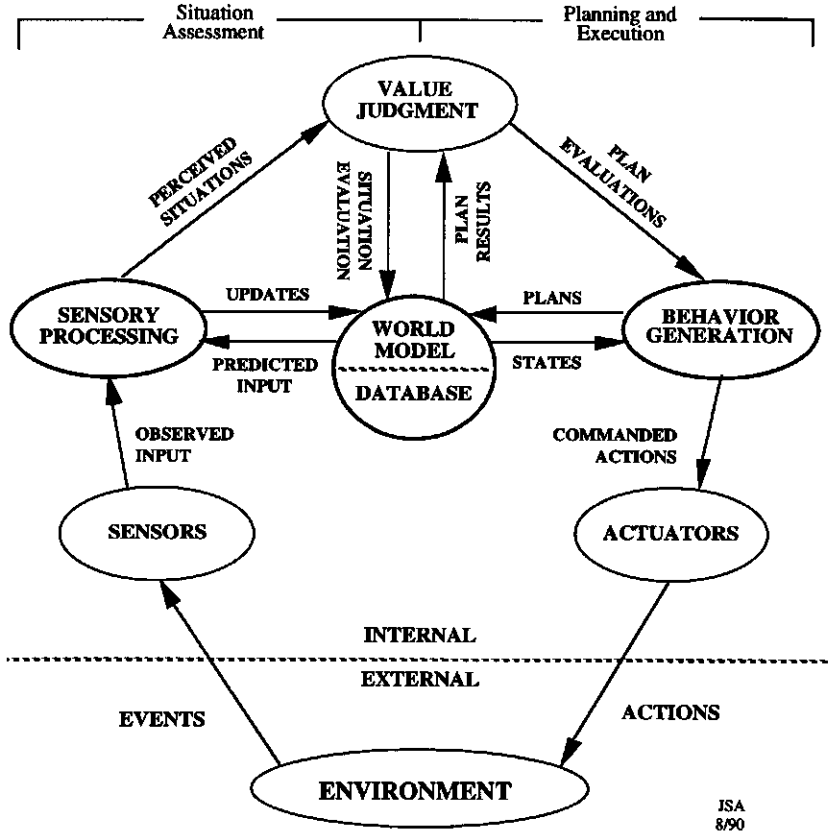


Figure 1. **The elements of intelligence and the functional relationships between them.**

At each level, the nodes and computing modules within the nodes, are interconnected to each other by a communication system. Within each computational node, the communication system provides intermodule communications of the type shown in Figure 1-1. Queries and task status are communicated from BG modules to WM modules. Retrievals of information are

communicated from WM modules back to the BG modules making the queries. Predicted sensory data is communicated from WM modules to SP modules. Updates to the world model are communicated from SP to WM modules. Observed entities, events, and situations are communicated from SP to VJ modules. Values assigned to the world model representations of these entities, events, and situations are communicated from VJ to WM modules. Hypothesized plans are communicated from BG to WM modules. Results are communicated from WM to VJ modules. Evaluations are communicated from VJ modules back to the BG modules that hypothesized the plans.

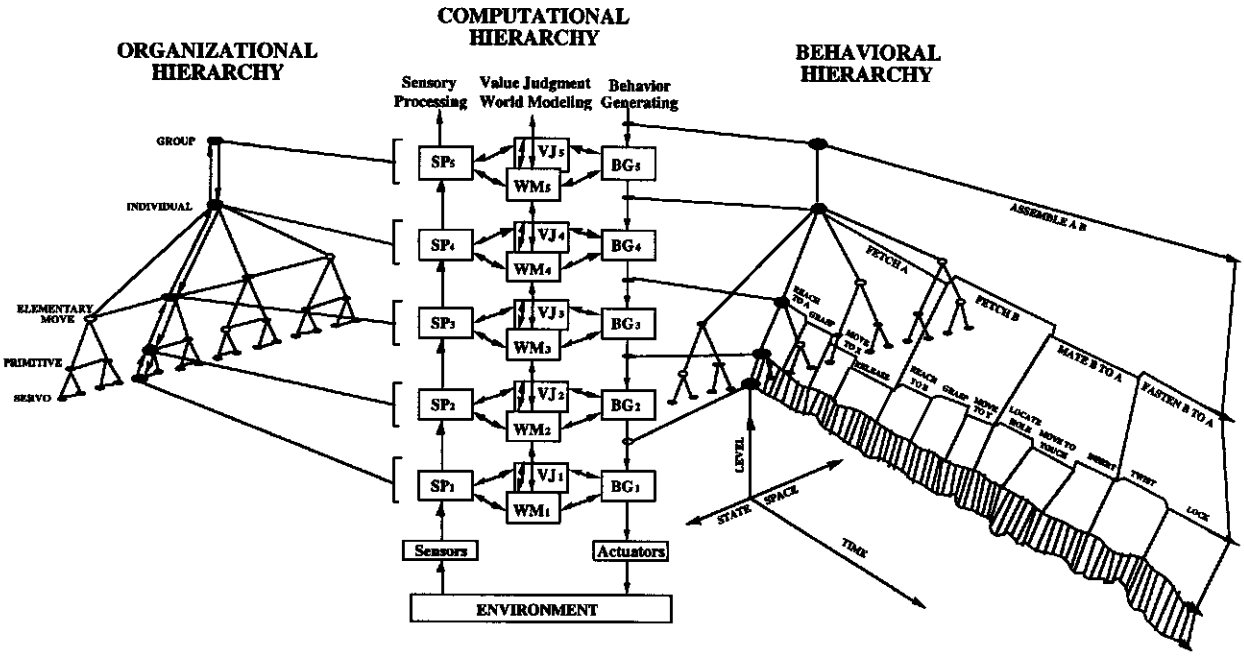


Figure 2. Relationships in hierarchical control systems. On the left, is an organizational hierarchy consisting of a tree of command centers, each of which possesses one supervisor and one or more subordinates. In the center, is a computational hierarchy consisting of BG, WM, SP, and VJ modules. Each actuator and each sensor is serviced by a computational hierarchy. On the right, is a behavioral hierarchy consisting of trajectories through state-time-space. Commands at each level can be represented by vectors, or points in state-space. Sequences of commands can be represented as trajectories through state-time-space.

The communications system also communicates between nodes at different levels. Commands are communicated downward from supervisor BG modules in one level to subordinate BG modules in the level below. Status reports are communicated back upward through the world model from lower level subordinate BG modules to the upper level supervisor BG modules from which commands were received. Observed entities, events, and situations detected by SP modules at one level are communicated upward to SP modules at a higher level. Predicted attributes of entities, events, and situations stored in the WM modules at a higher level are communicated downward to lower level WM modules. Output from the bottom level BG modules is

communicated to actuator drive mechanisms. Input to the bottom level SP modules is communicated from sensors.

The communications system can be implemented in a variety of ways. In a biological brain, communication is mostly via neuronal axon pathways, although some messages are communicated by hormones carried in the bloodstream. In artificial systems, the physical implementation of communications functions may be a computer bus, a local area network, a common memory, a message passing system, or some combination thereof. In either biological or artificial systems, the communications system may include the functionality of a communications processor, a file server, a database management system, a question answering system, or an indirect addressing or list processing engine. In the system architecture proposed here, the input/output relationships of the communications system produce the effect of a virtual global memory, or blackboard system [20].

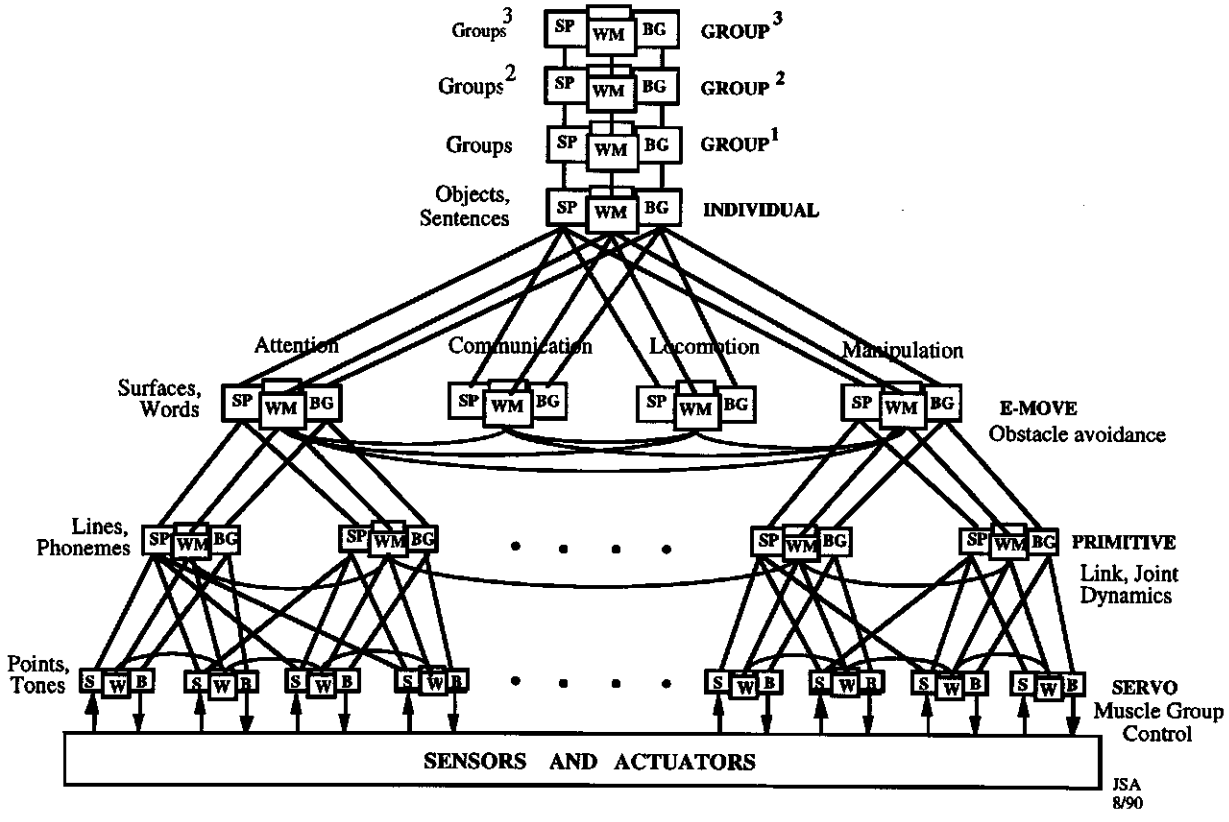


Figure 3. **An organization of processing nodes such that the BG modules form a command tree. On the right are examples of the functional characteristics of the BG modules at each level. On the left are examples of the type of visual and acoustical entities recognized by the SP modules at each level. In the center of level 3 are the type of subsystems represented by processing nodes at level 3.**

The input command string to each of the BG modules at each level generates a trajectory through state-space as a function of time. The set of all command strings create a behavioral hierarchy, as shown on the right of Figure 1-2. Actuator output trajectories (not shown in Figure1-2) correspond to observable output behavior. All the other trajectories in the behavioral hierarchy constitute the deep structure of behavior [21].

Hierarchical vs. Horizontal

Figure 1-3 shows the organizational hierarchy in more detail, and illustrates both the hierarchical and horizontal relationships involved in the proposed architecture. The architecture is hierarchical, commands and status feedback flow hierarchically up and down a behavior generating chain of command. The architecture is also hierarchical in that sensory processing and world modeling functions have hierarchical levels of temporal and spatial aggregation.

The architecture is horizontal in that data is shared horizontally between heterogeneous modules at the same level. At each hierarchical level, the architecture is horizontally interconnected by wide-bandwidth communication pathways between BG, WM, SP, and VJ modules in the same node, and between nodes at the same level, especially within the same command subtree. The horizontal flow of information is voluminous within a single node, but less between related nodes in the same command subtree. It has relatively low bandwidth between computing modules in separate command subtrees. Communications bandwidth is indicated in Figure 1-3 by the thickness of the horizontal connections.

The volume of information flowing horizontally within a subtree may be orders of magnitude larger than the amount flowing vertically in the command chain. The volume of information flowing vertically in the sensory processing system can also be very high, especially in the vision system.

The specific configuration of the command tree is task dependent and therefore not necessarily stationary in time. Figure 1-3 illustrates only one possible configuration that may exist at a single point in time. During operation, relationships between modules within and between layers of the hierarchy may be reconfigured in order to accomplish different goals, priorities, and task requirements. This means that any particular computational node with its BG, WM, SP, and VJ modules, may belong to one subsystem at one time and a different subsystem a very short time later. For example, the mouth may be part of the manipulation subsystem (while eating) and the communication subsystem (while speaking). Similarly, an arm may be part of the manipulation subsystem (while grasping) and part of the locomotion subsystem (while swimming or climbing).

In the biological brain, command tree reconfiguration can be implemented through multiple axon pathways that exist, but are not always activated, between BG modules at different hierarchical levels. These multiple pathways define a layered graph, or lattice, of nodes and

directed arcs, such as shown in Figure 1-4.

They enable each BG module to receive input messages and parameters from several different sources. During operation, goal driven switching mechanisms in the BG modules (discussed in Chapter 4) assess priorities, negotiate for resources, and coordinate task activities so as to select among the possible communication paths of Figure 1-4. As a result, each BG module accepts task commands from only one supervisor at a time, and hence the BG modules form a command tree at every instant in time.

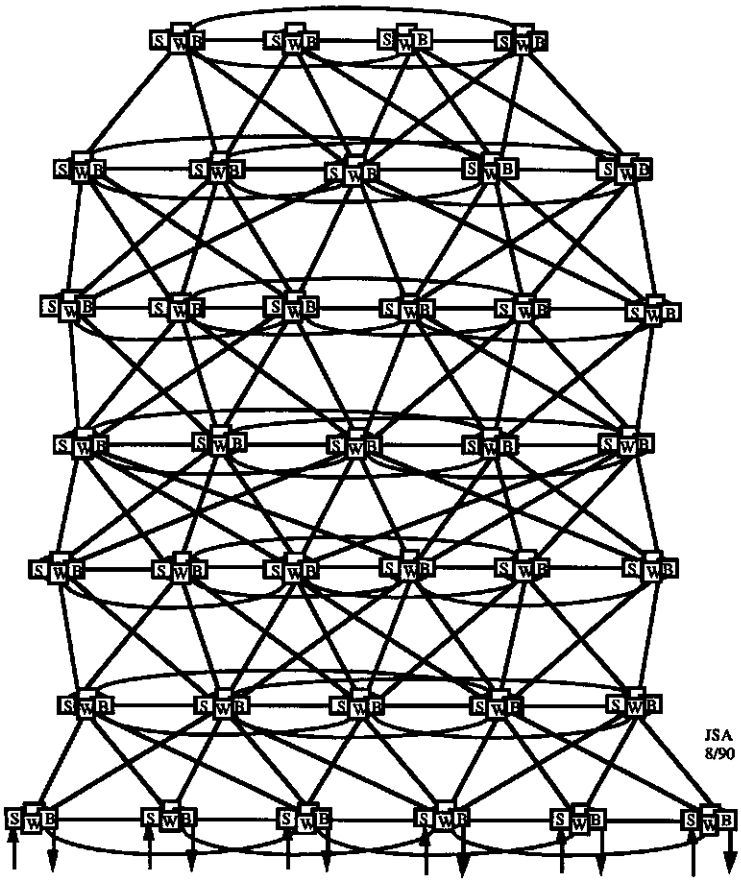


Figure 4. Each layer of the system architecture contains a number of nodes, each of which contains BG, WM, SP, and VJ modules. The nodes are interconnected as a layered graph, or lattice, through the communication system. Note that the nodes are richly, but not fully, interconnected. Outputs from the bottom layer BG modules drive actuators. Inputs to the bottom layer SP modules convey data from sensors. During operation, goal driven communication path selection mechanisms configure this lattice structure into the organizational tree shown in Figure 3.

The SP modules are also organized hierarchically, but as a layered graph, not a tree. At each higher level, sensory information is processed into increasingly higher levels of abstraction,

but the sensory processing pathways may branch and merge in many different ways.

Hierarchical Levels

Levels in the behavior generating hierarchy are defined by temporal and spatial decomposition of goals and tasks into levels of resolution. Temporal resolution is manifested in terms of loop bandwidth, sampling rate, and state-change intervals. Temporal span is measured by the length of historical traces and planning horizons. Spatial resolution is manifested in the branching of the command tree and the resolution of maps. Spatial span is measured by the span of control and the range of maps.

Levels in the sensory processing hierarchy are defined by temporal and spatial integration of sensory data into levels of aggregation. Spatial aggregation is best illustrated by visual images. Temporal aggregation is best illustrated by acoustic parameters such as phase, pitch, phonemes, words, sentences, rhythm, beat, and melody.

Levels in the world model hierarchy are defined by temporal resolution of events, spatial resolution of maps, and by parent-child relationships between entities in symbolic data structures. The spatial and temporal unification is achieved within the subsystem of Behavior Generation (BG) because the needs of SP and BG modules can differ within a level of resolution.

Theorem: In a hierarchically structured goal-driven, sensory-interactive, intelligent control system architecture:

a) control bandwidth decreases about an order of magnitude at each higher level,

b) perceptual resolution of spatial and temporal patterns decreases about an order-of-magnitude at each higher level,

c) goals expand in scope and planning horizons expand in space and time about an order-of-magnitude at each higher level, and

d) models of the world and memories of events decrease in resolution and expand in spatial and temporal range by about an order-of-magnitude at each higher level.

It is well known from control theory that hierarchically nested servo loops tend to suffer instability unless the bandwidth of the control loops differ by about an order of magnitude. This suggests, perhaps even requires, condition a) above. Numerous theoretical and experimental studies support the concept of hierarchical planning and perceptual "chunking" for both temporal and spatial entities [22, 23]. These support conditions b), c), and d) above.

In elaboration of the above assertion, we can construct a timing diagram, as shown in Figure 1-5. The range of the time scale increases, and its resolution decreases, exponentially by about an order of magnitude at each higher level. Hence the planning horizon and event summary interval increases, and the loop bandwidth and frequency of subgoal events decreases, exponentially at each higher level.

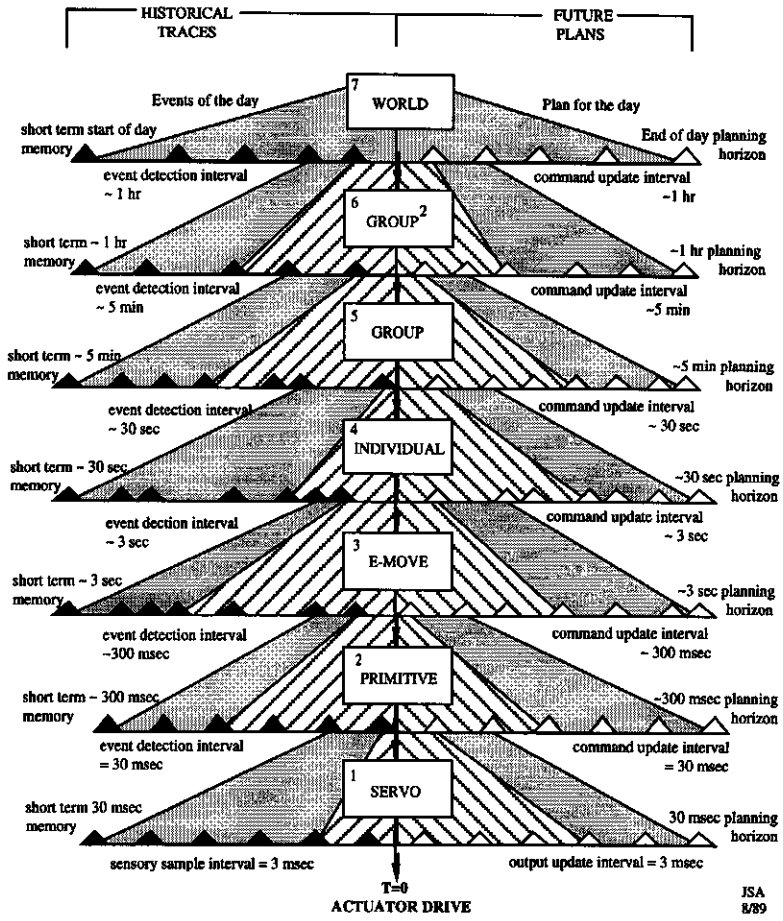


Figure 5. A timing diagram illustrating the temporal flow of activity in the task decomposition and sensory processing systems. At the world level, high level sensory events and circadian rhythms react with habits and daily routines to generate a plan for the day. Each element of that plan is decomposed through the remaining six levels of task decomposition into action.

The seven hierarchical levels in Figure 1-5 span a range of time intervals from three milliseconds to one day. Three milliseconds was arbitrarily chosen as the shortest servo update rate because that is adequate to reproduce the highest bandwidth reflex arc in the human body. One day was arbitrarily chosen as the longest historical-memory/planning-horizon to be considered. Shorter time intervals could be handled by adding another layer at the bottom. Longer time intervals could be treated by adding layers at the top, or by increasing the difference in loop bandwidths and sensory chunking intervals between levels.

The origin of the time axis in Figure 1-5 is the present, i.e. t=0. Future plans lie to the right of t=0, past history to the left. The open triangles in the right half-plane represent task goals in a future plan. The filled triangles in the left half-plane represent recognized task-completion events in a past history. At each level there is a planning horizon and a historical event summary interval. The heavy cross-hatching on the right shows the planning horizon for the current task.

The light shading on the right indicates the planning horizon for the anticipated next task. The heavy cross-hatching on the left shows the event summary interval for the current task. The light shading on the left shows the event summary interval for the immediately previous task. In Figure 1- 5, the scales as chosen so that the planning horizons look equal geometrically.

Figure 1-5 suggests a duality between the behavior generation and the sensory processing hierarchies. At each hierarchical level, planner modules decompose task commands into strings of planned subtasks for execution. At each level, strings of sensed events are summarized, integrated, and "chunked" into single events at the next higher level.

Planning implies an ability to predict future states of the world. Prediction algorithms based on Fourier transforms or Kalman filters typically use recent historical data to compute parameters for extrapolating into the future. Predictions made by such methods are typically not reliable for periods longer than the historical interval over which the parameters were computed. Thus at each level, planning horizons extend into the future only about as far, and with about the same level of detail, as historical traces reach into the past.

Predicting the future state of the world often depends on assumptions concerning which actions are to be taken and the reactions to be expected from the environment, including which actions may be taken by other intelligent agents. Planning of this type requires search over the space of possible future actions and probable reactions. Search-based planning takes place via a looping interaction between the BG, WM, and VJ modules. This is described in more detail in the section 4 discussion on BG modules.

Planning complexity grows exponentially with the number of steps in the plan (i.e. the number of decision steps in the search graph). If real-time planning is to succeed, any given planner must operate in a limited search space. If there is too much steps in the time line, or in the space of possible actions, the size of the search graph can easily become too large for real-time response. One method of resolving this problem is to use a multiplicity of planners in hierarchical layers [14, 23] so that at each layer no planner needs to search more than a given number (for example ten) steps deep in a game graph, and at each level there are no more than (ten) subsystem planners that need to simultaneously generate and coordinate plans. These criteria give rise to hierarchical levels with exponentially expanding spatial and temporal planning horizons, and characteristic degrees of detail for each level. The result of hierarchical spatio-temporal planning is illustrated in Figure 1-6. At each level, plans consist of at least one, and on average ten, subtasks. The planners have a planning horizon that extends about one-and-a-half average input command intervals into the future. In Figure 1-6, the scales at the time-axes are equal. We are dealing with a "funnel hierarchy" which focuses attention on a more and more narrow scope of attention top-down.

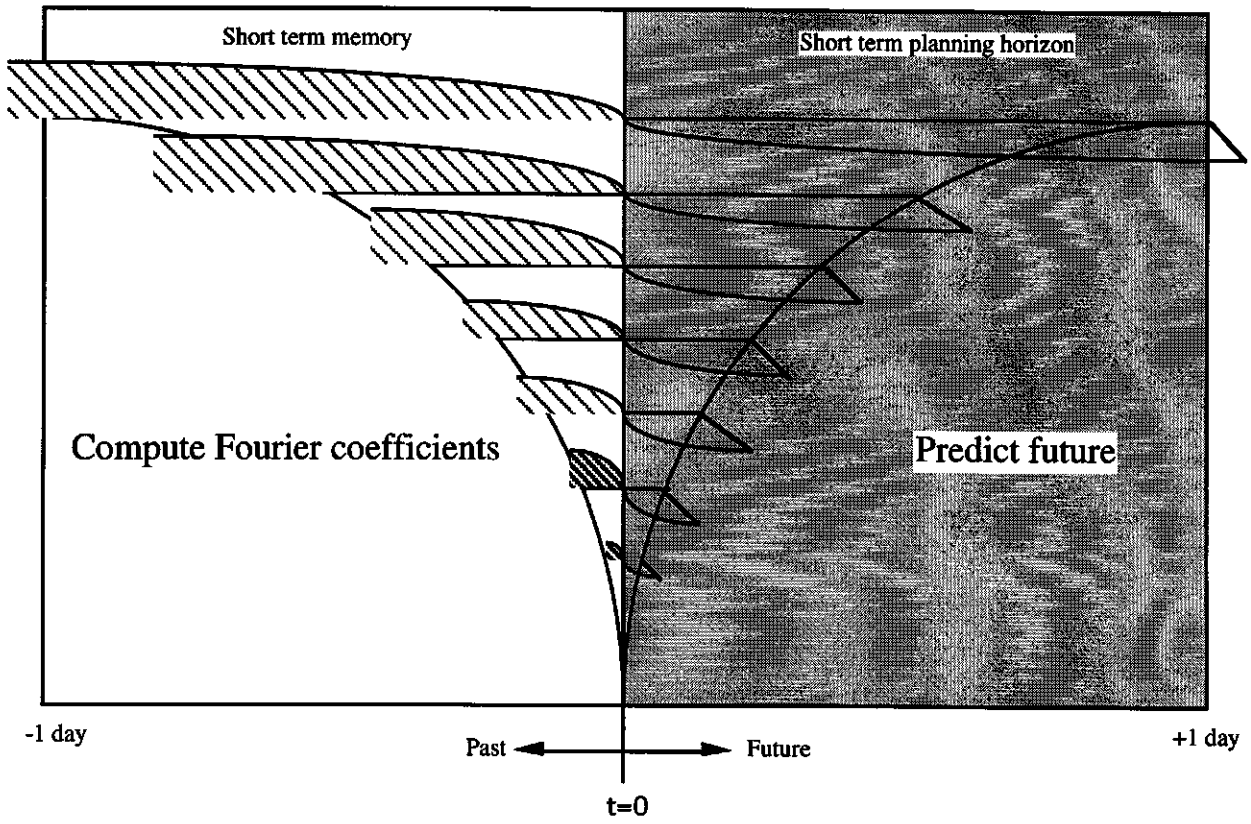


Figure 1-6. Hierarchical Spatio-temporal Planning

In a real-time system, plans must be regenerated periodically to cope with changing and unforeseen world conditions. Cyclic replanning may occur at periodic intervals. Emergency replanning begins immediately upon the detection of an emergency condition. Under full alert status, the cyclic replanning interval should be about an order of magnitude less than the planning horizon (or about equal to the expected output subtask time duration). This requires that real-time planners be able to search to the planning horizon about an order of magnitude faster than real time (this requires to have time intervals shorter that the ones selected for the real time process). This is possible only if the depth and resolution of search is limited through hierarchical planning.

Plan executors at each level are responsible for reacting to feedback every control cycle interval. Control cycle intervals are inversely proportional to the control loop bandwidth. Typically, the control cycle interval is an order of magnitude less than the expected output subtask duration. If the feedback indicates the failure of a planned subtask, the executor branches immediately (i.e. in one control cycle interval) to a preplanned emergency subtask. The planner simultaneously selects or generates an error recovery sequence which is substituted for the former plan which failed. Plan executors are also described in more detail in Chapter 6 of this book.

When a task goal is achieved at time t=0, it becomes a task completion event in the

historical trace. To the extent that a historical trace is an exact duplicate of a former plan, there were no surprises. For example, the plan was followed, and every task was accomplished as planned. To the extent that a historical trace is different from the former plan, there were surprises. The average size and frequency of surprises (i.e. differences between plans and results) is a measure of effectiveness of a planner.

At each level in the control hierarchy, the difference vector between planned (i.e. predicted) and observed events is an error signal, that can be used by executor submodules for servo feedback control (i.e. error correction), and by VJ modules for evaluating success and failure.

In the subsequent subsections, the system architecture outlined above will be elaborated and the functionality of the computational submodules for behavior generation, world modeling, sensory processing, and value judgment will be discussed.

1.4 Behavior Generation

This module of intelligent system receives a goal, retrieves relevant knowledge in the World Model and creates strings of Tasks for the Actuators (or the similar modules below in the hierarchy; the latter consider them their "goals".)

Definition: Space-time (spatio-temporal) representation presumed description of the process as a sequence of time-tagged states (temporal sequence) in which each state is a vector in the space with coordinates corresponding to all variables of the process (including input, output, and inner states variables.)

Definition: Behavior is the ordered set of consecutive-concurrent changes (in time) of the states registered at the output of a system (in space). In a goal-oriented system, behavior is the result of executing a series of tasks.

Definition: A task is a piece of work to be done, or an activity to be performed. It can be described as a data structure representing the assignment.

Definition: Action is an effort generated by the actuator producing changes in the World.

Axiom: Any intelligent system contains knowledge required to perform at least one set of tasks.

Definition: Goal the state to be achieved or an objective toward which task activity is directed (e.g. a particular event). A goal can be considered an event which successfully terminates a task.

Definition: A task command is an instruction to perform a named task. This is an assignment presented in the code pertaining to a particular module of the system. A task command may have the form:

DO <Task_name(parameters)> AFTER <Start State (or Event)> UNTIL <Goal State (or Event)>

Each task in this set can be assigned a name. The task vocabulary is the set of task names assigned to the set of tasks the system is capable of performing. For creatures capable of learning, the task vocabulary is not fixed in size. It can be expanded through learning, training, or programming. It may shrink from forgetting or program deletion.

Typically, a task is performed by a one or more "agents" on one or more objects. The performance of a task can usually be described as an activity which begins with a start-event and is directed toward a goal-event. This is illustrated in Figure 1-7.

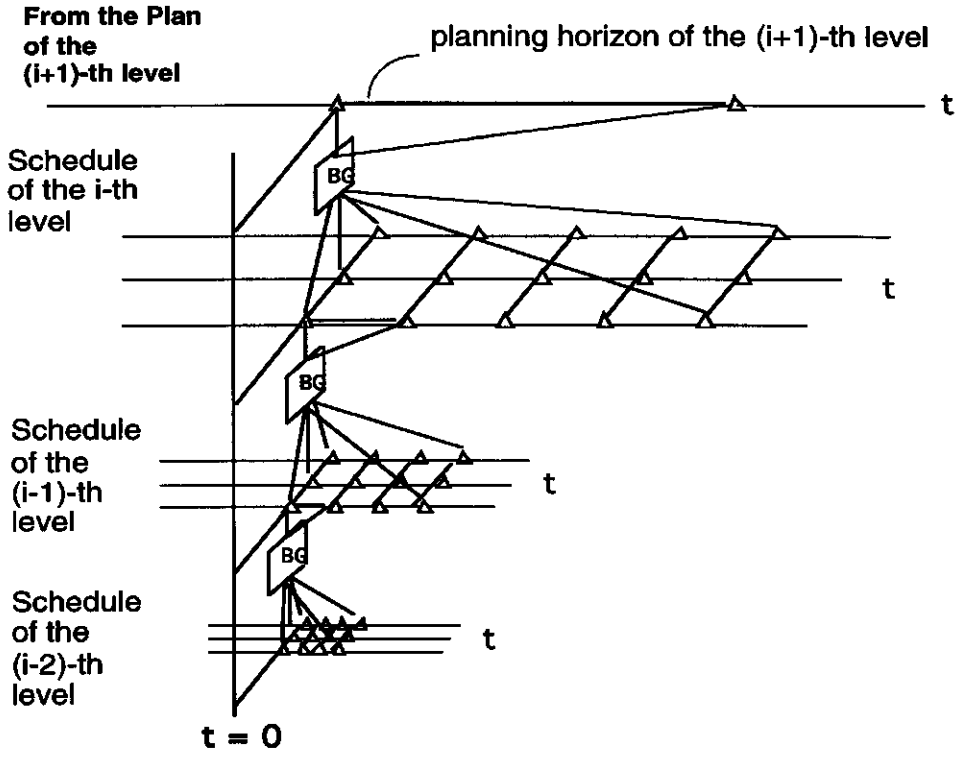


Figure 1-7. Spatio-temporal tasks distribution as a part of BG process

Task knowledge is knowledge of how to perform a task, including information concerning which tools, materials, time, resources, information, and conditions are required, plus information regarding which costs, benefits and risks to be expected.

Task knowledge may be expressed implicitly in fixed circuitry, either in the neuronal connections and synaptic weights of the brain, or in algorithms, software, and computing hardware. Task knowledge may also be expressed explicitly in data structures, either in the neuronal substrate or in a computer memory.

Task frame

Definition: A task frame is a data structure in which task knowledge can be stored.

In systems where task knowledge is explicit, a task frame [24] can be defined for each task in the task vocabulary. An example of a task frame is:

TASKNAME	-- name of the task
type	-- generic or specific
actor	-- agent performing the task
action	-- activity to be performed
object	-- thing to be acted upon
goal	-- state (or event) that successfully terminates or renders the task successful
parameters	-- priority
	-- status (e.g. active, waiting, inactive)
	-- timing requirements
	-- source of task command
requirements	-- tools, time, resources, and materials needed to perform the task
	-- enabling conditions that must be satisfied to begin, or continue, the task
	-- disabling conditions that will prevent, or interrupt, the task
	-- information that may be required
procedures	-- a state-graph or state-table defining a plan for executing the task
	-- functions that may be called
	-- algorithms that may be needed
effects	-- expected results of task execution
	-- expected costs, risks, benefits
	-- estimated time to complete

Explicit representation of task knowledge in task frames has a variety of uses. For example, task planners may use it for generating hypothesized actions. The world model may use it for predicting the results of hypothesized actions. The value judgment system may use it for computing how important the goal is and how many resources to expend in pursuing it. Plan executors may use it for selecting what to do next.

Task knowledge is typically difficult to discover, but once known, can be readily transferred to other tasks. Task knowledge may be acquired by trial and error learning, but more often it is acquired from a teacher, or from written or programmed instructions. For example, the common household task of preparing a food dish is typically performed by following a recipe. A recipe is an informal task frame for cooking. Gourmet dishes rarely result from reasoning about possible combinations of ingredients, still less from random trial and error combinations of food

stuffs. Exceptionally good recipes often are closely guarded secrets that, once published, can easily be understood and followed by others.

Making steel is a more complex task example. The human race took many millennia to discover how to make steel. However, once known, the recipe for making steel can be implemented by persons of ordinary skill and intelligence.

In most cases, the ability to successfully accomplish complex tasks depends more on the amount of task knowledge stored in task frames (particularly in the procedure section) than on the sophistication of planners in reasoning about tasks.

Behavior Generation

Behavior generation is inherently a hierarchical process. At each level of the behavior generation hierarchy, tasks are decomposed into subtasks that become task commands to the next lower level. At each level of a behavior generation hierarchy there exists a task vocabulary and a corresponding set of task frames. Each task frame contains a procedure state-graph. Each node in the procedure state-graph must correspond to a task name in the task vocabulary at the next lower level.

Behavior generation consists of both spatial and temporal decomposition. Spatial decomposition partitions a task into jobs to be performed by different subsystems. Spatial task decomposition results in a tree structure, where each node corresponds to a BG module, and each arc of the tree corresponds to a communication link in the chain of command (see Figure 1-3).

In a plan involving concurrent job activity by different subsystems, there may be requirements for coordination, or mutual constraints. For example, a start-event for a subtask activity in one subsystem may depend on the goal-event for a subtask activity in another subsystem. Some tasks may require concurrent coordinated cooperative action by several subsystems. Both planning and execution of subsystem plans may thus need to be coordinated.

There may be several alternative ways to accomplish a task. Alternative task or job decompositions can be represented by an AND/OR graph in the procedure section of the task frame. The decision as to which of several alternatives to choose is made through a series of interactions between the BG, WM, SP, and VJ modules. Each alternative may be analyzed by the BG module hypothesizing it, WM predicting the result, and VJ evaluating the result. The BG module then chooses the "best" alternative as the plan to be executed.

BG modules

In the control architecture defined in Figure 1-3, each level of the hierarchy contains one or more BG modules. At each level, there is a BG module for each subsystem being controlled. The function of the BG modules are to decompose task commands into subtask commands.

Input to BG modules consists of commands and priorities from BG modules at the next

higher level, plus evaluations from nearby VJ modules, plus information about past, present, and predicted future states of the world from nearby WM modules. Output from BG modules may consist of subtask commands to BG modules at the next lower level, plus status reports, plus "What Is?" and "What If?" queries to the WM about the current and future states of the world.

Temporal decomposition partitions each job into sequential subtasks along the time line. The result is a set of subtasks, all of which when accomplished, achieve the task goal, as illustrated in Figure 1-8. The term "spatial decomposition" should be understood as representation in a coordinate system in which each coordinate represents a particular variable of the process.

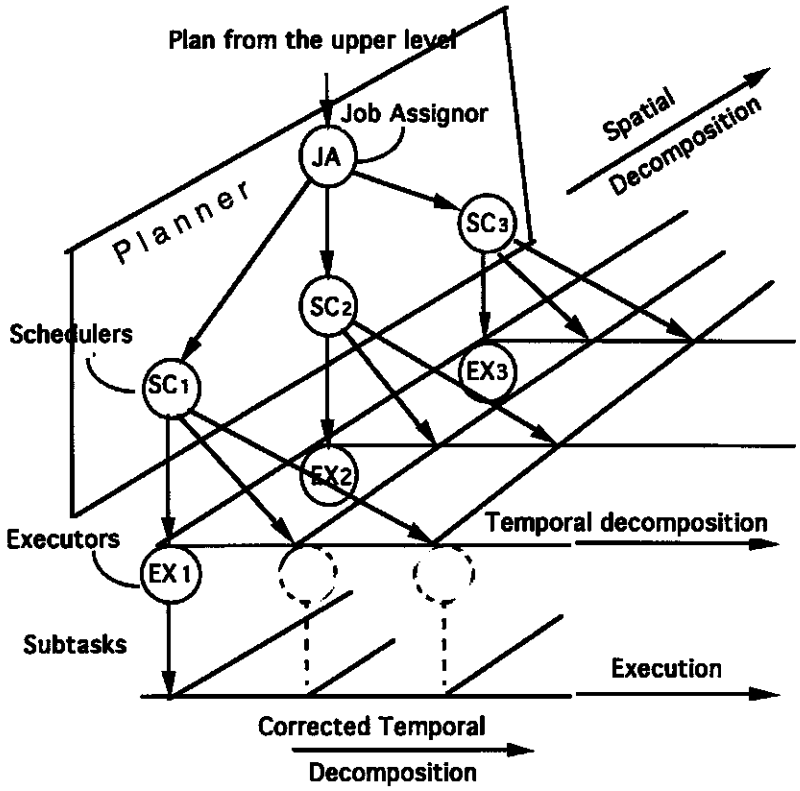


Figure 8. The job assignment JA module performs a spatial decomposition of the task. The schedulers SC(i) perform a temporal decomposition. The executors(i) corrects the temporal decomposition and execute the plans generated by the planners.

Planners, in turn, consist of two components: Job Assignor and Scheduler. Job Assignor performs the spatial decomposition (among the coordinates, i.e. among the actuators which will perform the job.) For each task decomposition ("spatial" decomposition) generated by JA, the temporal distribution of all subtasks is done by the Scheduler. After a number of such tentative spatial-temporal distributions, the best of them is selected, and this concludes the process of Planning.

1) The job assignment sublevel -- JA submodule

The JA submodule is responsible for spatial task decomposition. It partitions the input task command into N spatially distinct jobs to be performed by N physically distinct subsystems, where N is the number of subsystems currently assigned to the BG module. The JA submodule many assign tools and allocate physical resources (such as arms, hands, legs, sensors, tools, and materials) to each of its subordinate subsystems for their use in performing their assigned jobs. These assignments are not necessarily static. For example, the job assignment submodule at the individual level may, at one moment, assign an arm to the manipulation subsystem in response to a <use tool> task command, and later, assign the same arm to the attention subsystem in response to a <touch/feel> task command.

The job assignment submodule selects the coordinate system in which the task decomposition at that level is to be performed. In supervisory or telerobotic control systems such as defined by NASREM [14], the JA submodule at each level may also determine the amount and kind of input to accept from a human operator.

2) the Scheduler submodule -- SC(j) submodules j = 1, 2, . . , N

For each of the N subsystems, there exists a scheduler submodule SC(j). Each scheduler submodule is responsible for decomposing the job assigned to its subsystem into a temporal sequence of planned subtasks.

Scheduler submodules SC(j) may be implemented by case-based planners that simply select partially or completely prefabricated plans, scripts, or schema [20-22] from the procedure sections of task frames. This may be done by evoking situation/action rules of the form, IF(case_x)/THEN(use_plan_y). The planner submodules may complete partial plans by providing situation dependent parameters.

The range of behavior that can be generated by a library of prefabricated plans at each hierarchical level, with each plan containing a number of conditional branches and error recovery routines, can be extremely large and complex. For example, nature has provided biological creatures with an extensive library of genetically prefabricated plans, called instinct. For most species, case-based planning using libraries of instinctive plans has proven adequate for survival and gene propagation in a hostile natural environment.

Scheduler submodules SC(j) may also be implemented by search-based planners that search the space of possible actions. This requires the evaluation of alternative hypothetical sequences of subtasks, as illustrated in Figure 1-9. Each planner SC(j) hypothesizes some action or series of actions, the WM module predicts the effects of those action(s), and the VJ module computes the value of the resulting expected states of the world, as depicted in Figure 1-9(a). This results in a game (or search) graph, as shown in Fig. 1-9(b). The path through the game graph

chosen by the Planner's "selector," i.e. leading to the state with the best value, becomes the plan to be executed by EX(j). In either case-based or search-based planning, the resulting plan may be represented by a state-graph, as shown in Figure 1-9(c). Plans may also be represented by gradients, or other types of fields, on maps [36], or in configuration space.

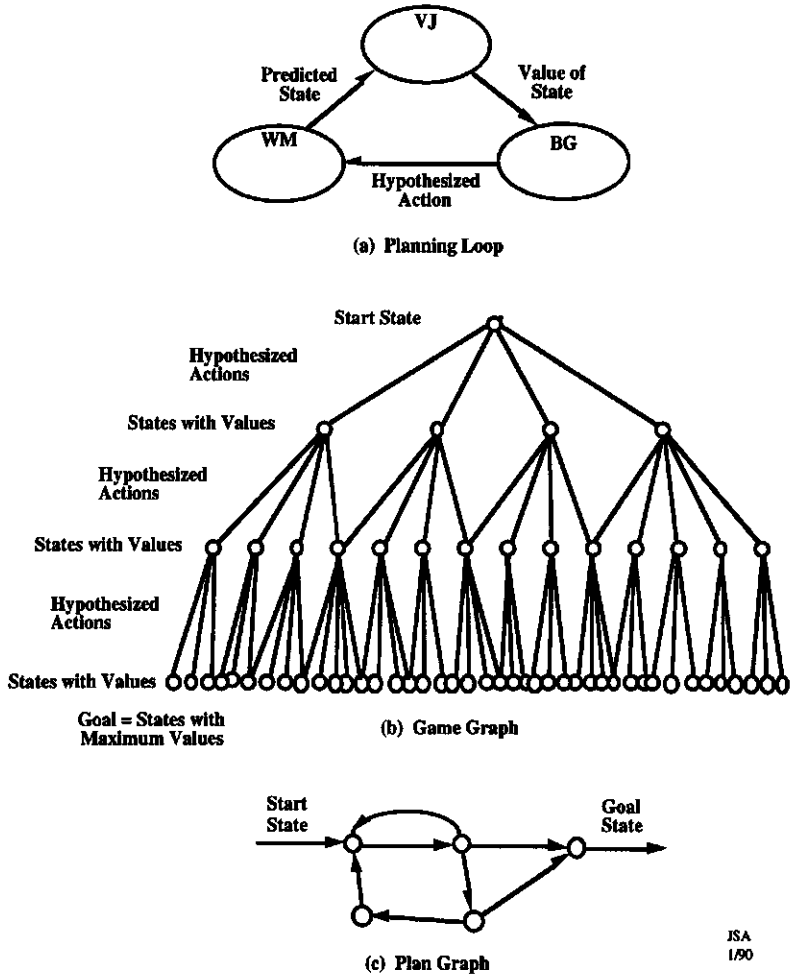


Figure 9. The planning loop (a) produces a game graph (b). A trace in the game graph from the start state to a goal state is a plan that can be represented as a plan graph (c). Nodes in the game graph correspond to edges in the plan graph, and edges in the game graph correspond to nodes in the plan graph. Multiple edges exiting nodes in the plan graph correspond to conditional branches.

Job commands to each planner submodule may contain constraints on time, or specify job-start and job-goal events. A job assigned to one subsystem may also require synchronization or coordination with other jobs assigned to different subsystems. These constraints and coordination requirements may be specified by, or derived from, the task frame. Each scheduler's SC(j) submodule is responsible for coordinating its schedule with schedules generated by each of the other N-1 schedulers at the same level, and checking to determine if there are mutually conflicting

constraints. If conflicts are found, constraint relaxation algorithms [24] may be applied, or negotiations conducted between SC(j) (cooperating Schedulers,) until a solution is discovered. If no solution can be found, the schedulers report failure to the job assignment submodule, and a new job assignment may be tried, or failure may be reported to the next higher level BG module.

3) the executor sublevel -- EX(j) submodules

There is an executor EX(j), or a group of executors for each planner PL(j). The executor submodules are responsible for successfully executing the plan state-graphs generated by Schedulers within their respective Planners. At each tick of the state clock, each executor measures the difference between the current world state and its current plan subgoal state, and issues a subcommand designed to null the difference. When the world model indicates that a subtask in the current plan is successfully completed, the executor steps to the next subtask in that plan. When all the subtasks in the current plan are successfully executed, the executor steps to the first subtask in the next plan. If the feedback indicates the failure of a planned subtask, the executor branches immediately to a preplanned emergency subtask. Meanwhile its planner begins work selecting or generating a new plan which can be substituted for the former plan which failed. Output subcommands produced by executors at level i become input commands to job assignment submodules in BG modules at level i-1.

Planners PL(j) operate on the future. For each subsystem, there is a planner that is responsible for providing a plan that extends to the end of its planning horizon. Executors EX(j) operate in the present. For each subsystem, there is an executor that is responsible for monitoring the current (t=0) state of the world and executing the plan for its respective subsystem. Each executor performs a READ-COMPUTE-WRITE operation once each control cycle. At each level, each executor submodule closes a reflex arc, or servo loop. Thus, executor submodules at the various hierarchical levels form a set of nested servo loops. Executor loop bandwidths decrease on average about an order of magnitude at each higher level.

The Behavior Generating Hierarchy

Task goals and task decomposition functions often have characteristic spatial and temporal properties. For any task, there exists a hierarchy of task vocabularies that can be overlaid on the spatial/temporal hierarchy shown in Figure 1-9.

Example

Level 1 is where commands for coordinated velocities and forces of body components (such as arms, hands, fingers, legs, eyes, torso, and head) are decomposed into motor commands to individual actuators. Feedback controllers are used to compensate for deviation from the plans created for the position, velocity, and force of individual actuators. In vertebrates, this is the level

of the motor neuron and stretch reflex.

Level 2 is where commands for maneuvers of body components are decomposed into smooth coordinated dynamically efficient trajectories. Feedback servos coordinated trajectory motions. This is the level of the spinal motor centers and the cerebellum.

Level 3 is where commands to manipulation, locomotion, and attention subsystems are decomposed into collision free paths that avoid obstacles and singularities. Feedback servos movements relative to surfaces in the world. This is the level of the red nucleus, the substantia nigra, and the primary motor cortex.

Level 4 is where commands for an individual to perform simple tasks on single objects are decomposed into coordinated activity of body locomotion, manipulation, attention, and communication subsystems. Feedback initiates and sequences subsystem activity. This is the level of the basal ganglia and pre-motor frontal cortex.

Level 5 is where commands for behavior of a small group of intelligent agents are decomposed into interactions between the self and nearby objects or agents. Planners together with the feedback compensation of the executors initiate and steer whole set of activities required by the task. Behavior generating levels 5 and above are hypothesized to reside in temporal, frontal, and limbic cortical areas.

Level 6 is where commands for behavior of the individual agents, which are members of multiple groups, are decomposed into small group interactions. Plans and feedback compensations commands steer small group interactions.

Level 7 (arbitrarily, the highest level in our discussion) is where long range goals are selected and plans are made for long range behavior relative to the world as a whole. Feedback steers progress toward long range goals.

The mapping of BG functionality onto levels one to four defines the control functions necessary to control a single intelligent individual in performing simple task goals. Functionality at levels one through three is more or less fixed and specific to each species of intelligent system [31]. At level four and above, the mapping becomes more task and situation dependent. Levels five and above define the control functions necessary to control the relationships of an individual relative to others in groups, multiple groups, and the world as a whole.

There is good evidence that hierarchical layers develop in the sensory-motor system, both in the individual brain as the individual matures, and in the brains of an entire species as the species evolves. It can be hypothesized that the maturation of levels in humans gives rise to Piaget's "stages of development" [32].

Of course, the biological motor system is typically much more complex than is suggested by the example model described above. In the brains of higher species there may exist multiple hierarchies that overlap and interact with each other in complicated ways. For example in primates, the pyramidal cells of the primary motor cortex have outputs to the motor neurons for

direct control of fine manipulation as well as the inferior olive for teaching behavioral skills to the cerebellum [33]. There is also evidence for three parallel behavior generating hierarchies that have developed over three evolutionary eras [34]. Each BG module may thus contain three or more competing influences: 1) the most basic (IF it smells good, THEN eat it), 2) a more sophisticated (WAIT until the "best" moment) where best is when success probability is highest, and 3) a very sophisticated (WHAT are the long range consequences of my contemplated action, and what are all my options).

On the other hand, some motor systems may be less complex. Not all species have the same number of levels. Insects, for example, may have only two or three levels, while adult humans may have more than seven. In robots, the functionality required of each BG module depends upon the complexity of the subsystem being controlled. For example, one robot gripper may consist of a dexterous hand with 15 to 20 force servoed degrees of freedom. Another gripper may consist of two parallel jaws actuated by a single pneumatic cylinder. In simple systems, some BG modules (such as the Primitive level) may have no function (such as dynamic trajectory computation) to perform. In this case, the BG module will simply pass through unchanged input commands (such as <Grasp>).

1.5 The World Model

Knowledge about the world should be maintained in a way that allows to support the needs of the behavior generation processes, and on the other hand, not to sacrifice the integrity of this knowledge which reflects both the unity and the diversity of the external world.

Definition: The world model is an intelligent system's internal representation of the external world. It is the system's best estimate of objective reality which integrates views pertinent to the different spatial and temporal scales (resolutions).

Over 100 years ago in the West a clear distinction between an internal representation of the world that exists in the mind, and the external world of reality, was first made by Schopenhauer [35]. In the East, it has been a central theme of Buddhism for millennia. Today, the concept of an internal world model is crucial to understanding perception and cognition. The world model provides the intelligent system with the information necessary to reason about objects, space, and time. The world model contains knowledge of things that are not directly and immediately observable. It enables the system to integrate noisy and intermittent sensory input from many different sources into a single reliable representation of spatio-temporal reality.

Knowledge in the world model may be represented either implicitly or explicitly. Implicit world knowledge may be embedded in the control and sensory processing algorithms and interconnections of a brain, or of a computer system. Explicit world knowledge may be represented in either natural or artificial systems by data in database structures such as maps, lists,

and semantic nets. Explicit world models require computational modules capable of map transformations, indirect addressing, and list processing. Computer hardware and software techniques for implementing these types of functions are well known. Neural mechanisms with such capabilities are discussed later in this Chapter.

WM Modules

The WM modules in each node of the organizational hierarchies presented in Figures 2 and 3 perform the following functions.

1) WM modules maintain the knowledge database, keeping it current and consistent. In this role, the WM modules perform the functions of a database management system. They update WM state estimates based on correlations and differences between world model predictions and sensory observations at each hierarchical level. The WM modules enter newly recognized entities, states, and events into the knowledge database, and delete entities and states determined by the sensory processing modules to no longer exist in the external world. The WM modules also enter estimates, generated by the VJ modules, of the reliability of world model state variables. Believability or confidence factors are assigned to many types of state variables.

2) WM modules generate predictions of expected sensory input for use by the appropriate sensory processing SP modules. In this role, a WM module performs the functions of a signal generator, a graphics engine, or state predictor, generating predictions that enable the sensory processing system to perform correlation and predictive filtering. WM predictions are based on the state of the task and estimated states of the external world. For example in vision, a WM module may use the information in an object frame to generate real-time predicted images which can be compared pixel by pixel, or entity by entity, with observed images.

3) WM modules answer "What is?" questions asked by the planners and executors in the corresponding level BG modules. In this role, the WM modules perform the function of database query processors, question answering systems, or data servers. World model estimates of the current state of the world are also used by BG module planners as a starting point for planning. Current state estimates are used by BG module executors for servoing and branching on conditions.

4) WM modules answer "What if?" questions asked by the planners in the corresponding level BG modules. In this role, the WM modules perform the function of simulation by generating expected status resulting from actions hypothesized by the BG planners. Results predicted by WM simulations are sent to value judgment VJ modules for evaluation. For each BG hypothesized action, a WM prediction is generated, and a VJ evaluation is returned to the BG planner. This BG-WM-VJ loop enables BG planners to select the sequence of hypothesized actions producing the best evaluation as the plan to be executed.

Data structures for representing explicit knowledge are defined to reside in a knowledge

database that is hierarchically organized and distributed such that there is a knowledge database for each WM module in each node at every level of the system hierarchy. The communication system provides data transmission and switching services that make the WM modules and the knowledge database behave like a global virtual common memory in response to queries and updates from the BG, SP, and VJ modules. The communication interfaces with the WM modules in each node which provides a window into the knowledge database for each of the computing modules in that node.

Knowledge Representation

The world model knowledge database contains both a-priori information that is available to the intelligent system before action begins, and a-posterior knowledge which is gained from sensing the environment as the action proceeds. The knowledge database contains information about space, time, entities, events, and states of the external world. It contains information about the intelligent system itself, such as values assigned to motives, drives, and priorities; values assigned to goals, objects, and events; parameters embedded in kinematic and dynamic models of the limbs and body; states of internal pressure, temperature, clocks, and blood chemistry or fuel level; plus the states of all of the processes currently executing in each of the BG, SP, WM, and VJ modules.

Knowledge about space is represented in maps. Knowledge about entities, events, and states is represented in lists or frames. Knowledge about the laws of physics, chemistry, optics, and the rules of logic and mathematics are represented as algorithms and their parameters in the WM functions that generate predictions and simulate results of hypothetical actions. Physical knowledge may be represented as algorithms, formulae, or IF/THEN rules of what happens under certain situations, such as when things are pushed, thrown, dropped, handled, or burned.

The correctness and consistency of world model knowledge is verified by sensory processing mechanisms that measure differences between world model predictions and sensory observations.

Geometrical Space

From psychophysical evidence, Gibson [36] concludes that the perception of geometrical space is primarily in terms of "medium, substance, and the surfaces that separate them". The medium through which the world is viewed is the air, water, fog, smoke, or falling snow. Substance is the material, such as earth, rock, wood, metal, flesh, grass, clouds, or water, that comprise the interior of objects. The surfaces that separate the viewing medium from the viewed objects are observed by the sensory system. The sensory input thus describes the external physical world primarily in terms of surfaces.

Surfaces are selected as the fundamental element for representing space in the proposed WM knowledge database. Volumes are treated as regions between surfaces. Objects are defined as circumscribed, often closed, surfaces. Lines, points and vertices lie on, and may define surfaces. Spatial relationships on surfaces are represented by maps.

Maps

Definition: A map is a multidimensional representation that puts in correspondence objects in the space and properties of the space with the multidimensional grid determined by the coordinate system. An example: a two dimensional map of the 3D reality is a two dimensional database that defines correspondence of the objects and properties of the space with a mesh or coordinate grid on a surface.

The surface represented by a map may be, but need not be, flat. For example, a map may be defined on a surface that is draped over, or even wrapped around, a 3-dimensional volume.

Assertion: Maps can be used to describe the distribution of entities in space. It is always possible and often useful to project the physical 3-D world onto a 2-D surface defined by a map. For example, most commonly used maps are produced by projecting the world onto the 2-D surface of a flat sheet of paper, or the surface of a globe. One great advantage of such a projection is that it reduces the dimensionality of the world from three to two. This produces an enormous saving in the amount of memory required for a database representing space. The saving may be as much as three orders of magnitude, or more, depending on the resolution along the projected dimension.

Map Overlays

Most of the useful information lost in the projection from 3-D space to a 2-D surface can be recovered through the use of map overlays.

Definition: A map overlay is an assignment of values, or parameters, to points on the map.

A map overlay can represent spatial relationships between 3-D objects. For example, an object overlay may indicate the presence of buildings, roads, bridges, and landmarks at various places on the map. Objects that appear smaller than a pixel on a map can be represented as icons. Larger objects may be represented by labeled regions that are projections of the 3-D objects on the 2-D map. Objects appearing on the map overlay may be cross referenced to an object frame database elsewhere in the world model. Information about the 3-D geometry of objects on the map may be represented in the object frame database.

Map overlays can also indicate attributes associated with points (or pixels) on the map. One of the most common map overlays defines terrain elevation. A value of terrain elevation (z) overlaid at each (x,y) point on a world map produces a topographic map.

A map can have any number of overlays. Map overlays may indicate brightness, color,

temperature, even "behind" or "in-front". A brightness or color overlay may correspond to a visual image. For example, when aerial photos or satellite images are registered with map coordinates, they become brightness or color map overlays.

Map overlays may indicate terrain type, or region names, or can indicate values, such as cost or risk, associated with regions. Map overlays can indicate which points on the ground are visible from a given location in space. Overlays may also indicate contour lines and grid lines such as latitude and longitude, or range and bearing.

Map overlays may be useful for a variety of functions. For example, terrain elevation and other characteristics may be useful for route planning in tasks of manipulation and locomotion. Object overlays can be useful for analyzing scenes and recognizing objects and places.

A map typically represents the configuration of the world at a single instant in time, i.e. a snapshot. Motion can be represented by overlays of state variables such as velocity or image flow vectors, or traces (i.e. trajectories) of entity locations. Time may be represented explicitly by a numerical parameter associated with each trajectory point, or implicitly by causing trajectory points to fade, or be deleted, as time passes.

Definition: A map pixel frame is a frame that contains attributes and attribute-values attached to that map pixel.

Theorem: A set of map overlays is equivalent to a set of map pixel frames.

Proof: If each map overlay defines a parameter value for every map pixel, then the set of all overlay parameter values for each map pixel defines a frame for that pixel. Conversely, the frame for each pixel describes the region covered by that pixel. The set of all pixel frames thus defines a set of map overlays, one overlay for each attribute in the pixel frames. QED

For example, a pixel frame may describe the color, range, and orientation of the surface covered by the pixel. It may describe the name of (or pointer to) the entities to which the surface covered by the pixel belongs. It may also contain the location, or address, of the region covered by the pixel in other coordinate systems.

In the case of a video image, a map pixel frame might have the following form:

PIXEL_NAME = location index on map (AZ, EL) (Sensor egosphere coordinates)

brightness	I
color	I_r, I_b, I_g
spatial brightness gradient	dI/dAZ, dI/dEL (sensor egosphere coordinates)
temporal brightness gradient	dI/dt
image flow direction	B (velocity egosphere coordinates)
image flow rate	dA/dt (velocity egosphere coordinates)
range	R to surface covered (from egosphere origin)
head egosphere location	az, el of egosphere ray to surface covered

inertial egosphere location	a, e of egosphere ray to surface covered
world map location	x, y, z of map point on surface covered
linear feature pointer	pointer to frame of line, edge, or vertex covered by pixel
surface feature pointer	pointer to frame of surface covered by pixel
object pointer	pointer to frame of object covered by pixel
object map location	X, Y, Z of surface covered in object coordinates
group pointer	pointer to group covered by pixel

Indirect addressing through pixel frame pointers allows the value of state-variables assigned to objects or situations to be inherited by map pixels. For example, value state-variables such as attraction-repulsion, love-hate, fear-comfort assigned to objects and map regions can also be assigned through inheritance to individual map and egosphere pixels.

Some experimental evidence suggests that map pixel frames exist in the mammalian visual system. For example, neuron firing rates in visual cortex have been observed to represent the values of attributes such as edge orientation, edge and vertex type, and motion parameters such as velocity, rotation, and flow field divergence. These firing rates are observed to be registered with retinotopic brightness images [37, 60].

Map resolution

The resolution required for a world model map depends on how the map is generated and how it is used. All overlays need not have the same resolution. For predicting sensory input, world model maps should have resolution comparable to the resolution of the sensory system. For vision, map resolution may be on the order of 64K to a million pixels. This corresponds to image arrays of 256 x 256 pixels to 1000 x 1000 pixels respectively. For other sensory modalities, resolution can be considerably less.

For planning, different levels of the control hierarchy require maps of different scale. At higher levels, plans cover long distances and times, and require maps of large area, but low resolution. At lower levels, plans cover short distances and times, and maps need to cover small areas with high resolution [18].

World model maps generated solely from symbolic data in long term memory may have resolution on the order of a few thousand pixels or less. For example, few humans can recall from memory the relative spatial distribution of as many as a hundred objects, even in familiar locations such as their own homes. The long term spatial memory of an intelligent creature typically consists of a finite number of relatively small regions that may be widely separated in space. Examples are our home, office, or school, the homes of friends and relatives, etc. These known regions are typically connected by linear pathways that contain at most a few hundred known waypoints and branchpoints. The remainder of the world is known little, or not at all. Unknown regions, which

constitute the vast majority of the real world, occupy little or no space in the world model.

The efficient storage of maps with extremely non-uniform resolution can be accomplished in a computer database by quadtrees [38], hash coding, or other sparse memory representations [39]. Pathways between known areas can be economically represented by graph structures either in neuronal or electronic memories. Neural net input-space representations and transformations such as are embodied in a CMAC [40, 41] give insight as to how non-uniformly dense spatial information might be represented in the brain.

Maps and Egospheres

It is well known that neurons in the brain, particularly in the cortex, are organized as 2-D arrays or maps. It is also known that conformal mappings of image arrays exist between the retina, the lateral geniculate, the superior colliculus, and several cortical visual areas. Similar mappings exist in the auditory and tactile sensory systems. For every map, there exists a coordinate system, and each map pixel has coordinate values. On the sensor egosphere, pixel coordinates are defined by the physical position of the pixel in the sensor array. The position of each pixel in other map coordinate systems can be defined either by neuronal interconnections, or by transform parameters contained in each pixel's frame.

There are three general types of map coordinate systems that are important to an intelligent system: world coordinates, object coordinates, and egospheres.

World coordinates

World coordinate maps are typically flat 2-D representations that are projections of the surface of the earth along the local perpendicular to the surface of the sphere. World coordinates are often expressed in a Cartesian frame, and referenced to a point in the world. In most cases, the origin is an arbitrary point on the ground. The z axis is defined by the vertical, and the x and y axes define points on the horizon. For example, y may point North and x East. The value of z is often set to zero at sea level.

World coordinates may also be referenced to a moving point in the world. For example, the origin may be associated with (attached to) some moving object in the world. In this case, stationary pixels on the world map must be scrolled as the reference point moves.

There may be several world maps with different resolutions and ranges.

Object coordinates

Object coordinates are defined with respect to features in an object. For example, the origin might be defined as the center of gravity with the coordinate axes defined by axes of symmetry, faces, edges, vertices, or skeletons [42]. There are a variety of surface representations that have been suggested for representing object geometry. Among these are generalized cylinders

[43, 44], B-splines [45], quadtrees [38], and aspect graphs [46]. Object coordinate maps are typically 2-D arrays of points painted on the surfaces of objects in the form of a grid or mesh. Other boundary representation can usually be transformed into this form.

Object map overlays can indicate surface characteristics such as texture, color, hardness, temperature, and type of material. Overlays can be provided for edges, boundaries, surface normal vectors, vertices, and pointers to object frames containing center lines, centroids, moments, and axes of symmetry.

Egospheres

An egosphere is a 2-dimensional spherical surface that is a map of the world as seen by an observer at the center of the sphere. Visible points on regions or objects in the world are projected on the egosphere wherever the line of sight from a sensor at the center of the egosphere to the points in the world intersect the surface of the sphere. Egosphere coordinates thus are polar coordinates defined by the self at the origin. As the self moves, the projection of the world flows across the surface of the egosphere.

Just as the world map is a flat 2-D (x,y) array with multiple overlays, so the egosphere is a spherical 2-D (AZ,EL) array with multiple overlays. Egosphere overlays can attribute brightness, color, range, image flow, texture, and other properties to regions and entities on the egosphere. Regions on the egosphere can thus be segmented by attributes, and egosphere points with the same attribute value may be connected by contour lines. Egosphere overlays may also indicate the trace, or history, of brightness values or entity positions over some time interval. Objects may be represented on the egosphere by icons, and each object may have in its database frame a trace, or trajectory, of positions on the egosphere over some time interval.

Map transformations

Theorem: If surfaces in real world space can be covered by an array (or map) of points in a coordinate system defined in the world, and the surface of a WM egosphere is also represented as an array of points, then there exists a function **G** that transforms each point on the real world map into a point on the WM egosphere, and a function **G'** that transforms each point on the WM egosphere for which range is known into a point on the real world map.

Proof: Figure 1-10 shows the 3-D relationship between an egosphere and world map coordinates. For every point (x,y,z) in world coordinates, there is a point (AZ,EL,R) in ego centered coordinates which can be computed by the 3x3 matrix function **G**

$$(AZ,EL,R)^T = \mathbf{G}\ (x,y,z)^T$$

There, of course, may be more than one point in the world map that gives the same (AZ,EL)

values on the egosphere.

Only the (AZ,EL) with the smallest value of R will be visible to an observer at the center of the egosphere. The deletion of egosphere pixels with R larger than the smallest for each value of (AZ,EL) corresponds to the hidden surface removal problem common in computer graphics.

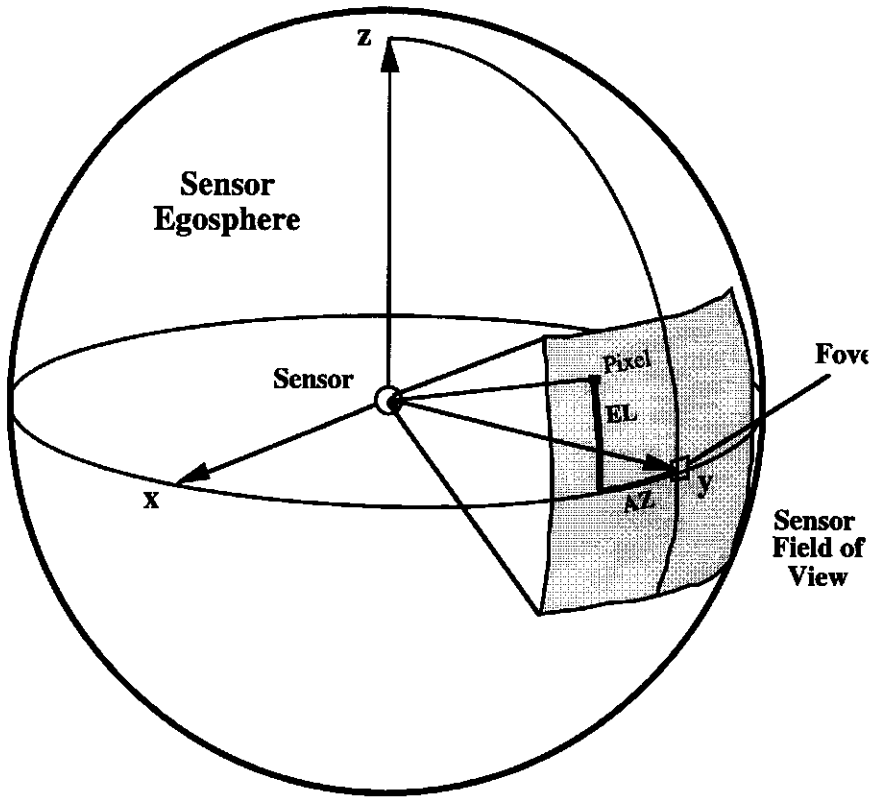


Figure 1-10. Sensor egosphere coordinates. Azimuth (AZ) is measured clockwise from the sensors y-axis in the x-y plane. Elevation (EL) is measured up and down (plus and minus) from the x-y plane.

For each egosphere pixel where R is known, (x,y,z) can be computed from (AZ,EL,R) by the function **G '**

$$(x,y,z)^T = \mathbf{G}'\,(AZ,EL,R)^T$$

Any point in the world topological map can thus be projected onto the egosphere (and vice versa when R is known). Projections from the egosphere to the world map will leave blank those map pixels that cannot be observed from the center of the egosphere. QED

There are 2×2 transformations of the form

$$(AZ,EL)^T = \mathbf{F}\,(az,el)^T$$

and

$$(az,el)^T = \mathbf{F}' (AZ,EL)^T$$

that can relate any map point (AZ,EL) on one egosphere to a map point (az,el) on another egosphere of the same origin. The radius R to any egosphere pixel is unchanged by the **F** and **F'** transformations between egosphere representations with the same origin.

As ego motion occurs (i.e. as the self object moves through the world), the egosphere moves relative to world coordinates and points on the egocentric maps flow across their surfaces. Ego motion may involve translation, or rotation, or both, in a stationary world, or a world containing moving objects. If egomotion is known, range to all stationary points in the world can be computed from observed image flow. Once range to any stationary point in the world is known, its pixel motion on the egosphere can be predicted from knowledge of egomotion. For moving points, prediction of pixel motion on the egosphere requires additional knowledge of object motion.

Egosphere Coordinate Systems

Our world model contains four different types of egosphere coordinates:

1) Sensor Egosphere Coordinates

The sensor egosphere is defined by the sensor position and orientation, and moves as the sensor moves. For vision, the sensor egosphere is the coordinate system of the retina. The sensor egosphere has coordinates of azimuth (AZ) and elevation (EL) fixed in the sensor system (such as an eye or a TV camera), as shown in Figure 1-10. For a narrow field of view, rows and columns (x,z) in a flat camera image array correspond quite closely to azimuth and elevation (AZ,EL) on the sensor egosphere. However, for a wide field of view, the egosphere and flat image array representations have widely different geometries. The flat image (x,z) representation becomes highly elongated for a wide field of view, going to infinity at plus and minus 90 degrees. The egosphere representation, in contrast, is well behaved over the entire sphere (except for singularities at the egosphere poles).

The sensor egosphere representation is useful for the analysis of wide angle vision such as occurs in the eyes of most biological creatures. For example, most insects and fish, many birds, and most prey animals such as rabbits have eyes with fields of view up to 180 degrees. Such eyes are often positioned on opposite sides of the head so as to provide almost 360 degree visual coverage. The sensor egosphere representation provides a tractable coordinate frame in which this type of vision can be analyzed.

2) Head Egosphere Coordinates

The head egosphere has (az,el) coordinates measured in a reference frame fixed in the head (or sensor platform). The head egosphere representation is well suited to fuse sensory data from multiple sensors, each of which has its own coordinate system. Vision data from multiple eyes or cameras can be overlaid and registered in order to compute range from stereo. Directional and range data from acoustic and sonar sensors can be overlaid on vision data. Data derived from different sensors, or from multiple readings of the same sensor, can be overlaid on the head egosphere to build up a single image of multidimensional reality.

Pixel data in sensor egosphere coordinates can be transformed into the head egosphere by knowledge of the position and orientation of the sensor relative to the head. For example, the position of each eye in the head is fixed and the orientation of each eye relative to the head is known from stretch sensors in the ocular muscles. The position of tactile sensors relative to the head is known from proprioceptive sensors in the neck, torso, and limbs.

Hypothesis: Neuronal maps on the tectum (or superior colliculus), and on parts of the extrastriate visual cortex, are represented in a head egosphere coordinate system.

Receptive fields from the two retinas are well known to be overlaid in registration on the tectum, and superior colliculus. Experimental evidence indicates that registration and fusion of data from visual and auditory sensors takes place in the tectum of the barn owl [47] and the superior colliculus of the monkey [48] in head egosphere coordinates. Motor output for eye motion from the superior colliculus apparently is transformed back into retinal egosphere coordinates. There is also evidence that head egosphere coordinates are used in the visual areas of the parietal cortex [49, 60].

3) Velocity Egosphere

The velocity egosphere is defined by the velocity vector and the horizon. The velocity vector defines the pole (y-axis) of the velocity egosphere, and the x-axis points to the right horizon as shown in Figure 1-11. The egosphere coordinates (A,B) are defined such that A is the angle between the pole and a pixel, and B is the angle between the y-o-z plane and the plane of the great circle flow line containing the pixel.

For egocenter translation without rotation through a stationary world, image flow occurs entirely along great circle arcs defined by B = constant. The positive pole of the velocity egosphere thus corresponds to the focus-of-expansion. The negative pole corresponds to the focus-of-contraction. The velocity egosphere is ideally suited for computing range from image flow.

EGOSPHERE RELATIONS

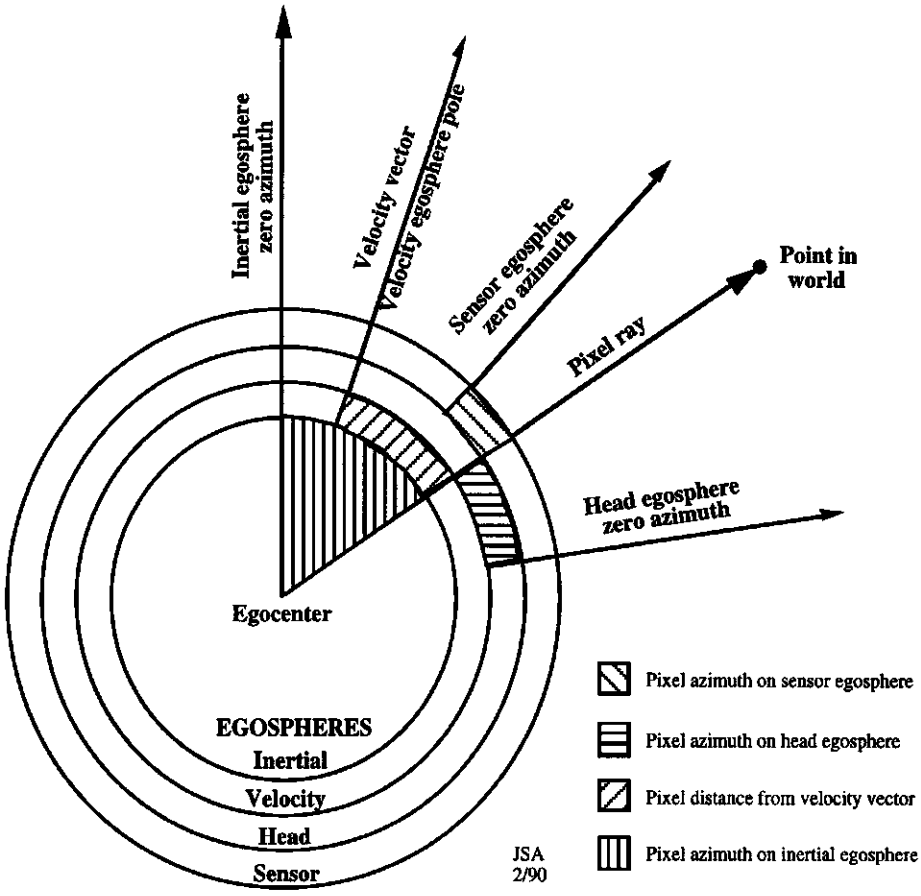


Figure 1-11. A 2-D projection of four egosphere representations illustrating angular relationships between egosperes. Pixels are represented on each egosphere such that images remain in registration. Pixel attributes detected on one egosphere may thus be inherited on others. Pixel resolution is not typically uniform on a single egosphere, nor is it necessarily the same for different egospheres, or for different attributes on the same egosphere.

4) Inertial Egosphere

The inertial egosphere has coordinates of azimuth measured from a fixed point (such as North) on the horizon, and elevation measured from the horizon.

The inertial egosphere does not rotate as a result of sensor or body rotation. On the inertial egosphere, the world is perceived as stationary despite image motion due to rotation of the sensors and the head.

Figure 1-11 illustrates the relationships between the four egosphere coordinate systems.

Pixel data in eye (or camera) egosphere coordinates can be transformed into head (or

sensor platform) egosphere coordinates by knowledge of the position and orientation of the sensor relative to the head. For example, the position of each eye in the head is fixed and the orientation of each eye relative to the head is known from stretch receptors in the ocular muscles (or pan and tilt encoders on a camera platform). Pixel data in head egosphere coordinates can be transformed into inertial egosphere coordinates by knowing the orientation of the head in inertial space. This information can be obtained from the vestibular (or inertial) system that measures the direction of gravity relative to the head and integrates rotary accelerations to obtain head position in inertial space. The inertial egosphere can be transformed into world coordinates by knowing the x,y,z position of the center of the egosphere. This is obtained from knowledge about where the self is located in the world. Pixels on any egosphere can be transformed into the velocity egosphere by knowledge of the direction of the current velocity vector on that egosphere. This can be obtained from a number of sources including the locomotion and vestibular systems.

All of the above egosphere transformations can be inverted, so that conversions can be made in either direction. Each transformation consists of a relatively simple vector function that can be computed for each pixel in parallel. Thus the overlay of sensory input with world model data can be accomplished in a few milliseconds by the type of computing architectures known to exist in the brain. In artificial systems, full image egosphere transformations can be accomplished within a television frame interval by state-of-the-art serial computing hardware. Image egosphere transformations can be accomplished in a millisecond or less by parallel hardware.

Hypothesis: The WM world maps, object maps, and egospheres are the brains data fusion mechanisms. They provide coordinate systems in which to integrate information from arrays of sensors (i.e. rods and cones in the eyes, tactile sensors in the skin, directional hearing, etc.) in space and time. They allow information from different sensory modalities (i.e. vision, hearing, touch, balance, and proprioception) to be combined into a single consistent model of the world.

Hypothesis: The WM functions that transform data between the world map and the various egosphere representations are the brain's geometry engine. They transform world model predictions into the proper coordinate systems for real-time comparison and correlation with sensory observations. This provides the basis for recognition and perception.

Transformations to and from the sensor egosphere, the inertial egosphere, the velocity egosphere, and the world map allow the intelligent system to sense the world from one perspective and interpret it in another. They allow the intelligent system to compute how entities in the world would look from another viewpoint. They provide the ability to overlay sensory input with world model predictions, and to compute the geometrical and dynamical functions necessary to navigate, focus attention, and direct action relative to entities and regions of the world.

Entities

Definition: An entity is an element from the set {point, line, surface, object, group}
The world model contains information about entities stored in lists, or frames. The knowledge database contains a list of all the entities that the intelligent system knows about. A subset of this list is the set of current-entities known to be present in any given situation. A subset of the list of current-entities is the set of entities-of-attention.

There are two types of entities: generic and specific. A generic entity is an example of a class of entities. A generic entity frame contains the attributes of its class. A specific entity is a particular instance of an entity. A specific entity frame inherits the attributes of the class to which it belongs.

An example of an entity frame might be:

ENTITY NAME	--	name of entity
kind	--	class or species of entity
type	--	generic or specific point, line, surface, object, or group
position	--	world map coordinates (uncertainty)
		egosphere coordinates (uncertainty)
dynamics	--	velocity (uncertainty)
		acceleration (uncertainty)
trajectory	--	sequence of positions
geometry	--	center of gravity (uncertainty)
		axis of symmetry (uncertainty)
		size (uncertainty)
		shape
		boundaries (uncertainty)
links	--	subentities
		parent entity
properties	--	physical
		mass
		color
		substance
		behavioral
		social (of animate objects)
capabilities	--	speed, range
value state-variables	--	attract-repulse
		confidence-fear
		love-hate

For example, upon observing a specific cow named Bertha, an entity frame in a farm visitor's brain might have the following values:

ENTITY NAME	-- Bertha
kind	-- cow
type	-- specific object
position	-- x,y,z (in pasture map coordinates)
	-- AZ, EL, R (in egosphere image of observer)
dynamics	-- velocity, acceleration (in egosphere or pasture map coordinates)
trajectory	-- sequence of map positions while grazing
geometry	-- axis of symmetry (right/left)
	size (6x3x10 ft)
	shape (quadruped)
links	-- subentities - surfaces (torso, neck, head, legs, tail, etc.)
	-- parent entity - group (herd)
properties	-- physical
	mass (1050 lbs)
	color (black and white)
	substance (flesh, bone, skin, hair)
	-- behavioral
	(standing, placid, timid, etc.)
capabilities	-- speed, range
value state-variables	-- attract-repulse = 3 (visitor finds cows moderately attractive)
	confidence-fear = -2 (visitor slightly afraid of cows)
	love-hate = 1 (no strong feelings)

Map - Entity Relationship

Map and entity representations are cross referenced and tightly coupled by real-time computing hardware. Each pixel on the map has in its frame a pointer to the list of entities covered by that pixel. For example, each pixel may cover a point entity indicating brightness, color, spatial and temporal gradients of brightness and color, image flow, and range for each point. Each pixel may also cover a linear entity indicating a brightness or depth edge or vertex; a surface entity indicating area, slope, and texture; an object entity indicating the name and attributes of the object covered; a group entity indicating the name and attributes of the group covered, etc.

Likewise, each entity in the attention list may have in its frame a set of geometrical parameters that enables the world model geometry engine to compute the set of egosphere or world map pixels covered by each entity, so that entity parameters associated with each pixel covered can

be overlaid on the world and egosphere maps.

Cross referencing between pixel maps and entity frames allows the results of each level of processing to add map overlays to the egosphere and world map representations. The entity database can be updated from knowledge of image parameters at points on the egosphere, and the map database can be predicted from knowledge of entity parameters in the world model. At each level, local entity and map parameters can be computed in parallel by the type of neurological computing structures known to exist in the brain.

Many of the attributes in an entity frame are time dependent state-variables. Each time dependent state-variable may possess a short-term memory queue that stores a state trajectory, or trace, describing its temporal history. At each hierarchical level, temporal traces stretch backward as far as the planning horizon at that level stretches into the future. At each hierarchical level, the historical trace of an entity state-variable may be captured by summarizing data values at several points in time throughout the historical interval. Time dependent entity state-variable histories may also be captured by running averages and moments, Fourier transform coefficients, Kalman filter parameters, or other analogous methods.

Each state-variable in an entity frame may have value state-variable parameters that indicate levels of believability, confidence, support, or plausibility, and measures of dimensional uncertainty. These are computed by value judgment functions that reside in the VJ modules. (See section 1-7 of this Chapter).

Entity Database Hierarchy

All entities are obtained by clustering of other entities. Each entity consists of a set of subentities, and is part of a parent entity. Thus, all entity databases are hierarchically structured. For example, an object may consist of a set of surfaces, and be part of a group. The definition of an object is quite arbitrary, however, at least from the point of view of the world model. For example, is a nose an object? If so, what is a face? Is a head an object? Or is it part of a group of objects comprising a body? If a body can be a group, what is a group of bodies?

Only in the context of a task, does the definition of an object become clear. For example, in a task frame, an object may be defined either as the agent, or as acted upon by the agent executing the task. Thus, in the context of a specific task, the nose (or face, or head) may become an object because it appears in a task frame as the agent or object of a task.

Perception in an intelligent system is task (or goal) driven, and the structure of the world model entity database is defined by, and may be reconfigured by, the nature of goals and tasks. It is therefore not necessarily the role of the world model to define the boundaries of entities. The boundaries demonstrate the scope of the task, they map regions and entities circumscribed by those boundaries with sufficient resolution to accomplish the task. It is the role of the sensory processing system to identify regions and entities in the external real world that correspond to

those represented in the world model, and to discover boundaries that circumscribe objects defined by tasks.

Theorem: The minimum complexity can be achieved at the value of accuracy assigned If the world model is hierarchically structured with map (iconic) and entity (symbolic) data structures at each level of the hierarchy,.

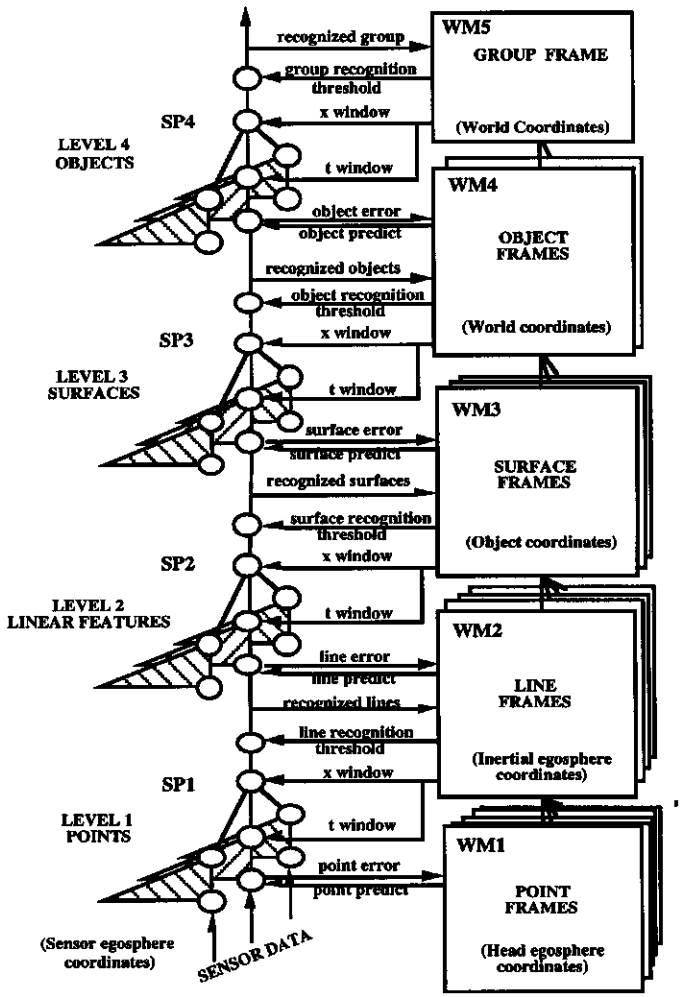


Figure 1-12. The nature of the interactions that take place between the world model and sensory processing modules. At each level, predicted entities are compared with observed. Differences are returned as errors directly to the world model to update the model. Correlations are forwarded upward to be integrated over time and space windows provided by the WM. Correlations that exceed threshold are recognized as entities.

See Figure 1-12. At level 1, the world model can represent map overlays for point entities. In the case of vision, point entities may consist of brightness or color intensities, and spatial and temporal derivatives of those intensities. Point entity frames include brightness spatial and

temporal gradients and range from stereo for each pixel. Point entity frames also include transform parameters to and from head egosphere coordinates. These representations are roughly analogous to Marr's "primal sketch" [60], and are compatible with experimentally observed data representations in the tectum, superior colliculus, and primary visual cortex (V1) [37].

At level 2, the world model can represent map overlays for linear entities consisting of clusters, or strings of point entities. In the visual system, linear entities may consist of connected edges (brightness, color, or depth), vertices, image flow vectors, and trajectories of points in space/time. Attributes such as 3-D position, orientation, velocity, and rotation are represented in a frame for each linear entity. Entity frames include transform parameters to and from inertial egosphere coordinates. These representations are compatible with experimentally observed data representations in the secondary visual cortex (V2) [60].

At level 3, the world model can represent map overlays for surface entities computed from sets of linear entities clustered or swept into bounded surfaces or maps, such as terrain maps, B-spline surfaces, or general functions of two variables. Surface entity frames contain transform parameters to and from object coordinates. In the case of vision, entity attributes may describe surface color, texture, surface position and orientation, velocity, size, rate of growth in size, shape, and surface discontinuities or boundaries. Level 3 is thus roughly analogous to Marr's "2 1/2-D sketch", and is compatible with known representation of data in visual cortical area V3.

At level 4, the world model can represent map overlays for object entities computed from sets of surfaces clustered or swept so as to define 3-D volumes, or objects. Object entity frames contain transform parameters to and from object coordinates. Object entity frames may also represent object type, position, translation, rotation, geometrical dimensions, surface properties, occluding objects, contours, axes of symmetry, volumes, etc. These are analogous to Marr's "3-D model" representation, and compatible with data representations in visual area V4.

At level 5, the world model can represent map overlays for group entities consisting of sets of objects clustered into groups or packs. This is hypothesized to correspond to data representations in visual association areas of parietal and temporal cortex. Group entity frames contain transform parameters to and from world coordinates. Group entity frames may also represent group species, center of mass, density, motion, map position, geometrical dimensions, shape, spatial axes of symmetry, volumes, etc.

At level 6, the world model can represent map overlays for sets of group entities clustered into groups of groups, or group^2 entities. At level 7, the world model can represent map overlays for sets of group^2 entities clustered into group^3 (or world) entities, and so on. At each higher level, world map resolution decreases and range increases by about an order of magnitude per level.

The highest level entity in the world model is the world itself, i.e. the environment as a

whole. The environment entity frame contains attribute state-variables that describe the state of the environment, such as temperature, wind, precipitation, illumination, visibility, the state of hostilities or peace, the current level of danger or security, the disposition of the gods, etc.

Events

Definition: At a particular level of resolution, an event is a state, condition, or situation that exists at a point in time, which at a higher level of resolution correspond to an interval in time which has an initial state, a final state and a particular action is known which started at the initial state and ended at the final state.

Events may be represented in the world model by frames with attributes such as the point, or interval, in time and space when the event occurred, or is expected to occur. Event frame attributes may indicate start and end time, duration, type, relationship to other events, etc.

An example of an event frame is:

EVENT NAME	-- name of event
kind	-- class or species
type	-- generic or specific
modality	-- visual, auditory, tactile, etc.
time	-- when event detected
interval	-- period over which event took place
position	-- map location where event occurred
links	-- subevents
	-- parent event
value	-- good-bad, benefit-cost, etc.

State-variables in the event frame may have confidence levels, degrees of support and plausibility, and measures of dimensional uncertainty similar to those in spatial entity frames. Confidence state-variables may indicate the degree of certainty that an event actually occurred, or was correctly recognized.

Like the entity databases, the event frame databases are hierarchical too. At each level of the sensory processing hierarchy, the recognition of a pattern, or string, of level (i) events makes up a single level(i+1) event.

Hypothesis: The hierarchical levels of the event frame database can be placed in one-to-one correspondence with levels of task decomposition and sensory processing hierarchies.

An event which is represented as a point at the time scale at a particular level of resolution can correspond to different time intervals at the higher resolution. For example, at different levels:

Level 1 -- an event may span a few milliseconds. A typical level(1) acoustic event might be

the recognition of a tone, hiss, click, or a phase comparison indicating the direction of arrival of a sound. A typical visual event might be a change in pixel intensity, or a measurement of brightness gradient at a pixel.

Level 2 -- an event may span a few tenths of a second. A typical level(2) acoustic event might be the recognition of a phoneme or a chord. A visual event might be a measurement of image flow or a trajectory segment of a visual point or feature.

Level 3 -- an event may span a few seconds, and consist of the recognition of a word, a short phrase, or a visual gesture, or motion of a visual surface.

Level 4 -- an event may span a few tens of seconds, and consist of the recognition of a message, a melody, or a visual observation of object motion, or task activity.

Level 5 -- an event may span a few minutes and consist of listening to a conversation, a song, or visual observation of group activity in an extended social exchange.

Level 6 -- an event may span an hour and include many auditory, tactile, and visual observations.

Level 7 -- an event may span a day and include a summary of sensory observations over an entire day's activities.

1.6 Sensory Processing

Definition: Sensory processing in all systems is equivalent to the mechanism of perception in living creatures.

Definition: Perception incorporates decoding and organization of the signals which are acquired from the sensors in such way as to provide for the establishment and maintenance of correspondence between the internal world model and the external real world. Perception includes also the preliminary stage of interpretation: the most general one, it performs the initial out of context labeling.

Definition: The function of sensory processing is to extract information about entities, events, states, and relationships in the external world, so as to keep the world model accurate and up to date.

Measurement of Surfaces

World model maps are updated by sensory measurement of points, edges, and surfaces. Such information is usually derived from vision or touch sensors, although some intelligent systems may derive it from sonar, radar, or laser sensors.

The most direct method of measuring points, edges, and surfaces is through touch. Many creatures, from insects to mammals, have antennae or whiskers that are used to measure the

position of points and orientation of surfaces in the environment. Virtually all creatures have tactile sensors in the skin, particularly in the digits, lips, and tongue. Proprioceptive sensors indicate the position of the feeler or tactile sensor relative to the self when contact is made with an external surface. This, combined with knowledge of the kinematic position of the feeler endpoint, provides the information necessary to compute the position on the egosphere of each point contacted. A series of felt points defines edges and surfaces on the egosphere.

Another primitive measure of surface orientation and depth is available from image flow (i.e. motion of an image on the retina of the eye). Image flow may be caused either by motion of objects in the world, or by motion of the eye through the world. The image flow of stationary objects caused by translation of the eye is inversely proportional to the distance from the eye to the point being observed. Thus, if eye rotation is zero, and the translational velocity of the eye is known, the focus of expansion is fixed, and image flow lines are defined by great circle arcs on the velocity egosphere that emanate from the focus of expansion and pass through the pixel in question [45]. Under these conditions, range to any stationary point in the world can be computed directly from image flow by the simple formula

$$R = \frac{v \sin A}{dA/dt} \tag{1.8.1}$$

where R is the range to the point
v is translational velocity vector of the eye
A is the angle between the velocity vector and the pixel covering the point
dA/dt is the image flow rate at the pixel covering the point

When eye rotation is zero and **v** is known, the flow rate dA/dt can be computed locally for each pixel from temporal and spatial derivatives of image brightness along flow lines on the velocity egosphere. dA/dt can also be computed from temporal cross-correlation of brightness from adjacent pixels along flow lines.

When the eye fixates on a point, dA/dt is equal to the rotation rate of the eye. Under this condition, the distance to the fixation point can be computed from (1.8.1), and the distance to other points may be computed from image flow relative to the fixation point.

If eye rotation is non-zero but known, the range to any stationary point in the world may be computed by a closed form formula of the form

$$R = F(x, y, \mathbf{T}, \mathbf{W}, \frac{\partial I}{\partial t}, \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}) \tag{7.2}$$

where x and z are the image coordinates of a pixel
T is the translational velocity vector of the camera in camera coordinates

W is the rotational velocity vector of the camera in camera coordinates

I is the pixel brightness intensity

This type of function can be implemented locally and in parallel by a neural net for each image pixel [52].

Knowledge of eye velocity, both translational and rotational, may be computed by the vestibular system, the locomotion system, and/or high levels of the vision system. Knowledge of, rotational eye motion may either be used in the computation of range by (1.8.2), or can be used to transform sensor egosphere images into velocity egosphere coordinates where (1.8.1) applies. This can be accomplished mechanically by the vestibulo-ocular reflex, or electronically (or neuronally) by scrolling the input image through an angle determined by a function of data variables from the vestibular system and the ocular muscle stretch receptors. Virtual transformation of image coordinates can be accomplished using coordinate transform parameters located in each map pixel frame.

Depth from image flow enables living creatures, from fish and insects to birds and mammals, to maneuver rapidly through natural environments filled with complex obstacles without collision. Moving objects can be segmented from stationary objects by their failure to match world model predictions for stationary objects. Near objects can be segmented from distant objects by their differential flow rates.

Distance to surfaces may also be computed from stereovision. The angular disparity between images in two eyes separated by a known distance can be used to compute range. Depth from stereo is more complex than depth from image flow in that it requires identification of corresponding points in images from different eyes. Hence it cannot be computed locally. However, stereo is simpler than image flow in that it does not require eye translation and is not confounded by eye rotation or by moving objects in the world. The computation of distance from a combination of both motion and stereo is more robust, and hence psychophysically more vivid to the observer, than from either motion or stereo alone.

Distance to surfaces may also be computed from sonar or radar by measuring the time delay between emitting radiation and receiving an echo. Difficulties arise from poor angular resolution and from a variety of sensitivity, scattering, and multipath problems. Creatures such as bats and marine mammals use multispectral signals such as chirps and clicks to minimize confusion from these effects. Phased arrays and synthetic apertures may also be used to improve the resolution of radar or sonar systems.

All of the above methods for perceiving surfaces are primitive in the sense that they compute depth directly from sensory input without recognizing entities or understanding anything about the scene. Depth measurements from primitive processes can immediately generate maps that can be used directly by the lower levels of the behavior generation hierarchy to avoid obstacles and approach surfaces.

Surface attributes such as position and orientation may also be computed from shading, shadows, and texture gradients. These methods typically depend on higher levels of visual perception such as geometric reasoning, recognition of objects, detection of events and states, and the understanding of scenes.

Recognition and Detection

Definition: Recognition is establishing a one-to-one match or correspondence between a real world entity and a world model entity .

The process of recognition may proceed top-down, or bottom-up, or both simultaneously. For each entity in the world model, there exists a frame filled with information that can be used to predict attributes of corresponding entities observed in the world. The top-down process of recognition begins by hypothesizing a world model entity and comparing its predicted attributes with those of the observed entity. When the similarities and differences between predictions from the world model and observations from sensory processing are integrated over a space-time window that covers an entity, a matching, or cross-correlation value is computed between the entity and the model. If the correlation value rises above a selected threshold, the entity is said to be recognized. If not, the hypothesized entity is rejected and another tried.

The bottom-up process of recognition consists of applying filters and masks to incoming sensory data and computing image properties and attributes. These may then be stored in the world model, or compared with the properties and attributes of entities already in the world model. Both top-down and bottom-up processes proceed until a match is found, or the list of world model entities is exhausted. Many perceptual matching processes may operate in parallel at multiple hierarchical levels simultaneously.

If a SP module recognizes a specific entity, the WM at that level updates the attributes in the frame of that specific WM entity with information from the sensory system. If the SP module fails to recognize a specific entity, but instead achieves a match between the sensory input and a generic world model entity, a new specific WM entity will be created with a frame that initially inherits the features of the generic entity. Slots in the specific entity frame can then be updated with information from the sensory input. If the SP module fails to recognize either a specific or a generic entity, the WM may create an "unidentified" entity with an empty frame. This may then be filled with information gathered from the sensory input.

When an unidentified entity occurs in the world model, the behavior generation system may (depending on other priorities) select a new goal to <identify the unidentified entity>. This may initiate an exploration task that positions and focuses the sensor systems on the unidentified entity, and possibly even probes and manipulates it, until a world model frame is constructed that adequately describes the entity. The sophistication and complexity of the exploration task depends on task knowledge about exploring things. Such knowledge may be very advanced and include

sophisticated tools and procedures, or very primitive. Entities may, of course, simply remain labeled as "unidentified" or "unexplained."

Definition: Event detection is analogous to entity recognition. Observed states of the real world are compared with states predicted by the world model. Similarities and differences are integrated over an event space-time window, and a matching, or cross- correlation value is computed between the observed event and the model event. When the cross-correlation value rises above a given threshold, the event is detected.

The Context of Perception

At every hierarchical level of the world model, there exists a short term memory in which a temporal history is stored consisting of strings of past scenes, states, values of time dependent entities and event attributes. Thus, it can be assumed that at any point in time, an intelligent system has a record in its short term memory of how it reached its current state. See Figure 1-5. Figures 1-5 and1-6 also imply that, for every planner in each behavior generating BG module at each level, there exists a plan, and that each executor is currently executing the first step in its respective plan. Finally, it can be assumed that the knowledge in all these plans and temporal histories, and all the task, entity, and event frames referenced by them, is available in the world model.

Thus it can be assumed that an intelligent system almost always knows where it is on a world map, knows how it got there, where it is going, what it is doing, and has a current list of entities of attention, each of which has a frame of attributes (or state variables). They describe the recent past, and provide a basis for predicting future states. This includes a prediction of what objects will be visible, where and how object surfaces will appear, and which surface boundaries, vertices, and points will be observed in the image produced by the sensor system. It also means that the position and motion of the eyes, ears, and tactile sensors relative to surfaces and objects in the world are known, and this knowledge is available to be used by the sensory processing system for constructing maps and overlays, recognizing entities, and detecting events.

Were the above not the case, the intelligent system would exist in a situation analogous to a person who suddenly awakens at an unknown point in space and time. In such cases, it typically is necessary even for humans to perform a series of tasks designed to "regain their bearings", i.e. to bring their world model into correspondence with the state of the external world and to initialize plans, entity frames, and system state variables.

It is possible for an intelligent creature to function for short periods in a totally unknown environment. They don't function well, because every intelligent creature uses historical information that forms the context of its current task. Without information about its environment, even the most intelligent creature is handicapped, but not too long. Because the sensory processing system continuously updates the world model with new information about the current situation and

its recent historical development, so that, within a few seconds, a functionally adequate map and a usable set of entity state variables can usually be acquired from the immediately surrounding environment.

Sensory Processing SP Modules

Filtering and Scaling are the major procedures which precede the process of information organization performed by SP. After these are completed, the core procedures of organization start that are based upon *comparison*. At each level of the intelligent system architecture, there are a number of computational nodes. Each of these contains an SP module, and each SP module consists of four sublevels, as shown in Figure 1-13 as Sublevel 1 (Comparison.) Each comparison submodule matches an observed sensory variable with a world model prediction of that variable. This comparison typically involves an arithmetic operation, such as multiplication or subtraction, which yields a measure of similarity and difference between an observed variable and a predicted variable. Similarities indicate the degree to which the WM predictions are correct, and hence are a measure of the correspondence between the world model and reality. Differences indicate a lack of correspondence between world model predictions and sensory observations. Differences imply that either the sensor data or world model is incorrect. Difference images from the comparator go three places:

a) They are returned directly to the WM for real-time local pixel attribute updates. This produces a tight feedback loop whereby the world model predicted image becomes an array of Kalman filter state-estimators. Difference images are thus error signals by which each pixel of the predicted image can be "trained" to correspond to current sensory input.
b) They are also transmitted upward to the integration sublevels where they are integrated over time and space in order to recognize and detect global entity attributes. This integration constitutes a summation, or "chunking", of sensory data into entities. At each level, lower order entities are "chunked" into higher order entities, i.e. points are chunked into lines, lines into surfaces, surfaces into objects, objects into groups, etc.
c) They are transmitted to the VJ module at the same level where statistical parameters are computed in order to assign confidence and believability factors to pixel entity attribute estimates.

Sublevel 2 -- Temporal integration

Temporal integration submodules integrate similarities and differences between predictions and observations over time intervals. Temporal integration submodules operating just on sensory data can produce a summary, such as a total, or average, of sensory information over a given time window. Temporal integrator submodules operating on the similarity and difference values computed by comparison submodules may produce temporal cross-correlation and covariance

functions between the model and the observed data. These correlation and covariance functions are measures of how well the dynamic properties of the world model entity match those of the real world entity. The boundaries of the temporal integration window may be derived from world model prediction of event durations, or form behavior generation parameters such as sensor fixation periods.

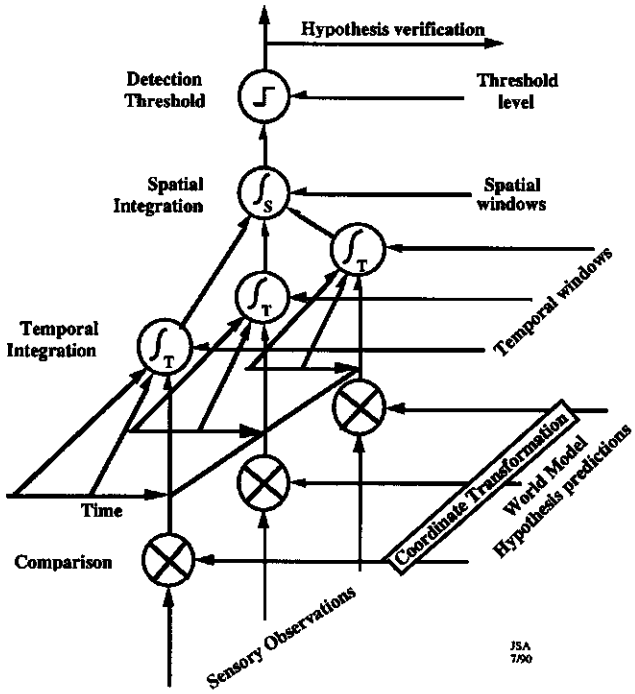


Figure 1-13. Each sensory processing SP module consists of: 1) a set of comparators that compare sensory observations with world model predictions, 2) a set of temporal integrators that integrate similarities and differences, 3) a set of spatial integrators that fuse information from different sensory data stream, and 4) a set of threshold detectors that recognize entities and detect events.

Sublevel 3 -- Spatial integration

Spatial integrator submodules integrate similarities and differences between predictions and observations over regions of space. This produces spatial cross-correlation or convolution functions between the model and the observed data. Spatial integration summarizes sensory information from multiple sources at a single point in time. It determines whether the geometric properties of a world model entity match those of a real world entity. For example, the product of an edge operator and an input image may be integrated over the area of the operator to obtain the correlation between the image and the edge operator at a point. The limits of the spatial integration window may be determined by world model predictions of entity size. In some cases, the order of temporal and spatial integration may be reversed or interleaved.

Sublevel 4 -- Recognition/Detection

When the spatio-temporal correlation function exceeds some threshold, object recognition (or event detection) occurs. For example, if the spatio-temporal summation over the area of an edge operator exceeds threshold, an edge is said to be detected at the center of the area.

Figure 1-14 illustrates the nature of the SP-WM interactions between an intelligent vision system and the world model at one level. On the left of Figure 1-14, the world of reality is viewed through the window of an egosphere such as exists in the primary visual cortex. On the right is a world model consisting of: 1) a symbolic entity frame in which entity attributes are stored, and 2) an iconic predicted image that is registered in real-time with the observed sensory image. In the center of Figure 1-14, is a comparator where the expected image is subtracted from (or otherwise compared with) the observed image.

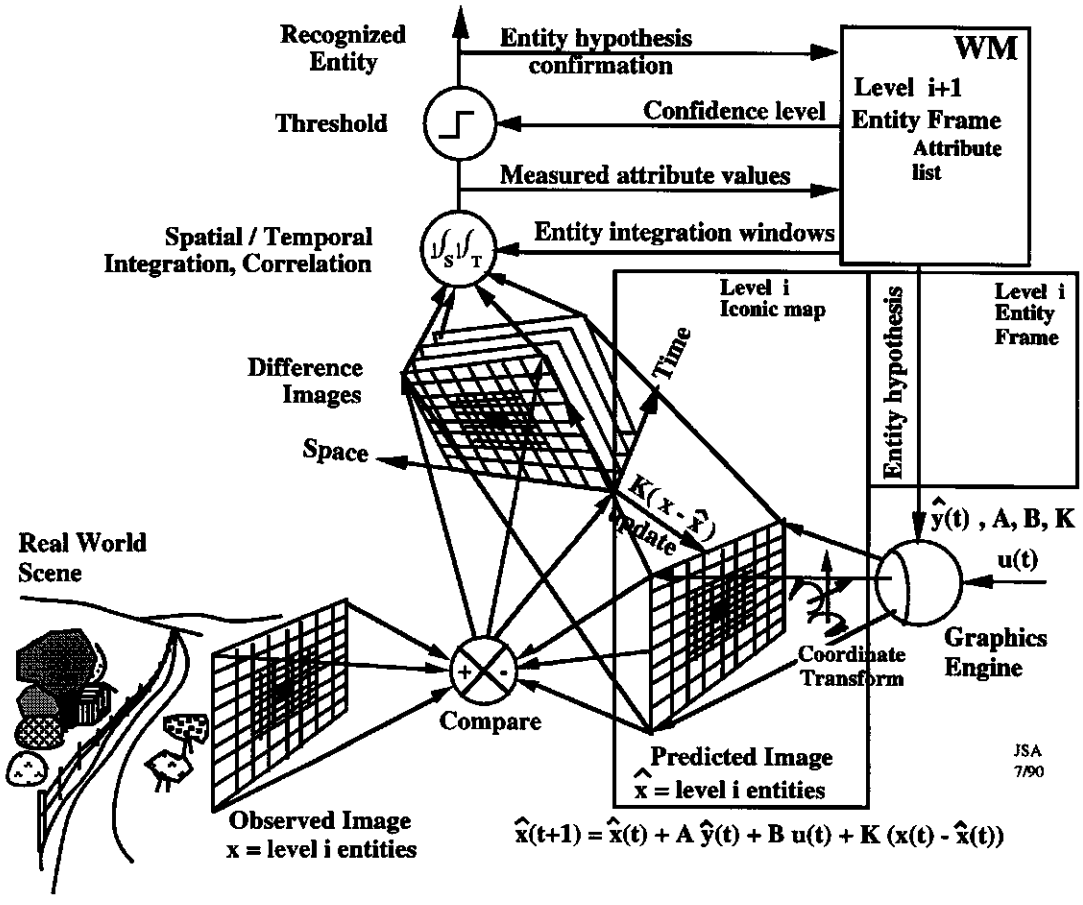


Figure 1-14. Interaction between world model and sensory processing. Difference images are generated by comparing predicted images with observed images. Feedback of differences produces a Kalman best estimate for each data variable in the world model. Spatial and temporal integration produce cross-correlation functions between the estimated attributes in the world model and the real world attributes measured in the observed image. When the correlation exceeds threshold, entity recognition occurs.

The level(i) predicted image is initialized by the equivalent of a graphics engine operating on symbolic data from frames of entities hypothesized at level (i+1). The predicted image is updated by differences between itself and the observed sensory input. By this process, the predicted image becomes the world model's "best estimated prediction" of the incoming sensory image. A high speed loop is closed between the WM and SP modules at level(i).

When recognition occurs in level (i), the world model level (i+1) hypothesis is confirmed and both level(i) and level (i+1) symbolic parameters that produced the match are updated in the symbolic database. This closes a slower, more global, loop between WM and SP modules through the symbolic entity frames of the world model. Many examples of this type of looping interaction can be found in the model matching and model-based recognition literature [47]. Similar closed loop filtering concepts have been used for years for signal detection and for dynamic systems modeling in aircraft flight control systems. Recently, they have been applied to high-speed visually guided driving of an autonomous ground vehicle [54].

The behavioral performance of intelligent biological creatures suggests that mechanisms similar to those shown in Figures 1-12 and 1-13 exist in the brain. In biological or neural network implementations, SP modules may contain thousands, even millions, of comparison submodules, temporal and spatial integrators, and threshold submodules. The neuroanatomy of the mammalian visual system suggests how maps with many different overlays, as well as lists of symbolic attributes, could be processed in parallel in real-time. In such structures, it is possible for multiple world model hypotheses to be compared with sensory observations at multiple hierarchical levels, all simultaneously.

World Model Update

Attributes in the world model predicted image may be updated by a formula of the form

(1.8.3) $$\hat{\mathbf{x}}(t+1) = \hat{\mathbf{x}}(t) + \mathbf{A}\,\hat{\mathbf{y}}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{K}(t)\,[\mathbf{x}(t) - \hat{\mathbf{x}}(t)]$$

where $\hat{\mathbf{x}}(t)$ is the best estimate vector for the variable $\mathbf{x}(t)$ of world model i-order entity attributes at time t

$\mathbf{A}$ is a matrix that computes the expected rate of change of $\hat{\mathbf{x}}(t)$ given the current best estimate of the i+1 order entity attribute vector $\hat{\mathbf{y}}(t)$

$\mathbf{B}$ is a matrix that computes the expected rate of change of $\hat{\mathbf{x}}(t)$ due to external input $\mathbf{u}(t)$

$\mathbf{K}(t)$ is a confidence factor vector for updating $\hat{\mathbf{x}}(t)$

The value of $\mathbf{K}(t)$ may be computed by a formula of the form

(1.8.4) $$\mathbf{K}(t) = K_s(j,t)\,[1 - K_m(j,t)]$$

where $K_s(j,t)$ is the confidence in the sensory observation of the j-th real world attribute $x(j,t)$ at time t, $0 \leq K_s(j,t) \leq 1$

$K_m(j,t)$ is the confidence in the world model prediction of the j-th attribute $\hat{x}(j,t)$ at time t , $0 = \leq K_m(j,t) \leq 1$

The confidence factors (K_m and K_s) in formula (7.4) may depend on the statistics of the correspondence between the world model entity and the real world entity (e.g. the number of data samples, the mean and variance of $[\mathbf{x}(t) - \hat{\mathbf{x}}(t)]$, etc.). A high degree of correlation between $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ in both temporal and spatial domains indicates that entities or events have been correctly recognized, and states and attributes of entities and events in the world model correspond to those in the real world environment. World model data elements that match observed sensory data elements are reinforced by increasing the confidence, or believability factor, $K_m(j,t)$ for the entity or state at location j in the world model attribute lists. World model entities and states that fail to match sensory observations have their confidence factors $K_m(j,t)$ reduced. The confidence factor $K_s(j,t)$ may be derived from the signal-to-noise ratio of the j-th sensory data stream.

The numerical value of the confidence factors may be computed by a variety of statistical methods such Baysian or Dempster-Shafer statistics.

The Mechanisms of Attention

Assertion: Sensory processing is an active process that is directed by goals and priorities generated in the behavior generating system.

In each node of the intelligent system hierarchy, the behavior generating BG modules request information needed for the current task from sensory processing SP modules. By means of such requests, the BG modules control the processing of sensory information and focus the attention of the WM and SP modules on the entities and regions of space that are important to success in achieving behavioral goals. Requests by BG modules for specific types of information cause SP modules to select particular sensory processing masks and filters to apply to the incoming sensory data. Requests from BG modules enable the WM to select which world model data to use for predictions and which prediction algorithm to apply to the world model data. BG requests also define which correlation and difference operators to use and which spatial and temporal integration windows and detection thresholds to apply.

Behavior generating BG modules in the attention subsystem also actively point the eyes and ears, and direct the tactile sensors of antennae, fingers, tongue, lips, and teeth toward objects of attention. BG modules in the vision subsystem control the motion of the eyes, adjust the iris and focus, and actively point the fovea to probe the environment for the visual information needed to pursue behavioral goals [55, 56]. Similarly, BG modules in the auditory subsystem actively direct the ears and tune audio filters to mask background noises and discriminate in favor of the acoustic signals of importance to behavioral goals.

Because of the active nature of the attention subsystem, sensor resolution and sensitivity is not uniformly distributed, but highly focused. For example, receptive fields of optic nerve fibers from the eye are several thousand times more densely packed in the fovea than near the periphery of the visual field. Receptive fields of touch sensors are also several thousand times more densely packed in the finger tips and on the lips and tongue than on other parts of the body e.g. the torso.

The active control of sensors with non-uniform resolution has profound impact on the communication bandwidth, computing power, and memory capacity required by the sensory processing system. For example, there are roughly 500,000 fibers in the the optic nerve from a single human eye. These fibers are distributed such that about 100,000 are concentrated in the ±1.0 degree foveal region with resolution of about 0.007 degrees. About 100,000 cover the surrounding ± 3 degree region with resolution of about 0.02 degrees. 100,000 more cover the surrounding ± 10 degree region with resolution of 0.07 degrees. 100,000 more cover the surrounding ± 30 degree region with a resolution of about 0.2 degrees. 100,000 more cover the remaining ± 80 degree region with resolution of about 0.7 degree [57]. The total number of pixels is thus about 500,000 pixels, or somewhat less than that contained in two standard commercial TV images. Without non-uniform resolution, covering the entire visual field with the resolution of the fovea would require the number of pixels in about 6000 standard TV images. Thus, for a vision sensory processing system with any given computing capacity, active control and non-uniform resolution in the retina can produce more than three orders of magnitude improvement in image processing capability.

SP modules in the attention subsystem process data from lower-resolution wide-angle sensors to detect regions of interest, such as entities that move or regions with discontinuities (edges and lines), or have high curvature (corners and intersections). The attention BG modules then actively maneuver the eyes, fingers, and mouth so as to bring the higher resolution portions of the sensory systems to bear precisely on these points of attention. The result gives the subjective effect of high resolution everywhere in the sensory field. For example, wherever the eye looks, it sees with high resolution, for the fovea is always centered on the item of current interest.

The act of perception involves both sequential and parallel operations. For example, the fovea of the eye is typically scanned sequentially over points of attention in the visual field [58].

Touch sensors in the fingers are actively scanned over surfaces of objects, and the ears may be pointed toward sources of sound. While this sequential scanning is going on, parallel recognition processes hypothesize and compare entities at all levels simultaneously.

The Sensory Processing Hierarchy

It has long been recognized that sensory processing occurs in a hierarchy of processing modules, and that perception proceeds by "chunking", i.e. by recognizing patterns, groups, strings, or clusters of points at one level as a single feature, or point in a higher level, more abstract space. It also has been observed that this chunking process proceeds by about an order of magnitude per level, both spatially and temporally [17,18]. Thus, at each level in the proposed architecture, SP modules integrate, or chunk, information over space and time by about an order of magnitude.

Figure 1-15 describes the nature of the interactions hypothesized to take place between the sensory processing and world modeling modules at the first four levels as the recognition process proceeds. The functional properties of the SP modules are coupled to and determined by the predictions of the WM modules in their respective processing nodes. The WM predictions are, in turn, effected by states of the BG modules.

Hypothesis: There exist both iconic (maps) and symbolic (entity frames) at all levels of the SP/WM hierarchy of the mammalian vision system.

Figure 1-14 illustrates the concept stated in this hypothesis. Visual input to the retina consists of photometric brightness and color intensities measured by rods and cones. Brightness intensities are denoted by I(k, AZ, EL, t), where I is the brightness intensity measured at time t by the pixel at sensor egosphere azimuth AZ and elevation EL of eye (or camera) k. Retinal intensity signals I may vary over time intervals on the order of a millisecond or less.

Image preprocessing is performed on the retina by horizontal, bipolar, amacrine, and ganglion cells. Center-surround receptive fields ("on-center" and "off-center") detect both spatial and temporal derivatives at each point in the visual field. Outputs from the retina carried by ganglion cell axons become input to sensory processing level 1 as shown in Figure 1-15. Level 1 inputs correspond to events of a few milliseconds duration.

It is hypothesized that in the mammalian brain, the level 1 vision processing module consists of the neurons in the lateral geniculate bodies, the superior colliculus, and the primary visual cortex (V1). Optic nerve inputs from the two eyes are overlaid such that the visual fields from left and right eyes are in registration. Data from stretch sensors in the ocular muscles provide information to the superior colliculus about eye convergence, and pan, tilt, and roll of the retina relative to the head. This allows image map points in retinal coordinates to be transformed into image map points in head coordinates (or vice versa) so that visual and acoustic position data can be registered and fused [47, 48]. In V1, registration of corresponding pixels from two separate

eyes on single neurons provides the basis for computing range from stereo for each pixel [36].

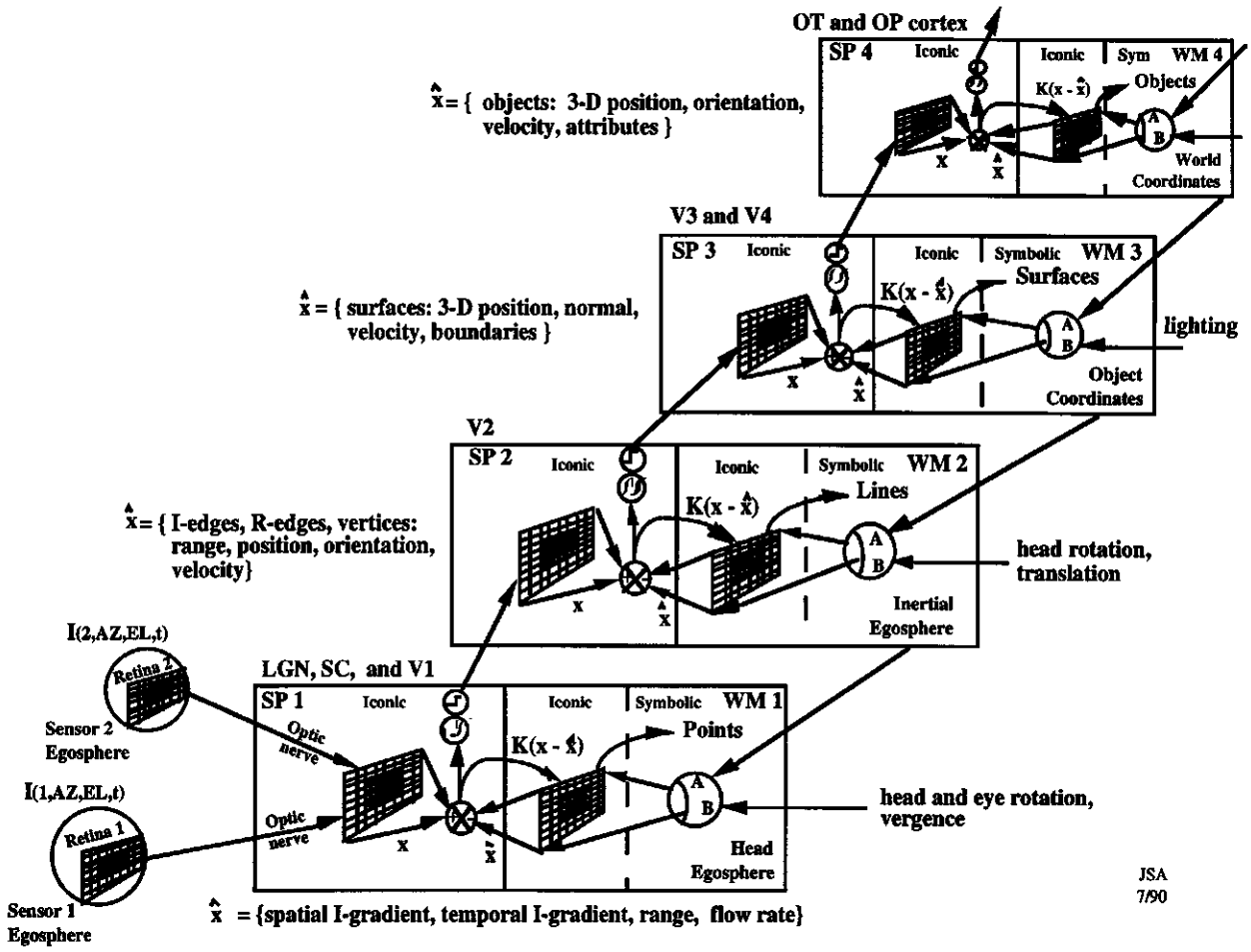


Figure 1-15. Hypothesized correspondence between levels in the proposed model and neuroanatomical structures in the mammalian vision system. At each level, the WM module contains both iconic and symbolic representations. At each level, the SP module compares the observed image with a predicted image. At each level, both iconic and symbolic world models are updated, and map overlays are computed. LGN=lateral geniculate nuclei, OT=occipital-temporal, OP=occipital-parietal, SC=superior colliculus.

At level 1, observed point entities are compared with predicted point entities. Similarities and differences are integrated into linear entities. Strings of level 1 input events are integrated into level 1 output events spanning a few tens of milliseconds. Level 1 outputs become level 2 inputs.

The level 2 vision processing module is hypothesized to consist of neurons in the secondary visual cortex (V2). Some individual neurons indicate edges and lines at particular orientations. Other neurons indicate edge points, curves, trajectories, vertices, and boundaries. At level 2, observed linear entities are compared with predicted linear entities. Similarities and differences are integrated into surface entities.

Input to the world model from the vestibular system indicates the direction of gravity and the rotation of the head. This allows the level 2 world model to transform head egosphere

representations into inertial egosphere coordinates where the world is perceived to be stationary despite rotation of the sensors.

Acceleration data from the vestibular system, combined with velocity data from the locomotion system, provide the basis for estimating both rotary and linear eye velocity, and hence image flow direction. This allows the level 2 world model to transform head egosphere representations into velocity egosphere coordinates where depth from image flow can be computed. Center-surround receptive fields along image flow lines can be subtracted from each other to derive spatial derivatives in the flow direction. At each point where the spatial derivative in the flow direction is non-zero, spatial and temporal derivatives can be combined with knowledge of eye velocity to compute the image flow rate dA/dt [45]. Range to each pixel can then be computed directly, and in parallel, from local image data using formula (7.1) or (7.2).

The above egosphere transformations do not necessarily imply that neurons are physically arranged in inertial or velocity egosphere coordinates on the visual cortex. If that were true, it would require that the retinal image be scrolled over the cortex, and there is little evidence for this, at least in V1 and V2. Instead, it is conjectured that the neurons that make up both observed and predicted iconic images exist on the visual cortex in retinotopic, or sensor egosphere, coordinates. The velocity and inertial egosphere coordinates for each pixel are defined by parameters in the symbolic entity frame of each pixel. The inertial, velocity (and perhaps head) egospheres may thus be "virtual" egospheres. The position of any pixel on any egosphere can be computed by using the transformation parameters in the map pixel frame as an indirect address off-set. This allows velocity and inertial egosphere computations to be performed on neural patterns that are physically represented in sensor egosphere coordinates. The possibility of image scrolling cannot be ruled out, however, particularly at higher levels. It has been observed that both spatial and temporal retinotopic specificity decreases about two orders of magnitude from V1 to V4 [54]. This is consistent with scrolling.

Strings of level 2 input events are integrated into level 3 input events spanning a few hundreds of milliseconds.

The level 3 vision processing module is hypothesized to reside in area V3 of the visual cortex. Observed surface entities are compared with predicted surface entities. Similarities and differences are integrated to recognize object entities. Cells that detect texture and motion of regions in specific directions provide indication of surface boundaries and depth discontinuities. Correlations and differences between world model predictions and sensory observations of surfaces give rise to meaningful image segmentation and recognition of surfaces. World model knowledge of lighting and texture allow computation of surface orientation, discontinuities, boundaries, and physical properties.

Strings of level 3 input events are integrated into level 4 input events spanning a few seconds. (This does not necessarily imply that it takes seconds to recognize objects, but that both

patterns of motion that occupy a few seconds, and objects, are recognized at level 3. For example, the recognition of a gesture, or dance step, might occur at this level.)

World model knowledge of the position of the self relative to object surfaces enables level 3 to compute off-set variables for each pixel that transform it from inertial egosphere coordinates into object coordinates.

The level 4 vision processing module is hypothesized to reside in area V4 of visual cortex. At level 4, observed objects are compared with predicted objects. Correlations and differences between world model predictions and sensory observations allow shape, size, and orientation, as well as location, velocity, rotation, and size-changes of objects to be recognized and measured.

World model input from the locomotion and navigation systems allow level 4 to transform object coordinates into world coordinates. Objects are clustered into groups, and strings of level 4 input events are grouped into level 5 input events spanning a few tens of seconds.

Level 5 vision is hypothesized to reside in the visual association areas of the parietal and temporal cortex. At level 5, observed groups of objects are compared with predicted groups. Correlations and differences are integrated to recognize group properties. For example, in the anterior inferior temporal region particular groupings of objects such as eyes, nose, and mouth are recognized as faces. Groups of fingers can be recognized as hands, etc. In the parietal association areas, map positions, orientations, rotations of groups of objects are detected. At level 5, the world model map has larger span and lower resolution than level 4. Strings of level 5 input events are detected as level 5 output events spanning a few minutes. Clusters of groups are recognized as group^2 entities.

At level 6, observed group^2 entities are compared with predicted group^2 entities, and clusters of group^2 entities are recognized as group^3 entities. Strings of level 6 input events are grouped into level 6 output events spanning a few tens of minutes. The world model map at level 6 has larger span and lower resolution than at level 5. At level 7, strings of input events are grouped into level 7 output events spanning a few hours.

Note that the neuroanatomy of the mammalian vision system is much more convoluted than suggested by Figure 1-15. Van Essen [59] has compiled a list of 84 identified or suspected pathways connecting 19 visual areas. Visual processing is accomplished in at least two separate subsystems that are not differentiated in Figure 1-13. The subsystem that includes the temporal cortex emphasizes the recognition of entities and their attributes such as shape, color, orientation, and grouping of features. The subsystem that includes the parietal cortex emphasizes spatial and temporal relationships such as map positions, timing of events, velocity, and direction of motion [54]. Similar figures could be drawn for other sensory modalities such as hearing and touch.

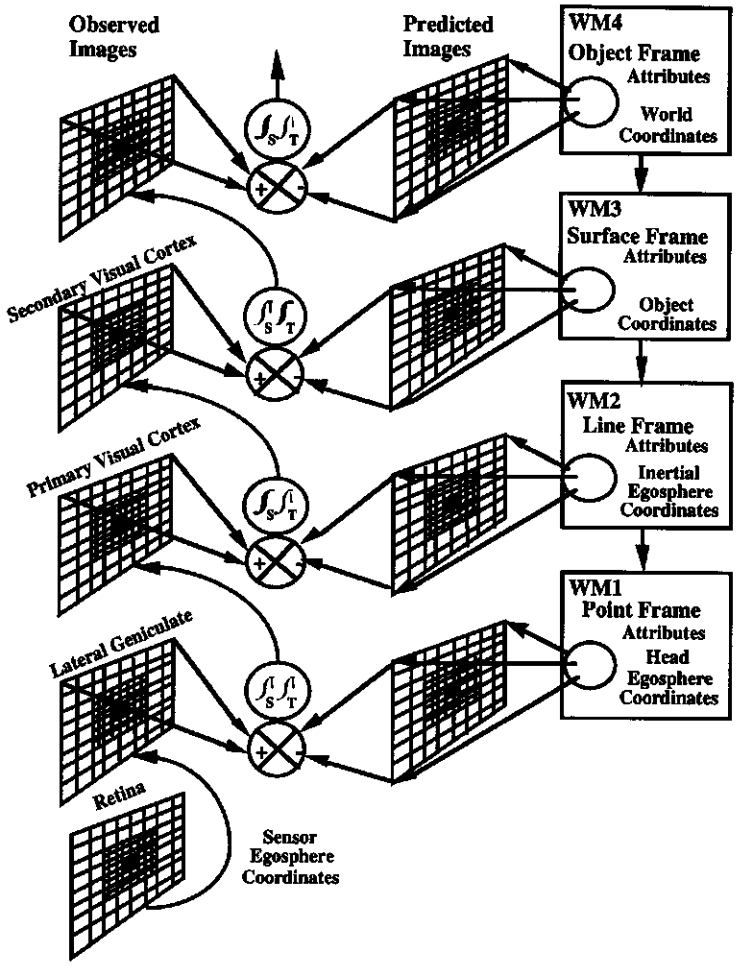


Figure 1-16. Correspondence between levels in the proposed model and neuranatomical structures in the human vision system. At each level, the observed image is compared with a predicted image. At each level, additional map overlays are computed in the coordinate system of that level. In the first four levels, there is no significant reduction in resolution, just increased aggregation of pixels into lower resolution entities.

Gestalt effects

When an observed entity is recognized at a particular hierarchical level, its entry into the world model provides predictive support to the level below. The recognition output also flows upward where it narrows the search at the level above. For example, a linear feature recognized and entered into the world model at level 2, can be used to generate expected points at level 1. It can also be used to prune the search tree at level 3 to entities that contain that particular type of linear feature. Similarly, surface features recognized at level 3 can generate specific expected linear features at level 2, and limit the search at level 4 to objects that contain such surfaces, etc. The recognition of an entity at any level thus provides to both lower and higher levels information that is useful in selecting processing algorithms and setting spatial and temporal integration windows to integrate lower level features into higher level chunks.

If the correlation function at any level falls below threshold, the current world model entity or event at that level will be rejected and others tried. When an entity or event is rejected, the rejection also propagates both upward and downward, broadening the search space at both higher and lower levels.

At each level, the SP and WM modules are coupled to form a feedback loop that has the properties of a relaxation process, or phase-lock loop. WM predictions are compared with SP observations, and the correlations and differences are fed back to modify subsequent WM predictions. WM predictions can be "servoed" into correspondence with the SP observations. Such looping interactions will either converge to a tight correspondence between predictions and observations, or will diverge to produce a definitive set of irreconcilable differences.

Perception is complete only when the correlation functions at all levels exceed threshold simultaneously. It is the nature of closed loop processes for lock-on to occur with a positive "snap". This is especially pronounced in systems with many coupled loops that lock on in quick succession. The result is a gestalt "aha" effect that is characteristic of many human perceptions.

Flywheeling, Hysteresis, and Illusion

Once recognition occurs, the looping process between SP and WM acts as a tracking filter. This enables world model predictions to track real world entities through noise, data dropouts, and occlusions.

In the system described above, recognition will occur when the first hypothesized entity exceeds threshold. Once recognition occurs, the search process is suppressed, and the thresholds for all competing recognition hypotheses are effectively raised. This creates a hysteresis effect that tends to keep the WM predictions locked onto sensory input during the tracking mode. It may also produce undesirable side effects, such as a tendency to perceive only what is expected, and a tendency to ignore what does not fit preconceived models of the world.

In cases where sensory data is ambiguous, there is more than one model that can match a particular observed object. The first model that matches will be recognized, and other models will be suppressed. This explains the effects produced by ambiguous figures such as the Necker cube.

Once an entity has been recognized, the world model projects its predicted appearance so that it can be compared with the sensory input. If this predicted information is added to sensory input (or multiplied by a positive bias), perception at higher levels will be based on a mix of sensory observations and world model predictions. By this mechanism, the world model may fill in sensory data that is missing, and provide information that may be left out of the sensory data. For example, it is well known that the audio system routinely "flywheels" through interruptions in speech data, and fills-in over noise bursts.

This merging of world model predictions with sensory observations may account for many

familiar optical illusions such as subjective contours and the Ponzo illusion. In pathological cases, it may also account for visions and voices, and an inability to distinguish between reality and imagination.

Merging of world model prediction with sensory observation is what Grossberg calls "adaption resonance" [61].

1.7 Value Judgments

Value judgments provide the criteria for making intelligent choices. Value judgments evaluate the costs, risks, and benefits of plans and actions, and the desirability, attractiveness, and uncertainty of objects and events. Value judgment modules produce evaluations that can be represented as value state-variables. These can be assigned to the attribute lists in entity frames of objects, persons, events, situations, and regions of space. They can also be assigned to the attribute lists of plans and actions in task frames. Value state-variables can label entities, tasks, and plans as good or bad, costly or inexpensive, as important or trivial, as attractive or repulsive, as reliable or uncertain. Value state-variables can also be used by the behavior generation modules both for planning and executing actions. Value judgments provide the criteria for decisions about which coarse of action to take [62].

Value state-variable parameters may be overlaid on the map and egosphere regions where the entities to which they are assigned appear. This facilitates planning. For example, approach-avoidance behavior can be planned on an egosphere map overlay defined by the summation of attractor and repulsor value state-variables assigned to objects or regions that appear on the egosphere. Navigation planning can be done on a map overlay whereon risk and benefit values are assigned to regions on the egosphere or world map.

Definition: Emotions are biological value state-variables that provide estimates of good and bad.

Emotion value state-variables can be assigned to the attribute lists of entities, events, tasks, and regions of space so as to label these as good or bad, as attractive or repulsive, etc. Emotion value state-variables provide criteria for making decisions about how to behave in a variety of situations. For example, objects or regions labeled with fear can be avoided, objects labeled with love can be pursued and protected, those labeled with hate can be attacked, etc. Emotional value judgments can also label tasks as costly or inexpensive, risky or safe.

Definition: Priorities are value state-variables that provide estimates of importance.

Priorities can be assigned to task frames so that BG planners and executors can decide what to do first, how much effort to spend, how much risk is prudent, and how much cost is acceptable, for each task.

Definition: Drives are value state-variables that provide estimates of need.

Drives can be assigned to the self frame, to indicate internal system needs and

requirements. In biological systems, drives indicate levels of hunger, thirst, and sexual arousal. In mechanical systems, drives might indicate how much fuel is left, how much pressure is in a boiler, how many expendables have been consumed, or how much battery charge is remaining.

The Limbic System

In animal brains, value judgment functions are computed by the limbic system. Value state-variables produced by the limbic system include emotions, drives, and priorities. In animals and humans, electrical or chemical stimulation of specific limbic regions (i.e. value judgment modules) has been shown to produce pleasure and pain as well as more complex emotional feelings such as fear, anger, joy, contentment, and despair. Fear is computed in the posterior hypothalamus. Anger and rage are computed in the amygdala. The insula computes feelings of contentment, and the septal regions produce joy and elation. The perifornical nucleus of the hypothalamus computes punishing pain, the septum pleasure, and the pituitary computes the body's priority level of arousal in response to danger and stress [63].

The drives of hunger and thirst are computed in the limbic system's medial and lateral hypothalamus. The level of sexual arousal is computed by the anterior hypothalamus. The control of body rhythms, such as sleep-awake cycles, are computed by the pineal gland. The hippocampus produces signals that indicate what is important and should be remembered, or what is unimportant and can safely be forgotten. Signals from the hippocampus consolidate (i.e. make permanent) the storage of sensory experiences in long term memory. Destruction of the hippocampus prevents memory consolidation [64].

In lower animals, the limbic system is dominated by the sense of smell and taste. Odor and taste provides a very simple and straight forward evaluation of many objects. For example, depending on how something smells, one should either eat it, fight it, mate with it, or ignore it. In higher animals, the limbic system has evolved to become the seat of much more sophisticated value judgments, including human emotions and appetites. Yet even in humans, the limbic system retains its primitive function of evaluating odor and taste, and there remains a close connection between the sense of smell and emotional feelings.

Input and output fiber systems connect the limbic system to sources of highly processed sensory data as well as to high level goal selection centers. Connections with the frontal cortex suggests that the value judgment modules are intimately involved with long range planning and geometrical reasoning. Connections with the thalamus suggests that the limbic value judgment modules have access to high level perceptions about objects, events, relationships, and situations; for example, the recognition of success in goal achievement, the perception of praise or hostility, or the recognition of gestures of dominance or submission. Connections with the reticular formation suggests that the limbic VJ modules are also involved in computing confidence factors derived from the degree of correlation between predicted and observed sensory input. A high

degree of correlation produces emotional feelings of confidence. Low correlation between predictions and observations generates feelings of fear and uncertainty.

The limbic system is an integral and substantial part of the brain. In humans, the limbic system consists of about 53 emotion, priority, and drive submodules linked together by 35 major nerve bundles [63].

Value state-variables

It has long been recognized by psychologists that emotions play a central role in behavior. Fear leads to escape; hate lends to rage and attack. Joy produces smiles and dancing. Despair produces withdrawal and despondent demeanor. All creatures tend to repeat what makes them feel good, and avoid what make them feel bad. All attempt to prolong, intensify, or repeat those activities that give pleasure or make the self feel confident, joyful, or happy. All try to terminate, diminish, or avoid those activities that cause pain, or arouse fear, or revulsion.

Emotions provide an evaluation of the state of the world as perceived by the sensory system. Emotions tell us what is good or bad, what is attractive or repulsive, what is beautiful or ugly, what is loved or hated, what provokes laughter or anger, what smells sweet or rotten, what feels pleasurable, and what hurts.

It is also widely known that emotions affect memory. Emotionally traumatic experiences are remembered in vivid detail for years, while emotionally non-stimulating everyday sights and sounds are forgotten within minutes after they are experienced.

Emotions are popularly believed to be something apart from intelligence -- irrational, beyond reason or mathematical analysis. The theory presented here maintains the opposite. In this model, emotion is a critical component of biological intelligence, necessary for evaluating sensory input, selecting goals, directing behavior, and controlling learning.

It is widely believed that machines cannot experience emotion, or that it would be dangerous, or even morally wrong to attempt to endow machines with emotions. However, unless machines have the capacity to make value judgments (i.e. to evaluate costs, risks, and benefits, to decide which coarse of action, and what expected results, are good, and which are bad) machines can never be intelligent or autonomous. What is the basis for deciding to do one thing and not another, even to turn right rather than left, if there is no mechanism for making value judgments? Without value judgments to support decision making, nothing can be intelligent, whether or not it is biological or artificial.

Some examples of value state-variables are listed below, along with suggestions of how they might be computed. This list is not complete.

Good-is a high-level positive value state-variable. It may be assigned to the entity frame of any event, object, or person. It can be computed as a weighted sum, or spatio-temporal integration, of

all other positive value state-variables assigned to the same entity frame.

Bad-is a high-level negative value state-variable. It can be computed as a weighted sum, or spatio-temporal integration, of all other negative value state-variables assigned to an entity frame.

Pleasure-Physical pleasure is a mid-level internal positive value state-variable that can be assigned to objects, events, or specific regions of the body. In the latter case, pleasure may be computed indirectly as a function of neuronal sensory inputs from specific regions of the body. Emotional pleasure is a high level internal positive value state-variable that can be computed as a function of highly processed information about situations in the world.

Pain-Physical pain is a low-level internal negative value state-variable that can be assigned to specific regions of the body. It may be computed directly as a function of inputs from pain sensors in specific regions of the body. Emotional pain is a high-level internal negative value state-variable that may be computed indirectly from highly processed information about situations in the world.

Success_observed-is a positive value state-variable that represents the degree to which task goals are met, plus the amount of benefit derived therefrom.

Success_expected-is a value state-variable that indicates the degree of expected success (or the estimated probability of success). It may be stored in a task frame, or computed during planning on the basis of world model predictions. When compared with success_observed it provides a base-line for measuring whether goals were met on, behind, or ahead of schedule; at, over, or under estimated costs; and with resulting benefits equal to, less than, or greater than those expected.

Hope-is a positive value state-variable produced when the world model predicts a future success in achieving a good situation or event. When high hope is assigned to a task frame, the BG module may intensify behavior directed toward completing the task and achieving the anticipated good situation or event.

Frustration-is a negative value state-variable that indicates an inability to achieve a goal. It may cause a BG module to abandon an on-going task, and switch to an alternate behavior. The level of frustration may depend on the priority attached to the goal, and on the length of time spent in trying to achieve it.

Love -is a positive value state-variable produced as a function of the perceived attractiveness and desirability of an object or person. When assigned to the frame of an object or person, it tends to produce behavior designed to approach, protect, or possess the loved object or person.

Hate-is a negative value state-variable produced as a function of pain, anger, or humiliation. When assigned to the frame of an object or person, hate tends to produce behavior designed to attack, harm, or destroy the hated object or person.

Comfort-is a positive value state-variable produced by the absence of (or relief from) stress, pain, or fear. Comfort can be assigned to the frame of an object, person, or region of space that is safe, sheltering, or protective. When under stress or in pain, an intelligent system may seek out places or persons with entity frames that contain a large comfort value.

Fear-is a negative value state-variable produced when the sensory processing system recognizes, or the world model predicts, a bad or dangerous situation or event. Fear may be assigned to the attribute list of an entity, such as an object, person, situation, event, or region of space. Fear tends to produce behavior designed to avoid the feared situation, event, or region, or flee from the feared object or person.

Joy-is a positive value state-variable produced by the recognition of an unexpectedly good situation or event. It is assigned to the self-object frame.

Despair-is a negative value state-variable produced by world model predictions of unavoidable, or unending, bad situations or events. Despair may be caused by the inability of the behavior generation planners to discover an acceptable plan for avoiding bad situations or events.

Happiness-is a positive value state-variable produced by sensory processing observations and world model predictions of good situations and events. Happiness can be computed as a function of a number of positive (rewarding) and negative (punishing) value state-variables.

Confidence-is an estimate of probability of correctness. A confidence state-variable may be assigned to the frame of any entity in the world model. It may also be assigned to the self frame, to indicate the level of confidence that a creature has in its own capabilities to deal with a situation. A high value of confidence may cause the BG hierarchy to behave confidently or aggressively.

Uncertainty-is a lack of confidence. Uncertainty assigned to the frame of an external object may cause attention to be directed toward that object in order to gather more information about it.

Uncertainty assigned to the self-object frame may cause the behavior generating hierarchy to be timid or tentative.

It is possible to assign a real non-negative numerical scalar value to each value state-variable. This defines the degree, or amount, of that value state-variable. For example, a positive real value assigned to "good" defines how good; i.e., if

(1.9.1) $e :=$ "good" and $0 \leq e \leq 10$

then, $e = 10$ is the "best" evaluation possible.

Some value state-variables can be grouped as conjugate pairs. For example, good-bad, pleasure-pain, success-fail, love-hate, etc. For conjugate pairs, a positive real value means the amount of the good value, and a negative real value means the amount of the bad value.

For example, if

(1.9.2) $e :=$ "good-bad" and $-10 \leq e \leq +10$

then $e = 5$ is good $e = -4$ is bad
$e = 6$ is better $e = -7$ is worse
$e = 10$ is best $e = -10$ is worst
$e = 0$ is neither good nor bad

Similarly, in the case of pleasure-pain, the larger the positive value, the better it feels. The larger the negative value, the worse it hurts. For example, if

(1.9.3) $e :=$ "pleasure-pain"

then $e = 5$ is pleasurable $e = -5$ is painful
$e = 10$ is ecstasy $e = -10$ is agony
$e = 0$ is neither pleasurable nor painful

The positive and negative elements of the conjugate pair may be computed separately, and then combined.

VJ modules

Value state-variables are computed by value judgment functions residing in VJ modules. Inputs to VJ modules describe entities, events, situations, and states. VJ value judgment functions compute measures of cost, risk, and benefit. VJ outputs are value state-variables.

Theorem: The VJ value judgment mechanism can be defined as a mathematical or logical function of the form

where **E** is an output vector of value state-variables

V is a value judgment function that computes **E** given **S**

S is an input state vector defining conditions in the world model, including the self. The components of **S** are entity attributes describing states of tasks, objects, events, or regions of space. These may be derived either from processed sensory information, or from the world model.

Value judgment function **V** in the VJ module computes a numerical scalar value (i.e. an evaluation) for each component of **E** as a function of the input state vector **S**. **E** is a time dependent vector. The components of **E** may be assigned to attributes in the world model frame of various entities, events, or states.

If time dependency is included, the function $\mathbf{E}(t+dt) = \mathbf{V}(\mathbf{S}(t))$ may be computed by a set of equations of the form

$$e(j,t+dt) = (k\, d/dt + 1) \sum_i s(i,t)\, w(i,j) \tag{1.9.5}$$

where e(j,t) is the value of the j-th value state-variable in the vector E at time t

s(i,t) is the value of the i-th input variable at time t

w(i,j) is a coefficient, or weight, that defines the contribution of s(i) to e(j).

Each individual may have a different set of "values", i.e. a different weight matrix in its value judgment function **V**.

The factor (k d/dt + 1) indicates that a value judgment is typically dependent on the temporal derivative of its input variables as well as on their steady-state values. If $k > 1$, then the rate of change of the input factors becomes more important than their absolute values. For $k > 0$, need reduction and escape from pain are rewarding. The more rapid the escape, the more intense, but short-lived, the reward.

Formula (1.9.6) suggests how a VJ function might compute the value state-variable "happiness".

(1.9.6) happiness = (k d/dt + 1) (success - expectation
+ hope - frustration
+ love - hate
+ comfort - fear
+ joy - despair)

where success, hope, love, comfort, joy are all positive value state-variables that contribute to happiness

and expectation, frustration, hate, fear, and despair are all negative value state-variables that tend to reduce or diminish happiness.

In this example, the plus and minus signs result from +1 weights assigned to the positive-value state-variables, and -1 weights assigned to the negative-value state-variables. Of course, different brains may assign different values to these weights.

Expectation is listed in formula (1.9.6) as a negative state-variable because the positive contribution of success is diminished if success_observed does not meet or exceed success_expected. This suggests that happiness could be increased if expectations were lower. However, when $k > 0$, the hope reduction that accompanies expectation downgrading may be just as punishing as the disappointments that result from unrealistic expectations, at least in the short term. Therefore, lowering expectations is a good strategy for increasing happiness only if expectations are lowered very slowly, or are already low to begin with.

Figure 1-17 shows an example of how a VJ module might compute pleasure-pain. Skin and muscle are known to contain arrays of pain sensors that detect tissue damage. Specific receptors for pleasure are not known to exist, but pleasure state-variables can easily be computed from intermediate state-variables that are computed directly from skin sensors.

The VJ module in Figure 1-17 computes "pleasure-pain" as a function of the intermediate state-variables of "softness", "warmth", and "gentle stroking of the skin".
These intermediate state-variables are computed by low-level SP modules. "warmth" is computed from temperature sensors in the skin. "softness" is computed as a function of "pressure" and "deformation" (i.e. stretch) sensors. "gentle stroking of the skin" is computed by a spatio-temporal analysis of skin pressure and deformation sensor arrays that is analogous to image flow processing of visual information from the eyes. Pain sensors go directly from the skin area to the VJ module.

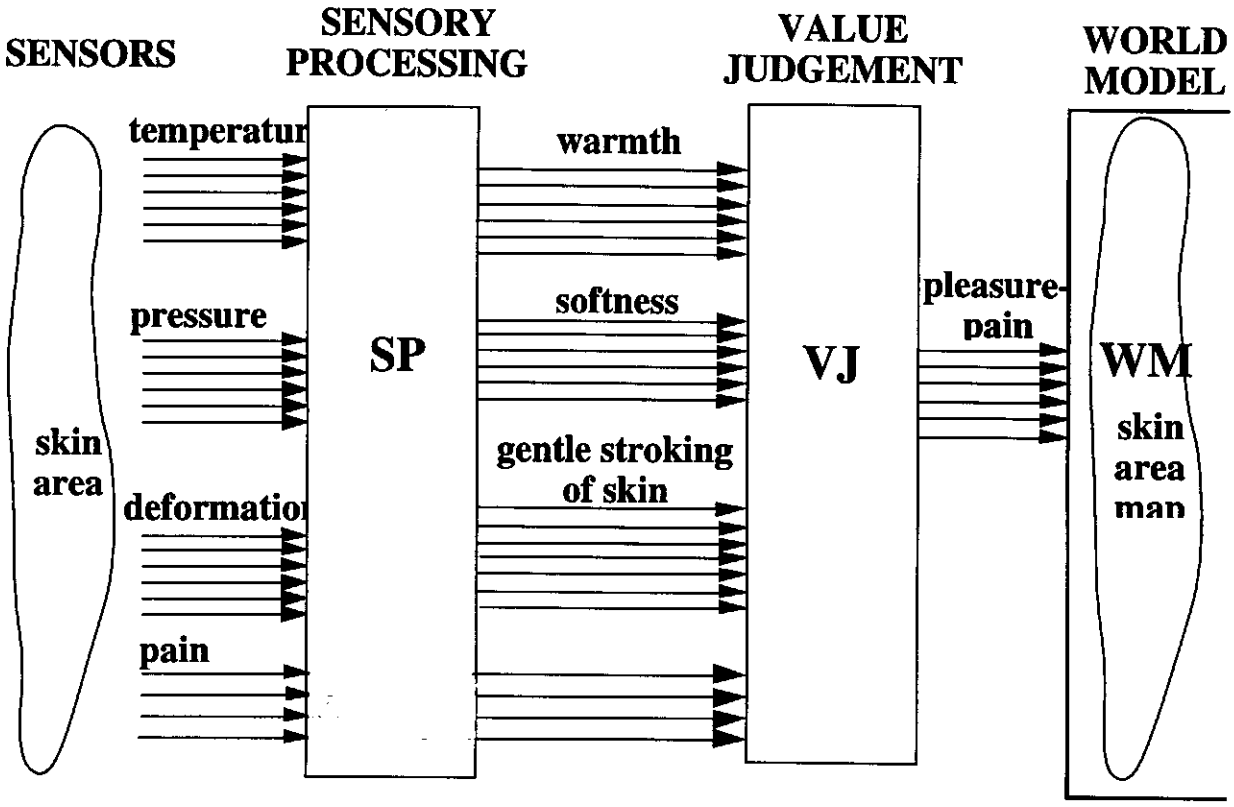
In the processing of data from sensors in the skin, all of the computations preserve the topological mapping of the skin area. Warmth is associated with the area in which the temperature sensors are elevated. Softness is associated with the area where pressure and deformation are in the correct ratio. Gentle stroking is associated with the area in which the proper spatio-temporal patterns of pressure and deformation are observed. Pain is associated with the area where pain sensors are located. Finally, pleasure-pain is associated with the area from which the pleasure-pain factors originate. A pleasure-pain state-variable can thus be assigned to the knowledge frames of the skin pixels that lie within that area.

Value State-Variable Map Overlays

When objects or regions of space are projected on a world map or egosphere, the value state-variables in the frames of those objects or regions can be represented as overlays on the projected regions. When this is done, value state-variables such as comfort, fear, love, hate, danger, and safe will appear overlaid on specific objects or regions of space. BG modules can

then perform path planning algorithms that steer away from objects or regions overlaid with fear, or danger, and steer toward or remain close to those overlaid with attractiveness, or comfort. Behavior generation may generate attack commands for target objects or persons overlaid with hate. Protect, or care-for, commands may be generated for target objects overlaid with love.

Projection of uncertainty, believability, and importance value state-variables on the egosphere enables BG modules to perform the computations necessary for manipulating sensors and focusing attention.



Confidence, uncertainty, and hope state-variables may also be used to modify the effect of other value judgments. For example, if a task goal frame has a high hope variable but low confidence variable, behavior may be directed toward the hoped-for goal, but cautiously. On the other hand, if both hope and confidence are high, pursuit of the goal may be much more aggressive.

The real-time computation of value state-variables for varying task and world model conditions provides the basis for complex situation dependent behavior [56].

1.8 Neural Computation

Theorem: All of the processes described in sections 4-8 for the BG, WM, SP, and VJ modules, whether implicit or explicit, can be implemented in neural net or connectionist architectures, and hence could be implemented in a biological neuronal substrate.

Modeling of the neurophysiology and anatomy of the brain by a variety of mathematical and computational mechanisms has been discussed in a number of publications [16,32,40,41,61,65-71]. Many of the submodules in the BG, WM, SP, and VJ modules can be implemented by functions of the form $\mathbf{P} = \mathbf{H}(\mathbf{S})$. This type of computation can be accomplished directly by a typical layer of neurons that might make up a section of cortex or a subcortical nucleus.

To a first approximation, any single neuron, such as illustrated in Figure 1-18, can compute a linear single valued function of the form

$$p(k) = h(S) = \sum_{i=1}^{N} s(i)\, w(i,k) \tag{1.10.1}$$

where $p(k)$ is the output of the k-th neuron

$\mathbf{S} = (s(1), s(2), \ldots s(i), \ldots, s(N))$ is an ordered set of input variables carried by input fibers defining an input vector

and $\mathbf{W} = (w(1,k), w(2,k), \ldots w(i,k), \ldots w(N,k))$ is an ordered set of synaptic weights connecting the N input fibers to the k-th neuron

$h(\mathbf{S})$ is the internal product between the input vector and the synaptic weight vector

A set of neurons of the type illustrated in Figure 1-18 can therefore compute the vector function

$$\mathbf{P} = \mathbf{H}(\mathbf{S}) \tag{1.10.2}$$

where $\mathbf{P} = (p(1), p(2), \ldots p(k), \ldots p(L))$ is an ordered set of output variables carried by output fibers defining an output vector

Axon and dendrite interconnections between layers, and within layers, can produce structures of the form illustrated in Figure 1-4. State driven switching functions produce structures such as illustrated in Figures 1-2 and 1-3. In sections 1-4 through 1-8 it has been shown how such structures can produce behavior that is sensory-interactive, goal-directed, and value driven.

The physical mechanisms of computation in a neuronal computing module are produced by the effect of chemical activation on synaptic sites. These are analog parameters with time constants

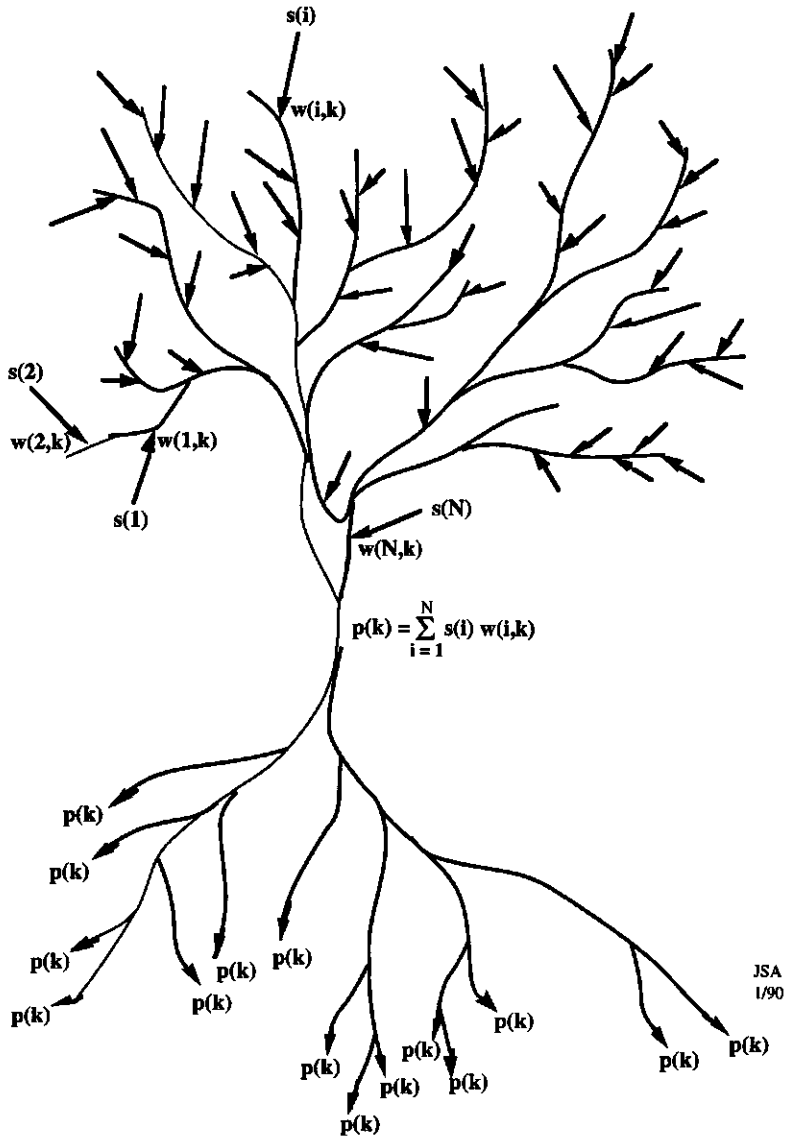


Figure 1-18. A neuron computes the scalar value p(k) as the inner product of the input vector s(1), s(2), . . . s(i), . . . s(N) and the weight vector w(1,k), w(2,k), . . .w(i,k), . . . w(N,k).

governed by diffusion and enzyme activity rates. Computational time constants can vary from milliseconds to minutes, or even hours or days, depending on the chemicals carrying the messages, the enzymes controlling the decay time constants, the diffusion rates, and the physical locations of neurological sites of synaptic activity.

The time dependent functional relationship between input fiber firing vector $\mathbf{S}(t)$ and the output cell firing vector $\mathbf{P}(t)$ can be captured by making the neural net computing module time dependent.

(1.10.3) $$\mathbf{P}(t+dt) = \mathbf{H}(\mathbf{S}(t))$$

The physical arrangement of input fibers in Figure 1-18 can also produce many types of non-linear interactions between input variables. It can, in fact, be shown that a computational module consisting of neurons of the type illustrated in Figure 1-18 can compute any single valued arithmetic, vector, or logical function, IF/THEN rule, or memory retrieval operation that can be represented in the form $\mathbf{P}(t+dt) = \mathbf{H}(\mathbf{S}(t))$. By interconnecting $\mathbf{P}(t+dt) = \mathbf{H}(\mathbf{S}(t))$ computational modules in various ways, a number of additional important mathematical operations can be computed, including finite state automata, spatial and temporal differentiation and integration, tapped delay lines, spatial and temporal auto- and cross-correlation, coordinate transformation, image scrolling and warping, pattern recognition, content addressable memory, and sampled-data, state-space feedback control. [65-69]

In a two layer neural net such as a Perceptron, or a brain model such as CMAC [32,40,41], the non-linear function

$$\mathbf{P}(t+dt) = \mathbf{H}(\mathbf{S}(t))$$

is computed by a pair of functions

(1.10.4) $$\mathbf{A}(\tau) = \mathbf{F}(\mathbf{S}(t))$$

(1.10.5) $$\mathbf{P}(t+dt) = \mathbf{G}(\mathbf{A}(\tau))$$

where

$\mathbf{S}(t)$ represents a vector of firing rates s(i,t) on a set of input fibers at time t

$\mathbf{A}(\tau)$ represents a vector of firing rates a(j,τ) of a set of association cells at time $\tau = t+dt/2$

$\mathbf{P}(t+dt)$ represents a vector of firing rates p(k,t+dt) on a set of output fibers at time t+dt

$\mathbf{F}$ is the function that maps $\mathbf{S}$ into $\mathbf{A}$

$\mathbf{G}$ is the function that maps $\mathbf{A}$ into $\mathbf{P}$

The function $\mathbf{F}$ is generally considered to be fixed, serving the function of an address decoder (or recoder) that transforms the input vector $\mathbf{S}$ into an association cell vector $\mathbf{A}$. The firing rate of each association cell a(j,t) thus depends on the input vector $\mathbf{S}$ and the details of the interconnecting matrix of interneurones between the input fibers and association cells that define the function $\mathbf{F}$. Recoding from $\mathbf{S}$ to $\mathbf{A}$ can enlarge the number of patterns that can be recognized by increasing the dimensionality of the pattern space, and can permit the storage of non-linear functions and the use of non-linear decision surfaces by circumscribing the neighborhood of

generalization. [40, 41].

The function **G** depends on the values of a set of synaptic weights w(j,k) that connect the association cells to the output cells. The value computed by each output neuron p(k,t) at time t is

$$p(k,t+dt) = \sum_j a(j)\, w(j,k) \tag{1.10.6}$$

where w(j,k) = synaptic weight from a(j) to p(k)

The weights w(j,k) may be modified during the learning process so as to modify the function **G**, and hence the function **H**.

Additional layers between input and output can produce indirect addressing and list processing functions, including tree search and relaxation processes [16,61]. Thus, virtually all of the computational functions required of an intelligent system can be produced by neuronal circuitry of the type known to exist in the brains of intelligent creatures.

2. Behavior Generation in the NIST-RCS Architecture

"Behavior" is the set of output activities of the System. Behavior Generation is an important function of control systems. It is especially important when we talk about complex systems. In the subsequent material we consider systems which are controlled by NIST-RCS[1] created for complex (and sometimes, large) systems.

"Behavior" is a term incorporated by the area of intelligent control only recently. It is sufficient to talk about "motion" in its general meaning: temporal development of the output coordinates, or "motion trajectory". It is convenient to cluster similar trajectories by their resemblance, or by their goal. Then, we receive such behaviors as "pursuit", "avoidance", "wall following", etc. In the evolution of NIST-RCS architecture, the roots of Behavior Generation are found in the definition of Task Decomposition:

> "TASK DECOMPOSITION -- Behavior is generated by a task decomposition system that plans tasks by decomposing them into subtasks, and scheduling them to achieve goals. Goals are selected and plans generated by a looping interaction between task decomposition, world modeling, and value judgment functions. The task decomposition system hypothesizes plans, the world model predicts the results of those plans, and the value judgment system evaluates those results. The task decomposition system then selects the plans with the best evaluations for execution. Task decomposition also monitors the execution of task plans, and modifies existing plans whenever the situation requires[2]."

Task decomposition is a part of planning which is equivalent to the design of systems' configuration. It can be done for a system with known components which can be arranged in various ways. The problem of such system decomposition into a set of interrelated subsystems is already solved at the design stage. Certainly, this affects the way the task is decomposed. On the other hand, in many cases, task decomposition and the system design are done simultaneously. This leads to more appropriate results for both. This Chapter is related to both of these cases: behavior generation in a previously designed system, and in a system which is being designed, but the design is not yet completed.

Task decomposition results in a hierarchy of tasks which corresponds to the hierarchy of the subsystems. In a goal oriented hierarchical system,[3] behavior is generated via special process of Task Decomposition. The latter includes Processes of Planning at each level of the hierarchy, and other processes-components of the hierarchical decision-making: execution including feedback compensation, task distribution, etc. This process is a *closed loop process*. Planning in turn, consists of several components including Job Assignment, Scheduling, Simulation, Evaluation,

---

[1] NIST-RCS is a technological representation of the architecture of rationally functioning systems. It is found is many natural and artificial examples. Discussion of these issues is beyond the goals of this paper.

[2] from J. Albus, "A Canonical Architecture for Intelligent Machine Systems" [72].

[3] Discussion of the theory and the laws of hierarchical systems exceeds the needs of this paper. Let us only notice that any hierarchical system is a goal-oriented system because the hierarchy emerges as a result of the intention to reduce the cost (e.g. complexity) of a system which already presumes the existence of a goal.

and Selection. After the best Plan is selected, it is supposed to be Executed which presumes allocating it with a proper subsystem of Actuation, Monitoring the process of Actuation and Compensation for the deviations that occur.

The processes of functioning can be adequately decided by using the set of *Loops of Functioning* . Each loop contains and circulates all information required to generate a particular loop behavior. The need in the loops of functioning is determined by the need of behavior generation and the need to provide *consistency of functioning* within the loop.

2.1 Loops of Functioning

2.1.1 Elementary Loop of Functioning (ELF)

Systems can be represented in two forms: in the form of input-output mappings and in the form of loops of functioning. The first approach is very common but is flawed: one should constantly worry about verification of the validity of mapping. Indeed, if the system maps input into output, a question should be asked: why? what for? To answer, we start describing what is going on in the external world. Then, we answer tentatively: probably, it maps inputs into outputs to provide for some external processes, and depending on these processes the input is shaped. We arrive at the second type of representation anyway: in the form of loops of functioning.

The loop of functioning is a tool for verifying our generalizations upon objects and attributes that we made in system representation. When we consider some particular scope of interest, it is based upon some limitations from above and from below. The upper bound of this scope is dictated by our priorities and based upon a tradeoff between our desire to broaden the scope and the growing burden upon our processing system. The lower bound determines the smallest interval measured and/or recorded (the value of resolution). Similar tradeoff also occurs. To make a representation within these bounds, we should ensure that this representation will be helpful for practice. The loop of functioning is the measure of our ability to apply our representation. Three conditions are supposed to be satisfied: a) the upper and lower bounds of the scope of attention should be selected properly, b) the resolution of representation must be satisfactory, c) under conditions "a" and "b", the objects are supposed to be properly determined within this limited world. Then we will succeed in the goals of our analysis: we will be able to construct a proper vocabulary of the loop of functioning; describe both the actions, and the measurements; and finally, will be able to close all cause-effect links generated within this loop.

The Elementary Loop of Functioning (ELF) is shown in Figure 2-1. This loop illustrates the obvious fact that all events in W (world's subset of interest) should be "sensed" (S). The results of sensing should be encoded (transformed into symbols) and organized (some primary identification is supposed to be performed), and submitted to the world model WM. On one hand, WM serves as a collection of knowledge about the World which is immediately perceived. On the

other hand, it should rely upon more complete and fundamental collection of knowledge (knowledge base) which will enable interpretation of the scenes and situations.

WM supports the subsystem of Behavior Generation (BG) where the decision-making occurs about all necessary activities (this subsystem is called sometimes "decision-making", or "planning-control"). BG submits its decisions to the system of actuation (A) which produces changes in the world. ELF is a simple (and obvious) way of organizing information about systems starting with bacteria and ecological niches and ending with systems of manufacturing and autonomous robots. All systems surrounding us in Reality can be represented in the form of ELF.

To judge the system's processes of functioning one should visualize the whole loop simultaneously otherwise some important components of the functioning can be omitted (see Figure 2-1,a). This loop contains two parts: the upper part includes SP, WM, and BG and is regarded as Control System and the lower part which includes S W and A and is regarded as Controlled System[4]. It is senseless to discuss one of them without another (see Figure 2-1b).

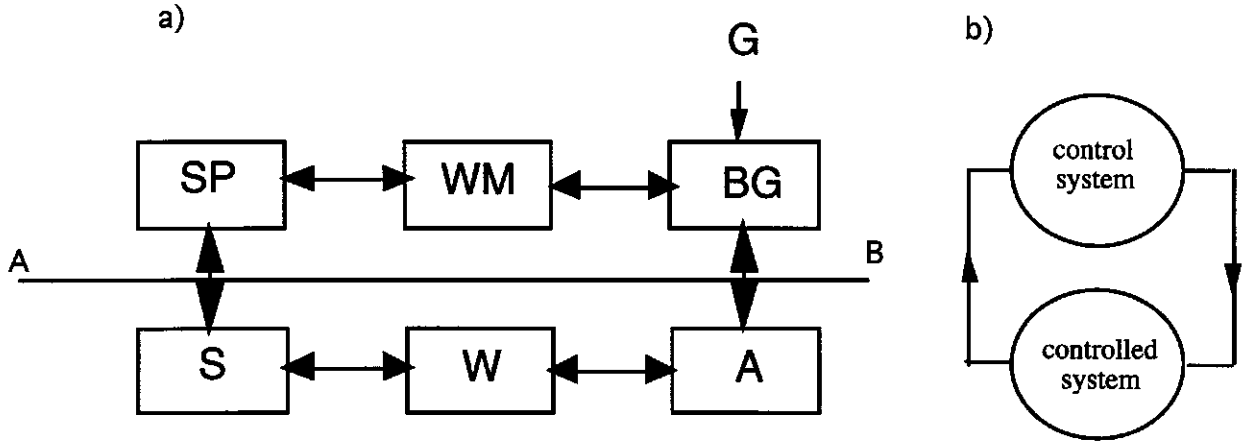


Figure 2-1 Elementary Loop of Functioning (ELF)
Notations: AB -the boundary between reality and its representation, G-goal, SP - sensory processing, or perception, WM-world model, or knowledge representation system, BG- behavior generation, A-actuators, W-world (the subset of interest), S-sensors

In this section we discuss the system without addressing the issue of its hierarchical structure. To describe a particular system with rational behavior in the form of ELF as shown in Figure 2-1, the following activities should be performed:

1) The vocabulary should be developed to describe the World situations. The list of possible situations of interest should be envisioned and possible description of these situations

[4] This terminology ("control systems", "controlled systems") is not always well understood, and sometimes is not acceptable for some users. This is true if the system includes humans (both on the supervising and performing levels). Therefore, in this paper, we try to adhere where possible, to the terms Decision-Making System (for the Control System) and Functioning System (for the Controlled System).

should be contemplated.

2) The means of receiving information about the World situations should be explicitly stated. In other words it should be explicated what are the sensors S (which may include physical devices of human sources of information). The World situations can be described (and later represented) by using a concrete set of sensors.

3) The signals from sensors S enter the sensory processing subsystem ("perception") SP that contains algorithms for decoding and organizing sensor information. Here, the results of sensing undergo their primary organization (including grouping and detection of correlations). The processes of primary organization should be capable of dealing with the sensor information arriving at the input of SP.

4) Primary organization requires knowledge of and communication with the subsystem of World Model (WM). WM contains relevant knowledge for interpretation of the SP output and for supporting the decision making activities[5]. The results of sensory processing are incorporated within or rejected by the knowledge base where the World Model is held and maintained (the set of knowledge and data bases in a variety of representational forms).

5) There exists a menu of possible goals. The responses for these goal assignments should be discussed. How should we respond to the assigned goal? What are the rules of decision-making that apply to the concrete case? After the goal G arrives to the subsystem of behavior generation BG, the latter performs decision-making including task decomposition, planning, and execution. BG requests and receives from WM the subset of knowledge required for the process of decision-making. BG employs WM for modeling and forecasting.

6) The list of actuators required becomes clear only after we discuss all possible alternatives of our responses to the goal assignments. From BG, the decision about behavior arrives to the actuator A which develop changes in the World W. World is considered as a system which is equivalent to its ontology. The validity of the ontology is regularly verified by *symbol grounding* operation.

ELF is a simplified version of the more complete diagram introduced by J. Albus and focusing upon the need in Value Judgment[6]. The ELF-diagram demonstrates an approximate "mirror" correspondence among the set of Reality containing physical devices (underneath the line AB) and the set of Cognitive activities in the computer system with its hardware and software (above this line). This means that the controlled system is represented within the control system or

[5] A question can emerge: how significant should the Knowledge Base (KB) be from which the WM is supposed to be taken? Later the lower level of resolution will be introduced containing its WM. The models of all levels, constitute a knowledge base which satisfies the need for a possible KB. In an intelligent system with learning, KB develops as a relational base in which all WM are stored and organized. In an artificial complex system, this KB should be an external source from which WM can be retrieved. Our discussion does not include many of supporting subsystems such as supply of knowledge, energy, materials, and so on.

[6] See Figure 1 in J. Albus, "Outline for a Theory of Intelligence" [2] or Figure 1-1 of this book.

the performing system—within the decision making system. This suggests that **BG** modules should be able to employ a direct representation of the **A** module[7]. More precisely, this employs all representations of **A** at all resolution levels. The role of the WM-module is to store and maintain these representations and make them available upon request from BG.

In the meantime, **A** produces changes in **W**. If these changes can be generalized and explained, we call them "behavior."[8] If no rational interpretation can be given then the changes are categorized as a random motion[9]. It should be understood that as Figure 2-2 demonstrates

Figure 2-2 A Fragment of the Controlled System of an Elementary Loop of Functioning
Notations: Δw -a change in the signal measured by a Sensor, a-an action developed by an Actuator

that the action **a** is generated by the actuator **A** and produces a change **Δw** in the world **W**. This change is considered to be a difference between the previous and the subsequent states of the world. It is measured by the sensors **S**. The string of n consecutive state changes $\{\Delta w_1, \Delta w_2, \ldots, \Delta w_n\}$ is called "(a particular) behavior" if a "law" of the string formation is found. Each elementary change happens during an elementary unit of the time scale accepted for a chosen space scale. Behavior can be defined also in the terms of actions produced that can lead to a different result since actions produced are not necessarily equivalent to the states observed.

Both time and space scales are determined by the particular resolution chosen to represent the present knowledge. Later, we show that a system should be represented at several levels of resolution simultaneously (i.e. at several time and space scales simultaneously.) The need for multiscale representation is determined by the fact that we face a problem of computational complexity. To deal with complex systems, we aggregate and decompose its components in space and time. This entails the hierarchies of representation and control.

---

[7] All of these modules are recursive ones which will be addressed later.

[8] In reality one can call "behavior" any kind of motion. "Motion" is a temporal change of states. But why introduce the idea of "behavior" instead of just saying "motion"? The term "behavior" is usually associated with an ability to characterize this behavior by its belonging to some general *type of behavior* satisfying some *purpose*. Otherwise, it would be more meaningful to use the word "motion" or "activities". It is linked also with our anthropomorphic tendencies to describe intelligent systems.

[9] Some researchers call it "random behavior".

Let us now consider an example of system decomposition linked with the scope of interest.

2.1.2 Primary decomposition of an ELF

ELF is defined as a system that is unified by a need to *exist as an entity* while achieving an assigned goal. In pursuit of this goal, ELF performs tasks that have been developed internally, or submitted externally. The phenomenon of being focused to achieve a concrete goal plays a *unifying role* and is fundamental in the discussion of ELFs and their operation. ELF can contain a set of subsystems which can be regarded as ELFs, too. This produces hierarchies of ELFs (see Subsection 2.1.4). We formulate the following statements as postulates.

Postulate (1) of Unity. The fundamental property of ELF is to exist as a goal seeking entity.

Postulate (2) of Recursion. Any ELF can be a part of another ELF (in which the ELF under consideration is nested) and can be a composition of other ELFs (which are nested in the ELF under consideration).

Postulate (3) of Existential Duality[10] . Each ELF consists of two parts (ELFs too) which are vitally required for its existence. The first handles goal-directed functioning while the second handles the subsistence (including maintenance) of the first one. They are denoted ELF^G and ELF^S correspondingly.

Corollaries. (1) Any ELF^G should be considered together with a hierarchy of goals to be pursued in external environment. (2) An ELF^S belongs to the environment in which ELF^G is functioning. (3) Any ELF^S should be considered together with a hierarchy of goals of maintenance to be performed internally. (4) An ELF^G is a part of environment for ELF^S. (5) Both ELF^G and ELF^S may form their nested systems.

ELF represents all activities of the system generalized. It can be applied both for ELF^G and ELF^S. Let us discuss the joint functioning of ELF^G and ELF^S within a ELF. The automata-like diagram in Figure 2-3 shows that any autonomous ELF has two groups of functioning: Goal Directed Functioning (GDF), and Regular Subsistence Functioning (RSF) which can be discussed by using separate loops of functioning..

GDF corresponds to the main goal-oriented function of a system, such as, the process of manufacturing of internal combustion engine. RSF corresponds to the maintenance system of the manufacturing plant. We consider three types of communication: W—GDF, W—RSF, and

[10] It should be introduced as a "postulate of existential plurality." However, because throughout the entire paper only the case with two parts of ELF is discussed, the duality was left in the formula of the postulate.

GDF—RSF. This communication is conducted in languages which do not allow for fully adequate interpretation of messages. Multiple stochastic functions acting within the environment (such as friction, backlash, temperature changes and others are the sources of error. Therefore, when GDF and RSF communicate with actuators A, the results of behavior generation often differ from the desired ones. This contributes to the reasons which invoke the need for feedback compensation.

A question can be raised concerning, how large the vocabularies of languages for automata representation should be. Their size obviously affects the size of transition and output functions of this sequential machine. The answer is embedded in the very nature of ELF, as can be visualized in Figure 2-1 to provide for communication among the modules of ELF. Prior experiences should be analyzed to find and use languages for this communication. Ultimately, ELF should become a learning machine[11], and learning is understood as a process of constantly generating new rules and concepts using experiences. The need to learn, affects all control subsystems of ELF as the controlled system is better understood (and/or undergoes changes). Learning is not simply a recording of all processes which have taken place; nor is it just memorization of experiences. Rather, it is development of the World Model and Rules of Sensory Processing and Behavior Generation by using generalization of experiences[12] [74].

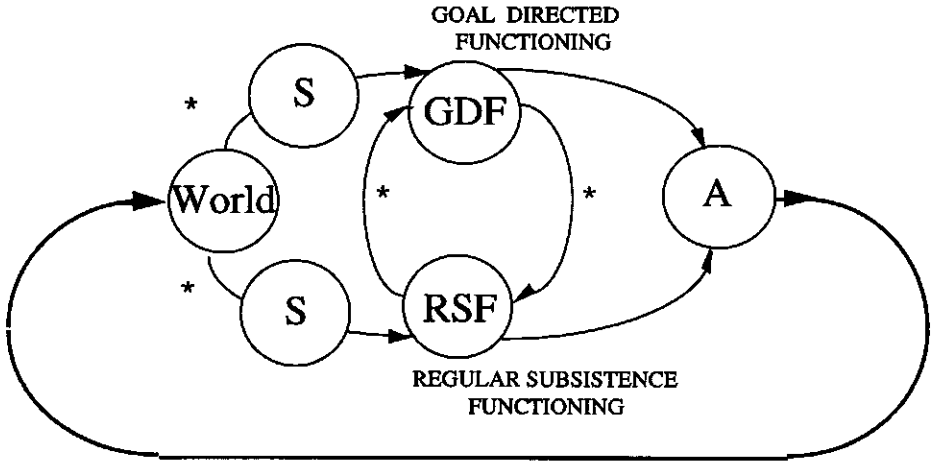


Figure 2-3. Decomposing ELF into two parts: one for a goal directed functioning (GDF) and another for a regular subsistence functioning (RSF)

[11] Learning can be distributed over the modules, or concentrated in a separate module (added to Figure 2-1.)

[12] Learning is understood as a process of collection of encoded experience and their generalization. Learning creates and maintains a multiresolutional system of knowledge representation. Notice, that the term learning is not clearly defined in literature. It includes a multiplicity of activities starting with memorization and updating and ending with proposing theories for generalization of prior observations.

Thus, the automaton in Figure 2-3 is not a standard automaton with finite vocabularies. It has open list vocabularies, transient, and output functions. All of these components of the automaton will change constantly if RCS is equipped with learning. The way this is done is determined by the architecture of NIST-RCS.

2.1.3 ELF as an Intelligent Agent

Instead of talking about loops of functioning, some professional communities talk about *intelligent agents*. An activity of a very complex system which is driven by a cognitive architecture can be substituted for joint functioning of many of the lowest level actuators. Each of them is equipped with a local "reactive intelligence," i.e. by "stimulus-response" rule of action. These "agents" are supposed to freely negotiate and discuss their local situations. (The expectation is that when the system relies upon the lowest level of rules "information-->action," the symbol grounding predicament[13] is always adequately resolved.) It is often assumed that the overall system becomes intelligent if all functions of the system are divided among simple intelligent agents equipped with a reactive/reflective rules, and each of them is oriented toward a simple elementary problem, thus, generating an elementary behavior. This is a completely unproven and highly questionable assumption.

Instead of using the ELF diagram (Figure 2-1), the structure of the agent is assumed in a form demonstrated in Figure 2-4.

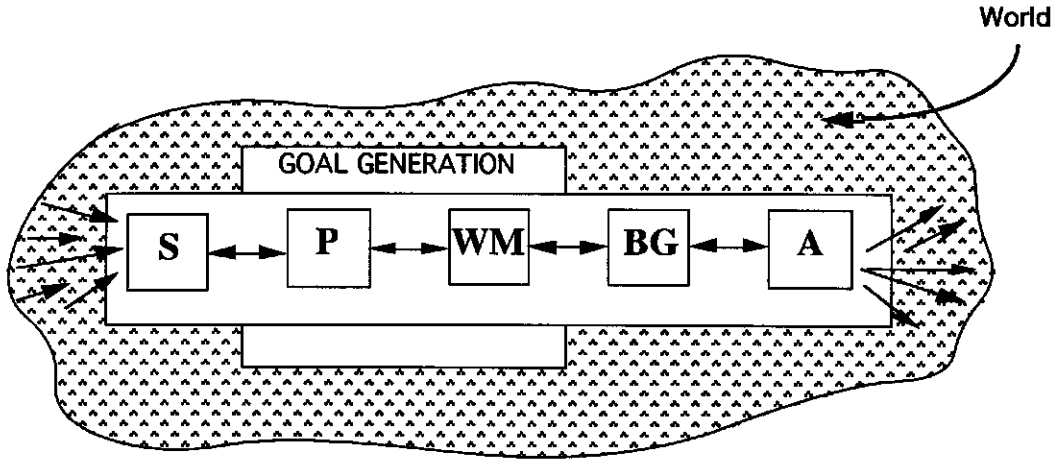


Figure 2-4. ELF viewed as an intelligent agent

So far, the only intelligent property demonstrated for this type of system design was "flocking together" of little mobile robots which are given a "skill of wandering aimlessly". There

[13] The problem of symbol grounding is typical for intelligent systems, of partially intelligent systems. The symbols utilized, or newly generated during functioning should be constantly verified by putting them in correspondence with the physical values represented by these symbols.

is no reason to expect that communities of simple "intelligent agents" can cope with large complicated problems with no centralized planning just by the virtue of their reactive behavior. Supposedly, this ability can emerge if the agents are given an opportunity to negotiate. So far there is no evidence that this can happen in other than simplistic cases. If the system is complex, the "skill of planning" should develop through generalization and formation of multiple levels of resolution.

It is also often assumed that *planning* and *reactive behavior* are completely different activities. Instead, they are simply different resolution viewpoints of the same activity of goal seeking. Within the intelligent system, the World is represented at many scales or at many resolutions. Intelligent systems generate behavior at many levels of resolution simultaneously. Any reactive behavior generated at a particular level is a "plan" outlined for the adjacent level of a higher resolution. Behavior is reactive if it reacts to the events observed in a situation. Certainly, plans can be reactive too. The picture is different when behavior is generated as a result of active anticipation (prediction) of the course of events. It will still remain reactive but it will react not to the current observations (this reactive response is always late) but rather to our anticipations (predictions) which can be often computed with high reliability.

Plans become active when we pursue the course of events by actively shaping the very event which is supposed to emerge. We call this "feedforward control" (FFC). On the contrary, reactive and even anticipatory compensation, we call "feedback control" (FBC)[14] [75]. Using this terminology helps in eliminating some of the persistent misconceptions such as counterpoising "planning" and "reactive control." Some of the AI researchers arrive at the same concept of FFC and FBC in a complicated way by discussing concepts of "situated actions" which presume to include "deliberative" and "reactive" actions as a kind of FFC and FBC incarnations[15] [76].

Finally, research is linked with a hope for a miracle. One such expectation is linked with a belief that "intelligence" is demonstrated when the solution emerges by itself out of communication among the mass of agents. "Emergence" has been observed for very large collections of nonlinear components (re: chaos, etc.) One can agree that the model of multiple elements, interacting and genetically searching, is a very inspiring model. Whether this model can produce "emergent" phenomena is hard to say. However, it is possible to show mathematically that complexity reduction requires a multilevel system of task-decomposition, and the reduction of complexity is formidable[16] [77].

---

[14] The concept of "feedforward control" is not a frequent term in the literature on control systems. Even less frequently one can find statement of equivalence between "feedforward control" and planning. This happened because the control community started thinking about planning-control continuum only recently.

[15] We already stated that any seemingly deliberative action, is actually a reactive action (at a lower resolution, larger scale, coarser granularity).

[16] Hierarchies of representation and organization has emerged as a result of the need in computational complexity reduction.

Another hope for a miracle frequently found in the literature, is a hope for an intelligent system *without representation*. This idea is motivated by a comparative analysis of "western" and "eastern" models of consciousness and thinking. The first is based upon discretization of the continuum into entities, while the second allows for "fluid", meditative processes which seem to be conducive of creativity.

2.1.4 Hierarchies of ELFs: the essence of NIST-RCS

In all these descriptions we talk about the system as a whole, not about a particular level of its hierarchy. It is important to realize that we will deal with the bulk of the

- processes of the overall actuation that either can be decomposed into a hierarchy of actions, or have actuators physically belonging to the different levels of the hierarchy, or both[17];
- processes of the overall measurement that also can either be decomposed or can be performed by sensors physically belonging to different levels of the hierarchy, or both.

The realization of this hierarchy of components has the following consequences:

1) it produces the levels of resolutions where the same system is represented, and the same processes develop at different temporal and spatial scales,

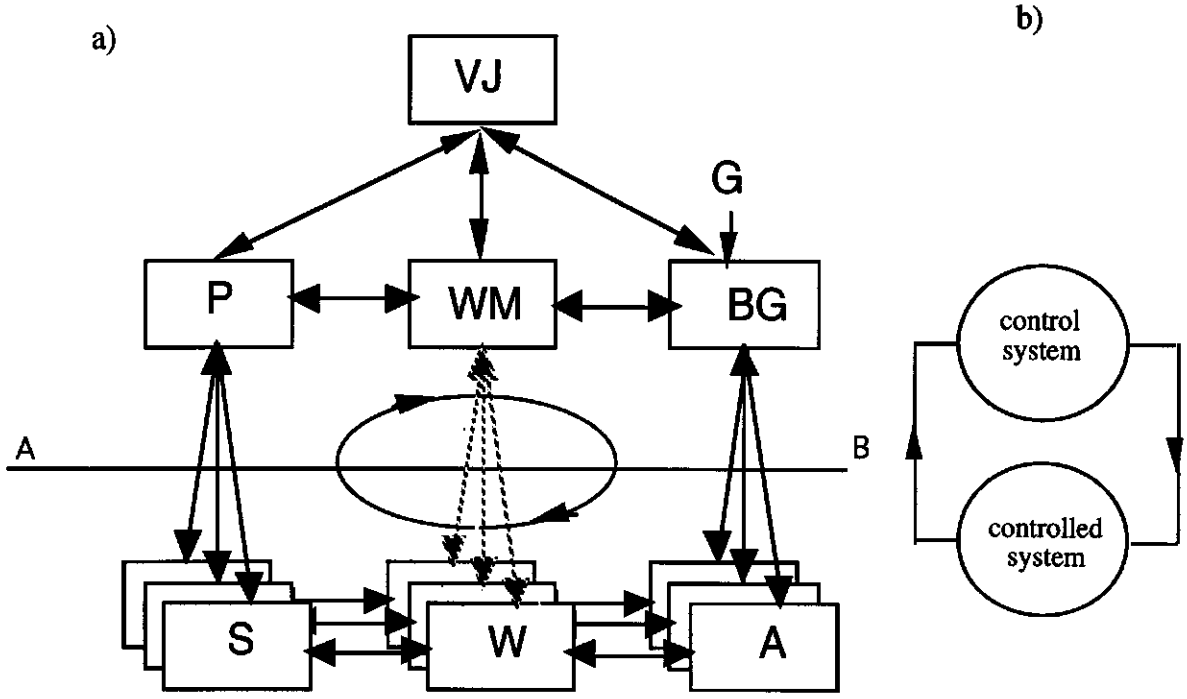


Figure 2-5. An Elementary Loop of Functioning with Multiple Controlled Subsystems

2) these levels have loops of control with different bandwidth intervals, and

[17] This process is similar to one which is often referred to as *abstraction propagation*.

equations with different sets of eigenvalues should be taken in consideration[18] [78],

3) each element of the control loop can be represented as a nested module.

The two level ELF is shown in Figure 2-5. The goal G arrives into BG-module ("behavior generation") where the solution of the problem is to be found. In Chapter 5, we will explain in detail how this occurs. The solution is to be transformed into a set of action tasks which should be submitted to the set of actuators ({A}) of the subsystems. All subsystems operate on the set of their Worlds ({W}). The latter are equipped by the set of sensors {S}. The signals from the sensors are integrated within the perception module (P), which updates the World Representation contained in the module of the World Model (WM). The best sensor {S}integration, best model {WM}, best solution obtained in {BG} are selected with the help of the Value Judgment module (VJ). AB - the boundary between reality and its representation.

The dotted arrow-lines between the module WM and the set {W} indicate a presence of the virtual correspondence between the real world W and its representation in the world model WM. These links are not channels of communication but they exist virtually[19]. The knowledge in WM is the system's best estimate of the real state of the World (W). The virtual links represent a noisy channel through which the system perceives the World (W). To the extent that the perception is a correct and relevant[20] representation of reality, behavior is more likely to be successful in achieving the goal. If virtual perception is not totally correct, behavior is less likely to be successful.

Figures 2-5a and b describe the Elementary Loop of Functioning consisting of two domains: the Control System and the Controlled System as shown in Figure 2-5,b. The number of controlled subsystems is not demonstrated. The issue is to separate the control and controlled subsystems from each other. Then, the levels of RCS can be separated too. Each of them will be considered a node of the NIST-RCS architecture.

The line AB is a divider between the domains of that which is to be controlled and that which is the system (or systems) of control[21]. It denotes the fact that the modules above this line are components of the control system. The designer is free to change them at the stage of design. The control engineer together with programmers are free to put different control algorithms and different knowledge representation in this part of the System. The modules underneath are the subsystems being controlled. Signals of several sensors are fused together within one P, and one BG can control several actuators. In the RCS hierarchy the lower level of the control system can be regarded as part of the system being controlled.

This leads to the multilevel structure shown in Figure 2-6 where 9 controlled units are

---

[18] This feature of multiresolutional systems is similar to the one featured by the "multirate control".

[19] This correspondence is possible because of our premise that the World is equivalent to its ontology. Lacking knowledge is taken care of by introducing stochastic components.

[20] The problem of relevance is addresses in a separate report on "World Modeling."

[21] Which in fact, can be a control system too, with its own "virtual" controlled system.

unified in 3 groups, possibly, these are three machines each equipped with three electrical motors.

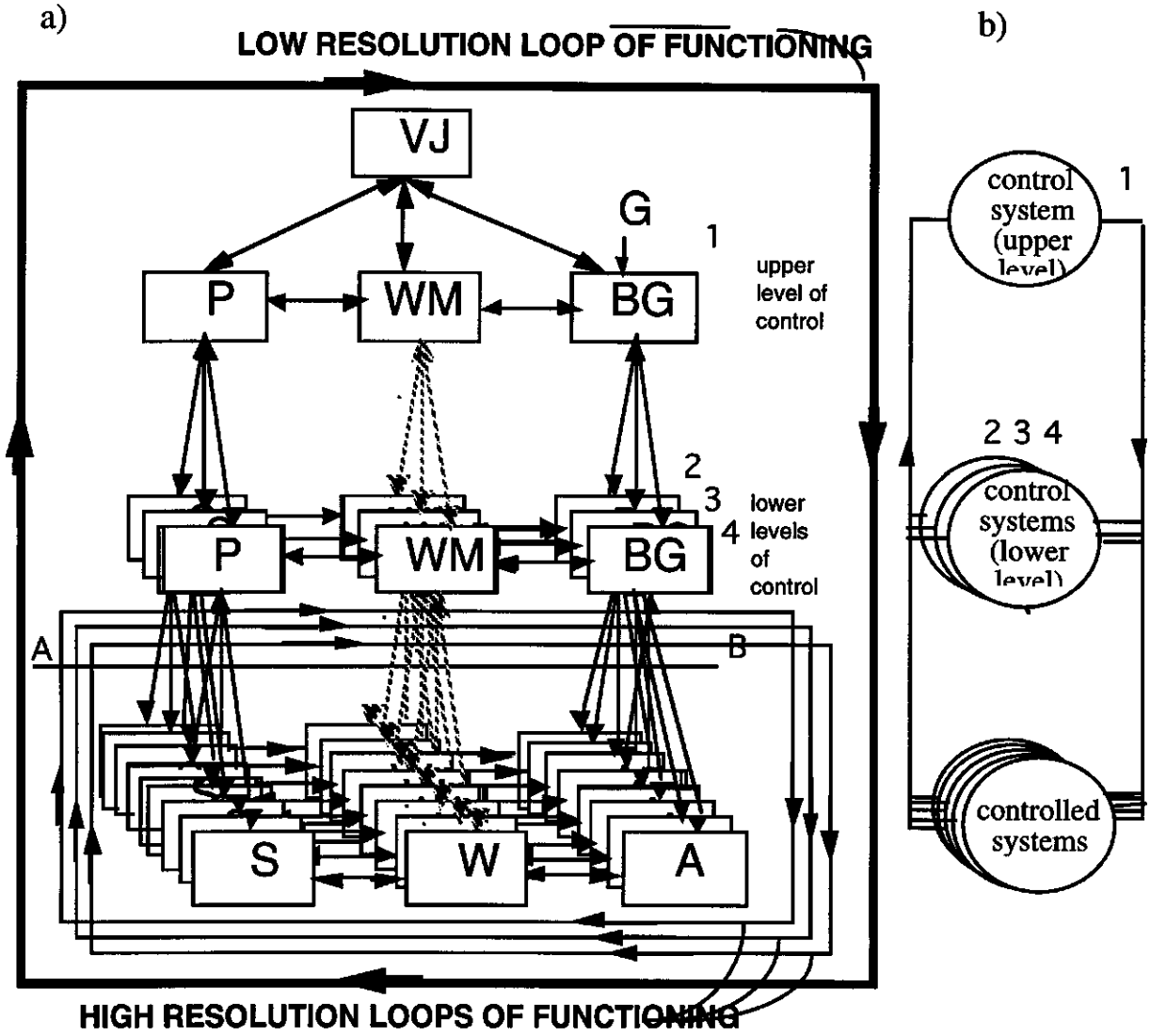


Figure 2-6. A two-level ELF, or NIST-RCS with 2 Control Levels (nodes)

These three machines together form a manufacturing cell. Functioning of this cell requires a control system (which is the upper level of control). This control system receives a goal which specifies what should be manufactured and how the results should be distributed in time (for the total horizon of time T_{cell}). This upper level control system generates some general control assignments to the machines which are part of the cell. Each of the machines is assigned what to manufacture and how to interact with each other within some horizon of time T_{mi}, where i=1, 2, 3 (for the total horizon of time $T_{mi}<T_{cell}$). Each of the machines has its own control system which

generates detailed control assignments to their actuators. These (detailed) control assignments prescribe the things to be done by each actuator. Often, it is impossible to use one control system for three controlled actuators; it is possible only in very simple cases. Then a new level of control should be introduced as shown later in Figures 2-10 and 2-11.

In Figure 2-6,b we represent our system in a simplified integrated manner as four nodes of the architecture (the number of controlled subsystems is not important). The levels are clearly determined. More details are given in Subsection 2.4 where the virtual loops are introduced.

2.2 Knowledge Propagation

Each ELF-loop of functioning is characterized by its vocabulary, relational grammar, and the set of messages that propagate a knowledge flow through the loop. At each level, it is required that:

- the output vocabulary of Perception be admitted by the input vocabulary of World Model,
- the output vocabulary of the World Model be admitted by the input vocabulary of Behavior Generation,
- the output statements of Behavior Generation module (in the form of commands) be admitted as the input Actuator commands,
- the output of the Actuators be realized as producing "motion" as temporal "changes" in the World Representation.
- the latter becomes the input vocabularies for Sensors and is transformed into the words of Sensors output vocabulary,
- the Sensor outputs arrive as the inputs of Perception module are transformed there into the input vocabulary of representation of the World Model. We can list these Knowledge Transformation Sets (K):

K_s (V_{is}, V_{os}, G_s, A_s)- Knowledge Transformation Sets of Sensors with its input and output vocabularies (V_{is}, V_{os},) transformation functions, which can be regarded as "grammars," "automata tables," or "transfer functions" (G_s) and the axioms (A_s,)

K_p (V_{ip}, V_{op}, G_p, A_p)- Knowledge Transformation Sets of Perception,

K_{wm} (V_{iwm}, V_{owm}, G_{wm}, A_{wm})- Knowledge Transformation Sets of World Model ,

K_{bg} (V_{ibg}, V_{obg}, G_{bg}, A_{bg})- Knowledge Transformation Sets of Behavior Generation,

K_a (V_{ia}, V_{oa}, G_a, A_a)- Knowledge Transformation Sets of Actuation,

K_w (V_{iw}, V_{ow}, G_w, A_w)- Knowledge Transformation Sets of the World.

It is important to understand that the loop operation can be consistent if and only if all Knowledge Transformation Sets manipulate with Knowledge represented and encoded at the same level of resolution. This means that the NIST-RCS controllers at each level of resolution should have a closed loop of signal (knowledge) flow.

One can deduce that ELF can function only if a mapping is possible between the

knowledge sets which pertain to the adjacent subsystems of ELF.

2.3 Integrated NIST-RCS Modules

We started exercising integration earlier when we considered a multiplicity of controlled systems to be one integrated control system, or a set of controllers— a control system at a level. Figure 2-6, b presents an example of integration.

From Figures 2.5 and especially 2.6 we can see that ELF-diagrams in the form of detailed graphs can be cumbersome because the number of levels grows, and the branching becomes large. This is why at each level of the hierarchy, the whole system underneath can be considered the "controlled system" and regarded as a single level of the "reality to be controlled." In Figure 2.7,a we show that the whole system at the "input-output terminals" of the upper level can be considered a controlled system within the dashed rectangle. Similar integration is performed in Figure 2-7,b.

The dividing line AB in Figure 2.7 works only for the first level of the ELF-hierarchy; the second level has its own dividing line A_vB_v which exists independently (in addition to AB).

a) b)

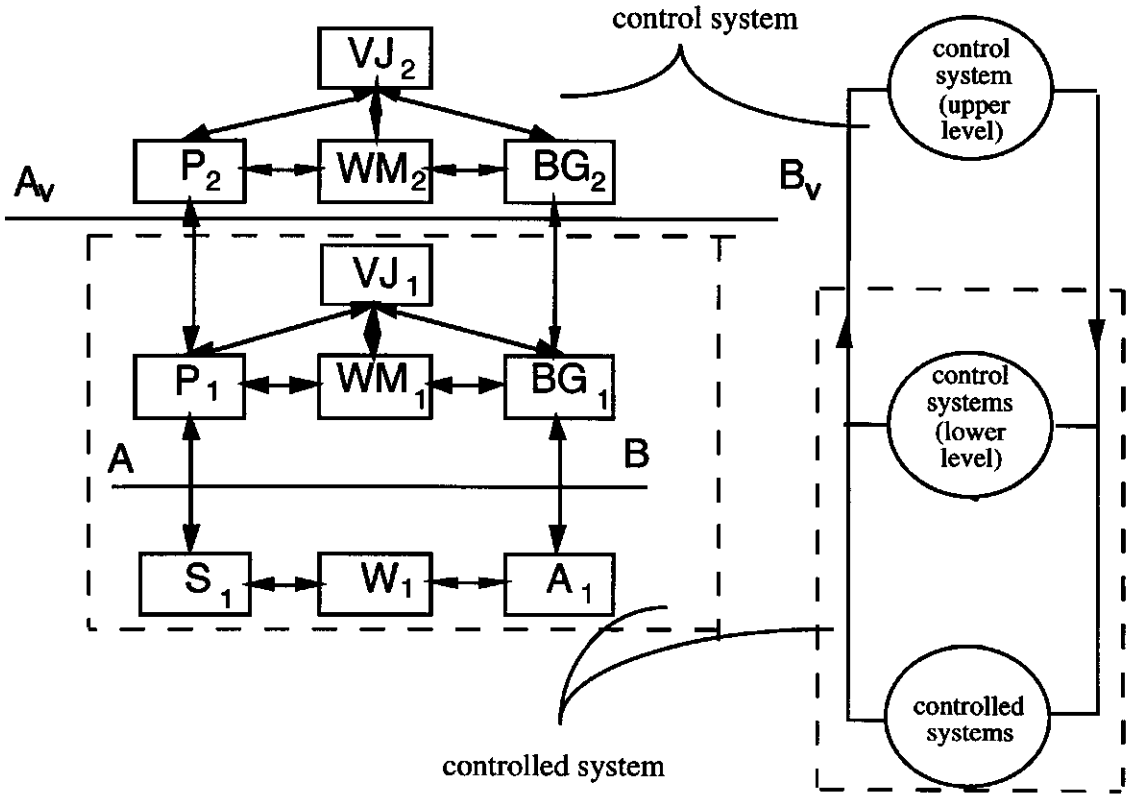


Figure 2-7 Modular Two-level NIST-RCS Structure in a Compact Form

Even at a comparatively low branching (in the case of Figure 2-5, branching is 3) the graphical representation becomes cumbersome. We treat the groups of modules which emerge after task decomposition as a single group module, each as shown in Figure 2-7. One can interpret this "shorthand" representation as introduction of the concept of "integrated modules" (or "group modules", Or "vector modules").

A vector module can be interpreted as an integrated entity at a resolution level which does not preclude the designer and/or a user from a successful pursuit of task decomposition. This is done by manipulating the Knowledge Transformation Sets (K) and by using linguistic or vector algebra techniques. The hierarchy of task decomposition is a linguistic hierarchy, which demonstrates the relation of inclusion (nesting) that exists among all corresponding sets K from different levels of resolution. This hierarchy H(K) is partially induced by the physical hierarchy of the structural system decomposition. Its construction as an aggregate of subsystems, represents the prior design efforts including the efforts to increase its efficiency by using different alternatives of task decomposition (e.g. experienced in the past, or intentionally synthesized).

2.4 Virtual Loops

Now, a question can be raised: what are the relationships between the modules at different levels? One of them is clear from the Figure 2-7: for the second level of control (see Figure 2-3), the first level of control system together with the controlled system, both surrounded by the dashed line are equivalent to the set of its virtual A-W-S as shown in Figure 2-8.

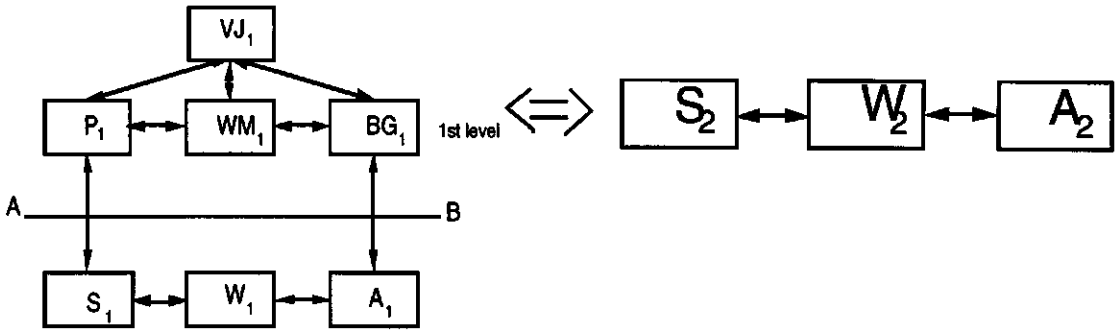


Figure 2-8.
The equivalence between the [System for the Level 1] and [a string $S_2W_2A_2$]

In the example with manufacturing cell (see sub-subsection 2.1.4. p. 91) it is clear that for the level assignment (a set of parts), or for the manufactured cell (a target object), it is not important that the cell consists of several machines, nor that each of these machines contains several actuators. This level would prefer to consider "cell" as a performing unit, or a control

system. The description of the job to be performed will be done in a generalized fashion as a group #xx which usually requires such effort in time, materials, and expense. If this level can avoid talking about the details linked with machining, or voltage inputs required by the actuators, the level (the RCS-node, the control system) will be able to handle several cells coordinating their activities by using the same input-output (interface) language.

Clearly, the [System for Level 1] includes the whole ELF of this level: with both the Controlled System $[S_1W_1A_1]$ and its Control System $[P_1WM_1BG_1VJ_1]$. Jointly they constitute the imaginary (virtual) Controlled System $[S_2W_2A_2]$ that exists as such, only in "imagination" of the second level of control. If one considers the first level of the control hierarchy (the lowest abstraction level, or the highest resolution level, or the low end controller) then it can be represented as it is shown in Figure 2-9a. In this Figure, the accurate parameters of the World Model are given for the high frequency clocks (the clocks of time discretization). The correctness of the physical model should hold only for a small vicinity of the present state.

This means that the overall physical model of the system can be simplified before using it for this control loop. The order of the dynamic model can be reduced, it can be linearized, and decoupled within an interval of time in the vicinity around the present state.

When the second level of resolution is considered, the external system of S-W-A will include the first loop as a component (see Figure 2-9,b). This is why the paradigm of execution for the level 2 is not the real world S_1-W_1-A_1 of the first level, but rather the *virtual* set including $\{VJ_1, P_1, WM_1, BG_1\}$ together with the system triplet $\{S_1$-W_1-$A_1\}$.

Everything in the "box" "SYSTEM CONTROLLED BY THE 2-ND LEVEL" can be considered a new (virtual set S_v-W_v-A_v consisting of virtual sensors, virtual world, and virtual actuators).

This world which can be denoted as $\{S_v$-W_v-$A_v\}$, is also real but its reality is valid only for the level 2. It has a larger discrete of time: the frequency of its clock is slower frequency of its clock or . Its physical model describes its processes of functioning within a more extended time-span (planning horizon) around the present state. However this model has a lower resolution of all values for the variables.

In conclusion, let us define virtual loop as the loop of functioning as it is visualized from the point of view of the observer associated with a particular resolution level. This "internal" observer is not interested either in a further decomposition of the "virtual", aggregated view, or in other levels of resolution and their scope of view (scope of attention). This allows us to introduce a level vocabulary that exercises the most efficient representation of the level.

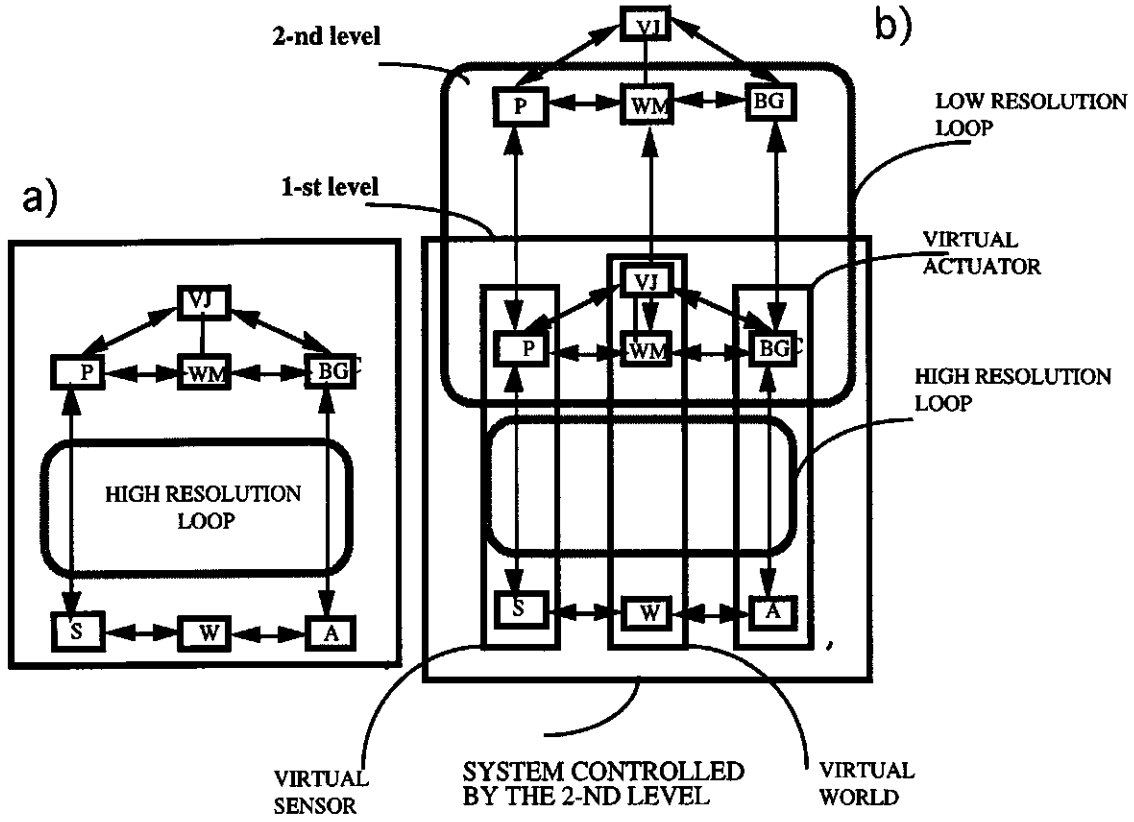


Figure 2-9. Representation of the "output" for different levels.

So, for the 2-nd level its system to be controlled, i.e. the virtual set of (ACTUATORS+WORLD+SENSORS)$_2$ is a set ({P_1, WM_1, BG_1, VJ_1, S_1, W_1, A_1} that requires considering the output of the virtual sensor S_2 as a result of corresponding generalization upon the information from S_1 combined with the activities of the set {P_1, WM_1, BG_1, VJ_1}. This is the meaning of the box "system controlled by the second level" in Figure 2-9.

The third level of resolution has the second level control loop as the substitute for the real external world (Figure 2-10). The time-vicinity within which the original physical model is allowed to be simplified is larger than in the case of the second level of resolution. The elementary time discrete is larger, the frequency is lower, and the accuracy is determined at lower frequency sampling intervals with the numerical value of the variables averaged over the increased period of sampling.

It is clear that both the SYSTEM CONTROLLED BY THE SECOND LEVEL from Figure 2-9 and the SYSTEM CONTROLLED BY THE 3RD LEVEL from Figure 2-10 are transformed into a set of virtual modules S_V-W_V-A_V represented for the corresponding level of control.

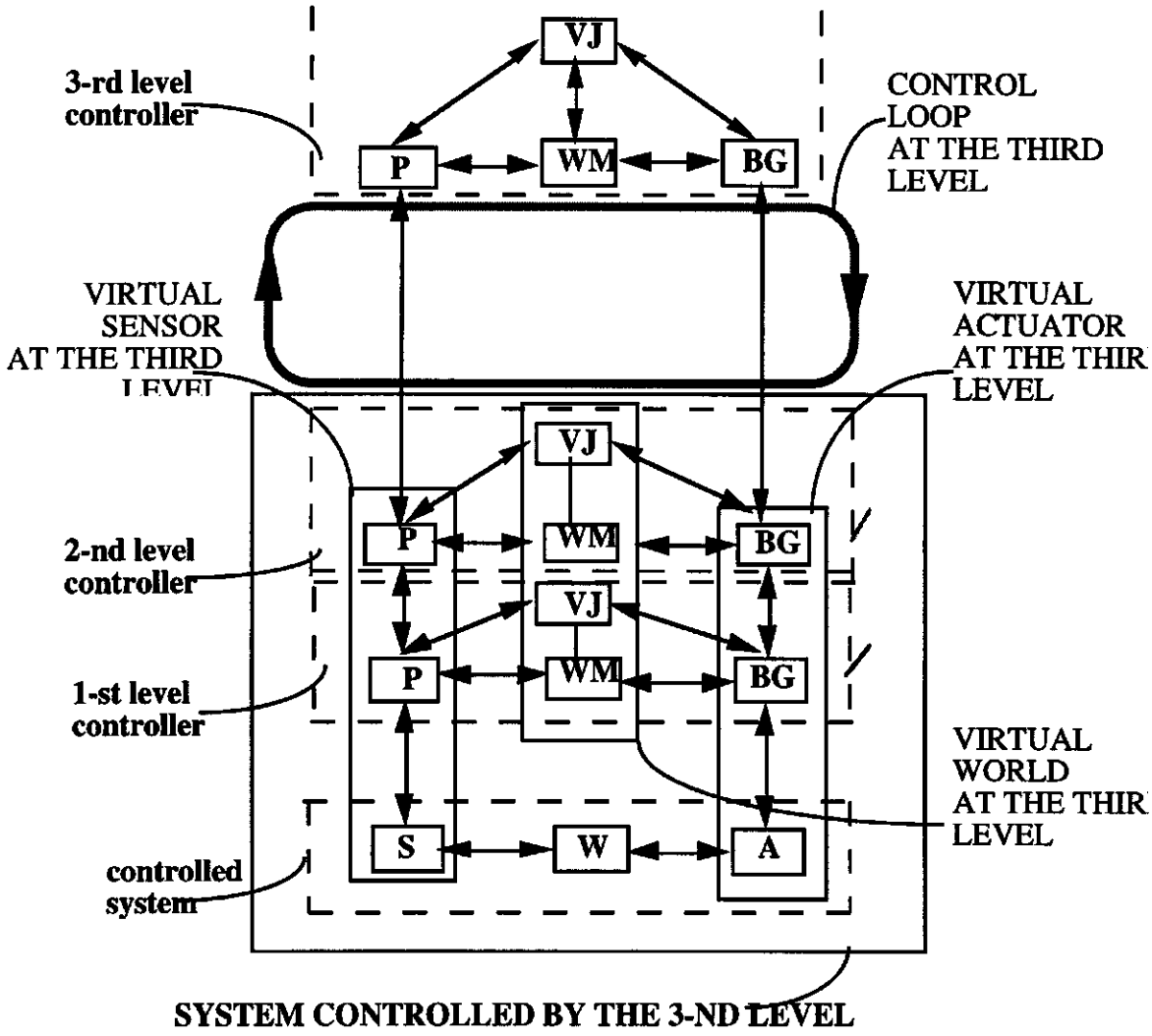


Figure 2-10. Virtual Triplet of Actuator-World-Sensor for the 3rd Level

The control loop "visualized" by the third level of control (its virtual control loop) works with the system of sensors-world-actuators which includes all other levels together. Each component of this virtual triplet constructed for the third level ($S_{VS}W_{V3}A_{V3}$) has its model and its parameters which has to represent the whole hierarchy of sensors and actuators with the ontological hierarchy[22] of the World at the third level.

On the other hand, for the lowest level controlled system, the whole hierarchy of controllers with its three level hierarchy of sensory processing, world model, and behavior generation, appears to be a single level virtual control system as shown in Figure 2-11. Figure

[22] We will consider the subset of the World at each particular level to be an ontological hierarchy constructed for this particular subset of the World in the view of a concrete Goal. The World is a "known World" as it is (or can be) represented in the Knowledge Base.

2-11 and Figure 2-10 do not contradict each other, they are complementary views.

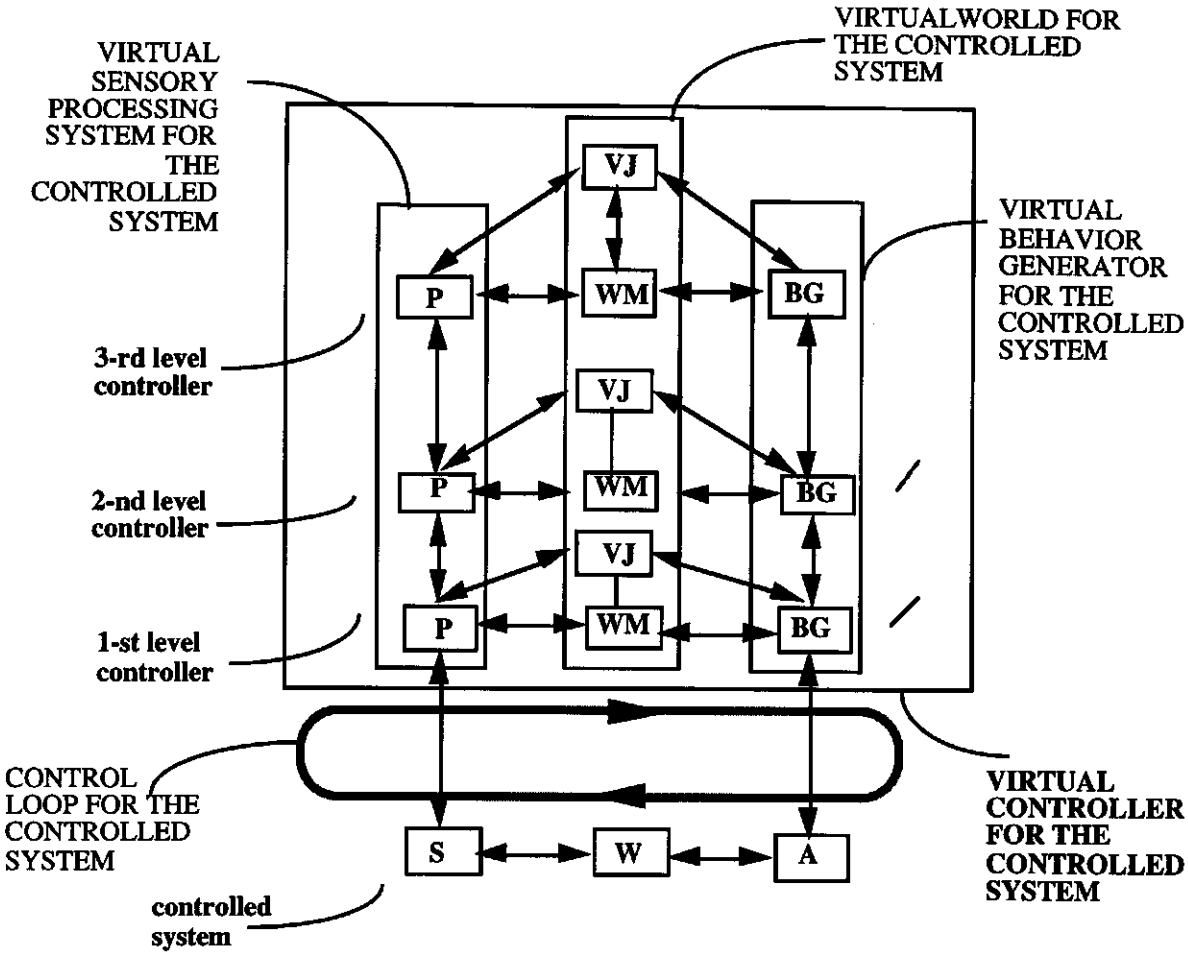


Figure 2-11 A three level control system as visualized by the "observer" linked with the controlled system

An intermediate level of the control system has the following options of representing itself in the terms of "virtual" representation. It can be represented:

• as a system which is simultaneously: a) a controlled system for the virtual control system above, or b) a control system for the virtual controlled system below (or both); in the first case, this system is the third level of the overall control system, in the second case, the system might be interpreted as the first level of the overall control system together with the lowest level controlled system

• as a part of the upper level virtual control system

• as a part of the lower level virtual controlled system.

These "statements of belonging" can be meaningful at the stage of design for modeling and

simulation as well as at the stage of functioning.

In the Figure 2-12 we present two views: with unified and with separated descriptions.

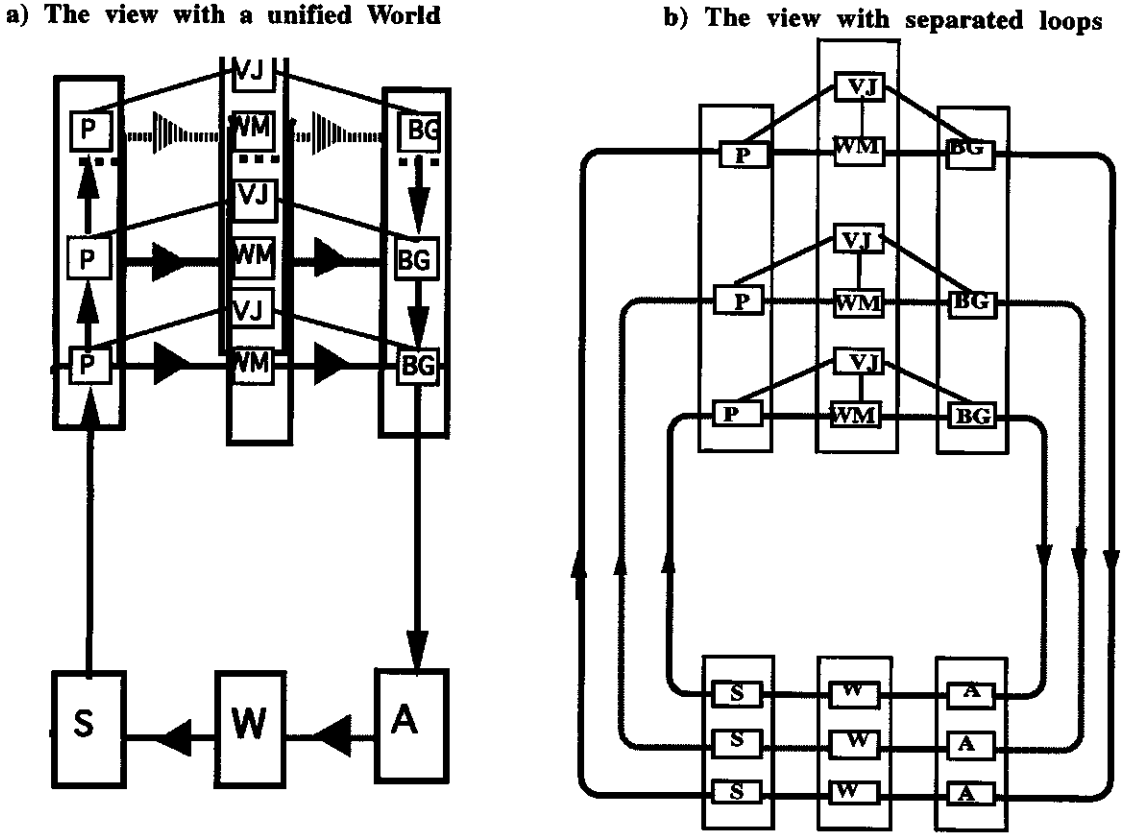


Figure 2-12. Two Views of the Loops of Functioning

Notations: P-perception, or sensory processing (SP), WM-world model, or knowledge, BG-behavior generation (or planning/control system), A-actuators, W-world, S-sensors

From Figure 2-12 we can see that the multilevel (multiresolutional) control architecture should be presented not by the Space-Time representation as it is typically done for conventional controllers, but rather by using a *set of Scale-Space-Time representations*. All of these loops have the same World as a part of their structure. However, because they have different vocabulary, the World is represented at different resolutions. Figure 2-12a is an intermediate step toward the Figure 2-12b. In the real system, we design and plan for each level separately, and the consistency of the results of this level ELF functioning can be achieved if we have satisfactory conditions determined by the actual rules of decomposition for the level above into the adjacent level below, or the rules of aggregation of the levels below into the adjacent level above. In the case of Figure 2-12,a it has been done for the levels (control systems) but it is not totally clear what happen with these loop at the very bottom of the diagram (within the controlled

system). In Figure 2-12,b we show that the loops can be separated from each other by decomposing the sensor-set, actuator-set, and even the World, (its ontology, of course.) There are three different Virtual Worlds driven by three different Virtual Actuators and perceived by three different sets of Virtual Sensors. (The real elements are parts of the hierarchy. The virtual ones are parts of the consistency check.)

In discussions of the same World, the descriptions are easily connected. Their connection should be demonstrated as a consistency-check. However, in each loop of the resolutional level the flow of knowledge is totally different from the flows of knowledge in other loops. The *virtual existence* of these different loops is the artifact of the RCS control systems.

In practice, the loops of different resolution levels are discussed and often used anyway. However, the consistency conditions are frequently not satisfied. Neither the vertical "between-the-loops" consistency is checked on a regular basis, nor the horizontal "within-the-loop" consistency. Representing the loops consistently internally and externally is one of the major problems of developing the World Model.

2.5 Real-time control and planning: how they are affected by the sources of uncertainty

The process of *recursive behavior generation* is one of the fundamental components of the hierarchical architectures and computational schemes. It determines both the design and the functioning of the NIST-RCS Architecture. The fundamentals of this concept are presented here with orientation to the further application of the concept of recursive behavior generation for the RCS Architecture of large systems (e.g. manufacturing). It was indicates in the literature [72, 73], that in many cases, the intelligent task decomposition[23]:

a) requires the ability to reason about space and time,

b) alludes to geometry and dynamics, and

c) contains an implicit demand to formulate, synthesize and/or select plans based on values such as cost, risk, utility, and goal priorities.

Task planning and execution must be done in the presence of uncertain, incomplete, and sometimes incorrect information which circulates in the Loop of Functioning. The subsystem of Behavior Generation should allow for task decomposition using the following sources of uncertainty simultaneously:

a) the implicit statistical experience embodied in the structure of the system,

b) the approximate models stored in the knowledge/data bases, and

c) the stochastic data available from sensors.

The great danger that should be avoided is that the relations among the loops should not be

[23] Notice, that "intelligent" task decomposition is expected to satisfy the specified performance and provide for the minimum of computational complexity, or maximum performance at the required complexity.

violated (see relations of inclusion in subsection 3.1).

For the System to succeed in a dynamic and unpredictable world, its task decomposition (and therefore, its planning) at the high resolution levels must be accomplished in real-time because the dynamics and unpredictability of the real world at these levels occurs in real time. Either one should properly predict these factors online, or properly respond to them online, or preferably, both. This term "online" reflects different aspects of NIST-RCS functioning. It means that NIST-RCS operates while the System[24] functions, i.e. simultaneously with the System it is designed to control.

The levels of low resolution do not require any fast, real-time decision making. Because the Loops of Functioning of the different levels (as perceived by NIST-RCS) operate at different time scales, the phenomenon of being "simultaneous" does not necessarily mean to happen at the same moment of astronomical time! Coincidence of the time instances is substituted by satisfaction of the conditions of inclusion for the sampling time units of corresponding resolution levels (see subsection 3.1).

Secondly, it means that NIST-RCS by itself has the same property of all modules at all levels operating simultaneously." (Some of the procedures are supposed to be done as the need emerges. Other issues such as learning presume that the process of "preparation" is done offline while the process of using the results of this "preparation" are executed online.)

Thirdly, in order to achieve online Behavior Generation via task decomposition for a sufficiently complex system, it is necessary to transform the planning problem into a hierarchy of planning problems pertaining to levels of resolution with different temporal planning horizons and different degrees of detail at each hierarchical level. This should be done to provide for a proper operation of the Loops of Functioning associated with each of the levels.

When the level is sufficiently low in abstraction and high in resolution, the planning processes should be performed in real time. They can be considered a "feedforward control" which presumes a particular horizon of planning at each level of resolution. Once this is done, it is possible to employ a multiplicity of planners to simultaneously generate and coordinate plans for many different subsystems at many different levels. "Behavior results from a behavior generating system element that selects plans and executes tasks. Tasks are recursively decomposed into subtasks, and subtasks are sequenced so as to achieve task goals. . . The behavior generating system hypothesizes plans, . . . then selects the plans with the highest evaluation for execution. The behavior generating system element also monitors the execution of plans, and modifies existing plans whenever the situation requires."[25] [73]

---

[24] Any machine or aggregate including a robot, an autonomous vehicle, a manufacturing floor, or others which are supposed to be controlled by NIST-RCS, are referred to as the *System*.

[25] Monitoring the execution of plans can be translated into a normal control systems paradigm: it is the feedback control.

3. The General Framework.

3.1. Properties of the Recursive Hierarchy

The purpose of Behavior Generation is the determination of the description and parameters of the intermediate steps needed to get from the present state to the Goal state. This is done via Task Decomposition. The latter is a part of the overall process of decision making, spreading top-down and bottom-up in the multiresolutional hierarchy of the RCS controller. Task decomposition leads to a hierarchy of tasks and subtasks. All sufficiently complex system are designed, or tend to organize themselves in a hierarchy since it increases the efficiency of computation[26]. This efficiency has been discussed in many sources both qualitatively and numerically.[27]

The NIST-RCS applies a special type of hierarchy called a *recursive hierarchy*. This hierarchy has a special relation between the nodes of the adjacent levels: all nodes of the set of m nodes $\{n_j\}_i$, (where j=1, 2, ... , m; at the i-th level of resolution counting levels from bottom to top) attached to a particular k-th node $n_{k(i+1)}$ of the adjacent (i+1)-th level from above, are obtained as a result of the special procedure called *decomposition* or *refinement*.. This procedure presumes inversely that the properties of the node $n_{k(i+1)}$ can be obtained from the properties of nodes $\{n_j\}_i$ by a special procedure called *aggregation* or *generalization*[28] [21, 77, 79-81] interpreted as integration over the j index. Decomposition alludes to the refinement of properties and functions while aggregations done via their integration which is considered to be the essence of generalization.[29]

These recursive properties of our multiresolutional hierarchy are determined by the Inclusion Properties of the Knowledge Transformation Sets (K):

$$K_{si} \supset K_{s(i-1)} \supset \ldots \supset K_{s2} \supset K_{s1} \qquad K_{bgi} \supset K_{bg(i-1)} \supset \ldots \supset K_{bg2} \supset K_{bg1}$$

$$K_{pi} \supset K_{p(i-1)} \supset \ldots \supset K_{p2} \supset K_{p1} \qquad K_{ai} \supset K_{a(i-1)} \supset \ldots \supset K_{a2} \supset K_{a1}$$

$$K_{wmi} \supset K_{wm(i-1)} \supset \ldots \supset K_{wm2} \supset K_{wm1} \qquad K_{wi} \supset K_{w(i-1)} \supset \ldots \supset K_{w2} \supset K_{w1}$$

The sign of inclusion denotes that the knowledge set of a particular module at a particular level is a subset (obtained by decomposition) of the knowledge set belonging to the same module at the next higher resolution level. The multiresolutional hierarchy of knowledge under these

---

[26] We give references only in some cases when it is required in the context of the issue discussed. Most of the references will be given only in the final version of the material.

[27] Aggregation is creation of assemblies out of objects (or concepts) components. Generalization is necessarily associated with forming a concept which belongs to a different granularity of representation (a coarser granularity, a lower resolution, a larger scale). Some of the assemblies are generalizations. Abstraction is finding a class of objects (or concepts) based upon a class-property. Some of the abstractions are generalizations.

[28] Both decomposition and generalization invoke the issues of instantiation us generalization, and specialization us abstraction. These issues are not discussed in this book.

[29] One can see that the relationships which is characteristic of this hierarchy is parametric and functional "inclusion" rather than "partition" and "inheritance" which in some cases are just particular cases of the "inclusion".

conditions can be represented graphically as a tree focused (by attention) at each level of resolution (see Figure 3-1,a). For comparison, we show a tree not truncated by attention (Figure 3-1, b).

As we consider development of all processes in time, we should add to our set condition of inclusion also a "simultaneity condition" which can be represented as follows:

$$\Delta t_{wmi} \supset \Delta t_{m,i-1} \supset \ldots \supset \Delta t_{wm2} \supset \Delta t_{wm1}$$

which states the nestedness of the corresponding time intervals.

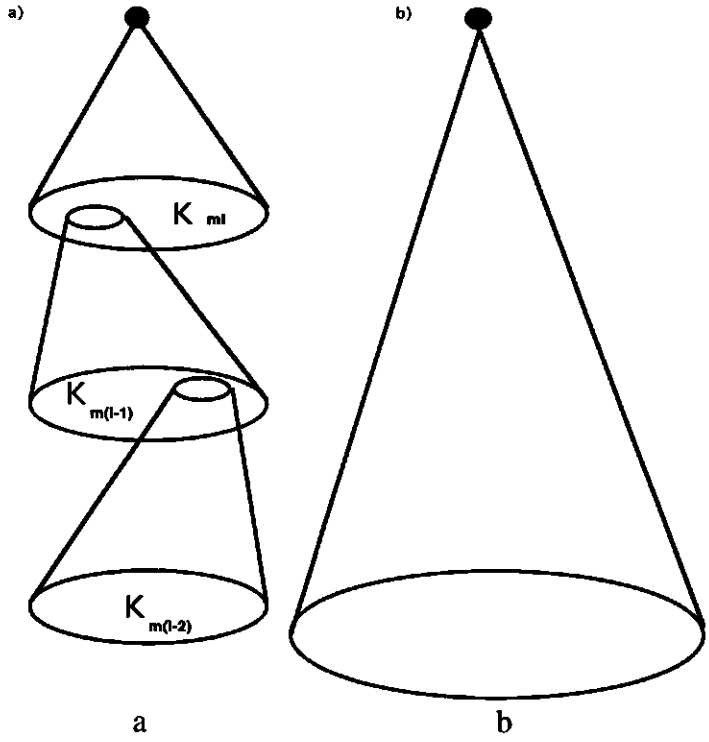


Figure 3-1 Tree of Knowledge Representation Focused by Attention

Recursive hierarchy is linked with the processes of recursive algorithms of decision-making which generate top-down/bottom-up processes of refinement/generalization propagating within the NIST-RCS control system[30] [82]. These lead to the following interesting property which entails the Inclusion Properties of the Knowledge Transformation Sets: all global properties and functions of the system which can be represented by the recursive hierarchy *contain* all properties of the lower levels of the hierarchy (higher resolution levels) in an integrated (generalized) form. The subsystems of the higher resolution levels are not totally autonomous, but are expected to carry out the assigned tasks assisted by other cooperating agents at their own level in supporting the systemic goals and objectives of their own parent. Their autonomy is limited by the fact that

[30] These algorithms are frequently called *relaxation* algorithms. They employ the same beneficial properties of increasing resolution and reducing the scope of attention from level to level top-down as NIST-RCS does. These techniques of forming multiresolutional systems are employed not only in the area of domain decomposition [82] but in many other areas including waveless, fractal, MultiMate control, and others.

they are contained within the adjacent lower resolution level in the pursuit of its Goal. The system of recursive hierarchy is built using the recursive modules[31].

The architecture of NIST-RCS belongs to the class of recursive hierarchies which are designed to process information and data by changing the computational model from level to level. This increases the resolution, which is compensated by a reduction in the interval of computation. It is possible to demonstrate that recursive computations always lead to nested structures of information and data and ends up with a system of nested Knowledge Transformation Sets {K}. The property of *nesting* allows for decoupling of the levels functionally. However, even after decoupling is done, levels depend on each other. Although the decoupling is done, the normal functioning requires satisfaction of nesting conditions with the neighbors from above and from below. This property is especially important if a search is required instead of using analytical methods, and if the changes in resolution are accompanied by changes in vocabulary.

Figure 3-1 implies that a set of concrete levels of resolution is determined for a system. The values of resolution (granularity, scale, accuracy which all define the indistinguishability zone) cannot be assigned arbitrarily. It has been shown, that there exists a set of resolution values that minimizes the computational complexity for a given set of specifications [77].

3.2 Nesting of the Virtual ELFs

Hierarchical control always implies the phenomenon of nesting. Let us apply this implication to the system earlier presented in Figures 2-6 or 2-12. These multilevel systems can be represented in the form of the virtual ELF-loop which is shown in Figure 3-2. The situation in the world W_i is measured by the sensors S_i, and their signals enter to the subsystem of perception P_i (which contains algorithms for processing sensor information.) Here, the results of sensing undergo their primary organization. After this, they are incorporated (or rejected) by the Knowledge Base K_i where the World Model WM_i is held and maintained (the set of knowledge and data bases in a variety of representation forms.)

After the goal G arrives to the subsystem of behavior generation BG_i , the latter performs decision-making including task decomposition, planning, and execution. BG_i requests and receives from K the subset of knowledge required for the process of decision making. From BG_i, the decision about behavior arrives to the actuator A_i which develops changes in the World W_i.

[31] The term "recursive" should be understood in the sense of the recurrence relation being used as a basis for the computations supported (see [83]). Recursive computations and consequently designed hierarchies is typical in information systems (84), estimation and control (85), signal processing (86), and others.

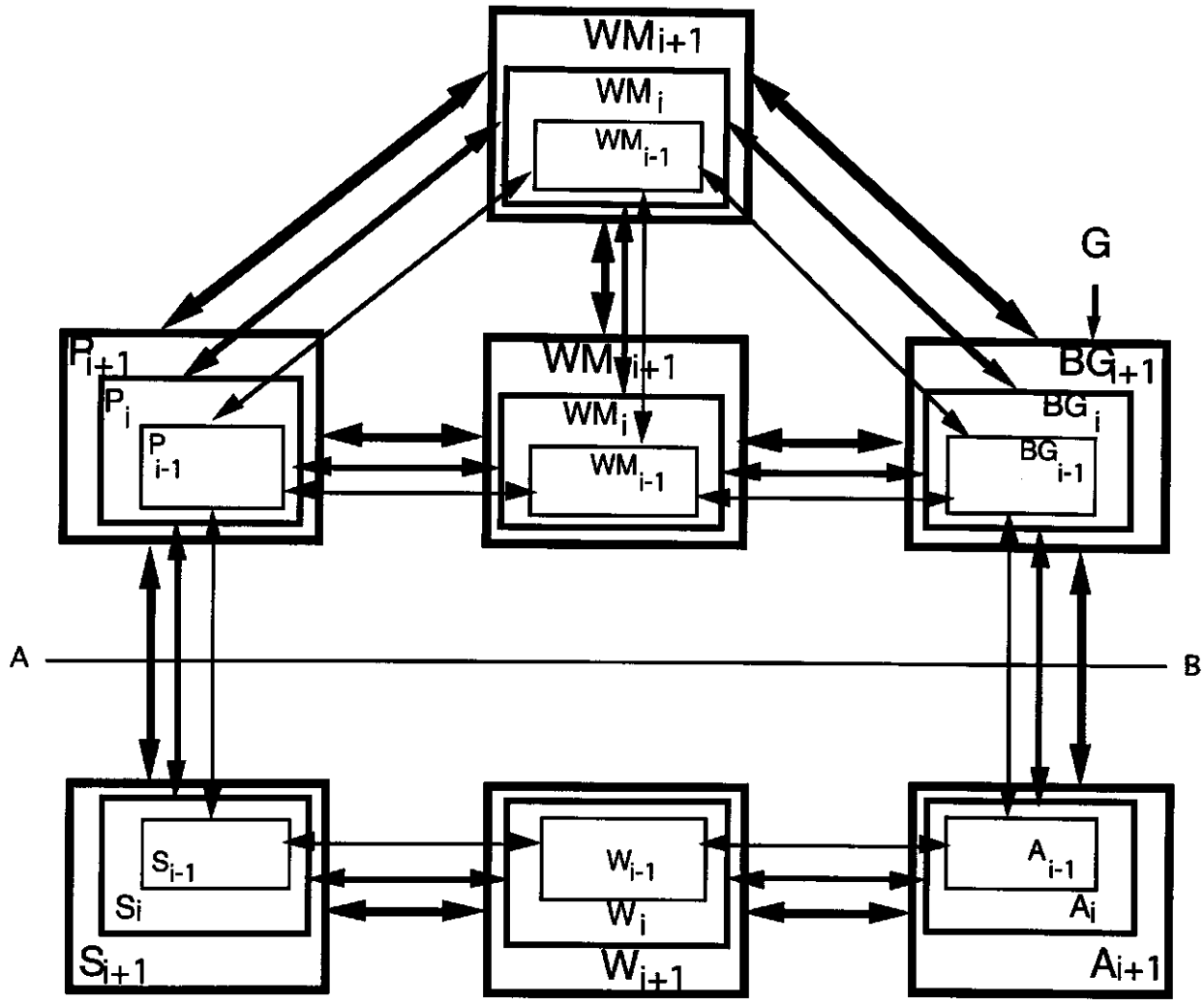


Figure 3-2 Nested ELFs

Notations: AB -the boundary between reality and its representation, G-goal, P-perception (or Sensory Processing, SP), WM-World Model, BG- behavior generation (or "planning/control"), A-actuators, W-world, S-sensors, VJ-value judgment

This description is related to each of the levels of the system. Each of them starts operating at its own resolution and accuracy. It is important to realize that for proper functioning of the i-th level, it should receive and submit information from and to its neighbor from below (i-1-th level), and should receive and submit information from and to its neighbor from above (i+1-th level). Each level contains knowledge (e.g. world models at its resolution). If a new model is required, it should be generalized upon the models which are held within the higher resolution levels. In Figure 2-2, the fact is shown that for a consistent functioning of the multiresolutional hierarchy, the direct link is to be provided. This demonstrates how the data are generalized bottom-up, and

how they are instantiated top-down within the couples $K_i \supset K_{(i-1)}$ for all modules of ELF.

The property of nesting implies that the levels of decomposition (aggregation) are also levels of different resolution (granularity, or scale). The nesting of ELFs implies that each i-th ELF nested within the (i+1)-th ELF is an ELF of a higher resolution (finer granularity, smaller scale).

The property of nesting is implied by the very nature of the recursive NIST-RCS hierarchy (see Subsection 3.1). The purpose of the hierarchy of control is to achieve the goal of the system by generating efficient behavior. To do this, the number of control activities is reduced by unifying elementary units of the system into generalized units. This leads to the following conditions called the *consistency conditions of nesting*:

1) The *goals* for the levels of hierarchy are nested within each other: each goal of the upper level contains goals of the lower level explicitly or implicitly. The goals of the higher resolution (HRGs) are represented within the lower resolution goals (LRGs) ultimately as their components. On the contrary, each LRG is a generalized form of all HRGs which are nested within them.

2) The *world models* for the levels of hierarchy are nested within each other. The world models of the higher resolution (HRWMs) are represented within the lower resolution world models (LRWMs) in a generalized form. Therefore, a set of HRWMs is nested within each LRWM as its components.

3) The *behaviors* for the levels of hierarchy are nested within each other. The behaviors of the higher resolution (HRBs) are represented within the lower resolution behaviors (LRBs) in a generalized form. Therefore, a set of HRBs is nested within each LRB as its components. This leads to the similar statements concerned with components of the behaviors: plans, control, and actions.

3*) The *plans* for the level of hierarchy are nested within each another. The plans of the higher resolution (HRPLs) are represented within the lower resolution plans (LRPLs), ultimately as their components. On the other hand, each LRPL is a generalized form of all HRPLs which are nested within them.

3**) The *actions* for the level of hierarchy are nested within each other. The actions of higher resolution (HRAs) are represented within the lower resolution actions (LRAs) in a generalized form. Therefore, a set of HRAs is nested within each LRA as its components.

This suggests that:

a) Any process of the World at i-th level of resolution directly consists of the processes of the (i-1)-th resolution level which, in turn, contain all processes of the (i-2)-nd level. Both systems and the processes in them can be decomposed in resolution levels.

b) Virtual sensor module of the i-th level can be regarded as a generalization of several real sensors at the (i-1)th level. Thus, these sensors of the (i-1)th level are directly nested within the virtual sensor of the i-th level. Sometimes, a rough sensor can be installed (physically) at a lower

level of resolution. Then, it can be regarded as a generalization upon nonexistent (virtual) high resolution sensors.

c) The same consideration (b) can be repeated for the actuators. Virtual Actuator of the i-th level ("motion producer") can be a generalization of the real actuators performing several motions at the (i-1)th level. The A_i module should contain a direct representation of the set of the $\{A_{i-1}\}$ modules[32], contain all representations of A at all subsequent resolution levels.

d) World Model of the i-th level should contain elements of the world models of each of the several units of the (i-1)-th level (in a generalized form). The values of parameters of these components of the overall model are constantly updated. This is why the set of $\{WM_{j,i-1}; j=1,...\}$ is directly nested within the WM_i.

e) All operations of ELF should be performed taking in account the property of nesting in a multiresolutional model. For example, the BG_i module performs planning. Any planning presumes contemplation of events to happen in the future and, therefore, contains simulations as a component. To judge the consistency of the results of planning at the i-th level, a condition should be checked to assure that the plans of the (i-1)-th level modules are contained within the best alternative of the plan at the i-th level. The operation of BG_{i-1} should be contained within the operation of BG_i.

f) The same should be repeated for the subsystem of SP_i. Recognition processes at the i-th level require permanent interaction with recognition processes of the (i-1)-th level. Most of the existing recognition processes (e.g. in the area of computer vision) follow this recommendation *ad hoc*. In many cases this recommendation is neglected or even contradicted.

3.3 The Nested Modules

All nested modules are expected to interact. Let us focus attention on this phenomenon of interaction. The concept of level which works as a provisional idea for describing the hierarchy is incomplete because it obscures the functional essence of the system. The processes of functioning can be described only if we consider a loop in which the totality of ELF-diagram processes can be recognized, and all causal links recorded and explained. This includes all modules in which it happens: i.e. sensors, perception, knowledge base, decision making, and actuators as this set of subsystems is interacting with the world. Discussion of the "level" of resolution, or hierarchical architecture, actually refers to the "loop" of the ELF-diagram which can be substantiated at this level. These levels are nested in each other exactly in the same manner as the modules are nested in each other.

This phenomenon of a "nested module" should be discussed in more detail. Consider a module (WM, or BG, or A) at a particular level of resolution; associated with this module are

[32] All of these modules are recursive ones which are explained later.

the models of the System, the snapshots of the world, the Plans, and the actual trajectories of motion will be described with a particular resolution. This means that all of the higher resolution levels of information are supposed to be fully represented within each of the boxes of interest. In this section, we are interested primarily in the following part of ELF (see Figure 3-4):

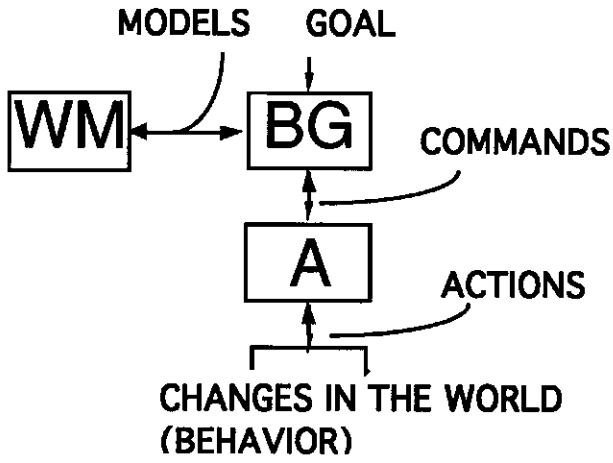


Figure 3-4 A Fragment of the Elementary Loop of Functioning
Notations: WM-world model, BG -behavior generation, A-actuator

The previous section showed that the level of behavior generation can be maintained. Then, the other levels can be recursively derived from the description of a single level. We consider a Loop of the Level to be the unit that performs the original G. Saridis' triplet of the intelligent operations: organization, coordination, and execution[33] [87]. Analysis shows that each of these three functions is being performed *at each level of resolution*, within ELF of this level, rather than being distributed top-down over the entire architecture of the system in the following manner: upper levels for organization only, middle ones for coordination only, and lower levels for execution only.

As we decompose the subsystems into next level sub-subsystems we immediately arrive at the phenomenon of nesting. If the subsystem can be decomposed, the sub-subsystems are the inner parts of it. To describe the functioning of the system, one should refer to the functioning of the sub-subsystems. Then, what is the purpose of having this separate consideration of subsystems and their parts? It is difficult to talk about functioning of a subsystem if the reference to all its sub-subsystems-components is required. The property of nestedness (which is a direct result of decomposition/aggregation) allows us to simplify

[33] Each level of resolution in a multiresolutional system has among its functions all these three activities: organization (possibly, planning, or feedforward control), coordination (job assignment to multiple agents and scheduling with communication among the negotiating agents), and execution (formation of the commands corrected by the results of monitoring, or feedback control). This triplet can be considered a crisp description of the operator of behavior generation.

representation, information channels, and communication.

Let us make an experiment, and decompose all modules of the system shown in Figure 3-4. Figure 3-5 shows the decomposition of all the modules in the system, which is extremely cumbersome. It demands demonstration of a multiplicity of loops of functioning, and the complexity of this effort rapidly grows. Compare it with the compact representation that is achievable if the concept of nestedness is employed as shown in Figure 3-2. It becomes especially difficult to retain all information about inclusions $K_{si} \supset K_{s(i-1)}$ for all modules of ELFs.

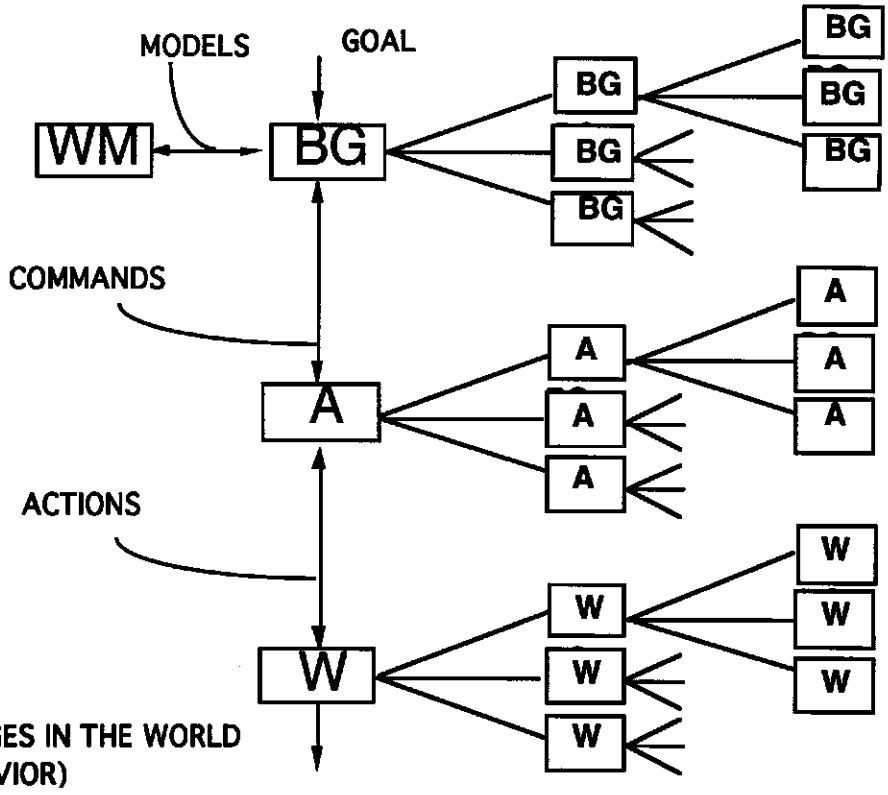


Figure 3-5. The same fragment (see Figure 3-4) with the hierarchical decomposition shown

Instead of dealing with a cumbersome hierarchy shown in Figure 3-5, we demonstrate that the relation of "belonging to" or "consisting of" can be substituted by using a kindred relation "nested in." This addresses an important issue of having all objects-results of decomposition and all processes of the subsystems to be components of the processes belonging to the systems which undergo a decomposition. In Figure 3-6, only one module of the next level below is shown that is nested within the module of the concrete level. In fact, branching is a typical phenomenon, and several modules can be nested in each module of the architecture.

We will refer to all these factors as measures for reduction of computational complexity.

Figures 3-5 and 3-6 demonstrate the fact that a high resolution level information is

contained at the lower resolution level models and descriptions of the processes. This condition is the condition of consistency. If this condition does not hold, any operation produced by the system will be erroneous. This is important because the conditions of nesting might be different in different parts of the process. The latter can even include the words of the vocabulary.

It is important that the concept of nesting be clear on the scale of the overall system (see Figure 3-6): all levels of the hierarchy within each subsystem (K, BG, or others) form a *nested system.* This means that the operation of the inner box is a *part of functioning* of the external box and *not a separate operation.* Functioning of the lower resolution box (see Figure 3-5) cannot be separated from functioning of the higher resolution boxes which are parts of the lower resolution boxes as shown in Figure 3-6. Inner boxes present the *symbol grounding* for the external boxes, their *consistency check.* The "consistency check" is a mandatory component of any implementation of the NIST-RCS system. This check should be considered as a part of the communication process.

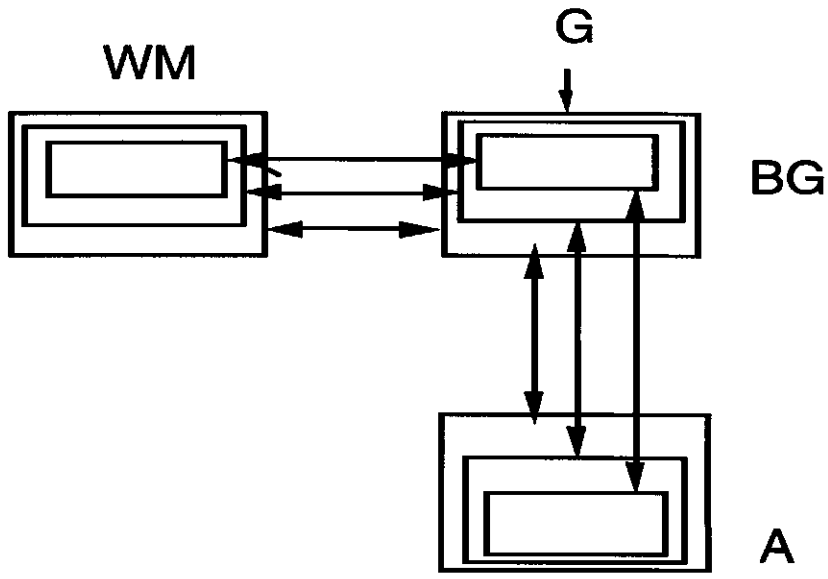


Figure 3-6. The Same fragment of the Elementary Loop of Functioning (see Figure 3-5) shown as a nested structure for three levels of the system

Each inner module is obtained from the external module as a result of refinement. Each inner box considers the processes of its corresponding external box at higher resolution with more detail, enhanced vocabulary, and smaller unit of the time scale. The totality of all processes of functioning of the inner modules is equivalent to the processes of functioning of the corresponding external module, but represented in this external module in the generalized form with lower resolution.

Change of the time scale should be interpreted as the fact that at higher spatial resolution, the processes should be considered that are faster than at lower resolution. This means that at

higher resolution the "frequency band" shifts toward the area of higher frequencies.

This phenomenon of "nested modules" allows for a better organization of the control system, for a convenient software organization and automatically guided procedures of NIST-RCS design.

3.4 What is automated and what is not

It does not matter! We discuss the NIST-RCS architecture which embodies the principles of BEHAVIOR GENERATION inferred from the abundant experience of human activities and the experience of engineering science. The format of architecture means that the jobs in the modules should be performed either by a human or by a machine depending on the available resources. The algorithms of operation and their arrangement should follow the arrangements delineated by the architecture regardless of the performer: a human, or a computer!

The general framework of the RCS architecture outlined in this Section, concerns rather the general laws of knowledge representation or general laws of symbolic representation of the World and its processes, than a set of particular applications. It is related rather to the general laws of thinking—and this is why the results of this Section are equally important at both the stage of systems design and the stage of analysis of system's functioning. The results can be related to both non-automated, as well as to automated systems including autonomous robots.

The roots of our discussion are situated theoretically in the area of Applied Semiotics—a discipline which focuses upon general laws of symbolic representation, organization, and processing of knowledge within a variety of its application domains [89-91].

4. The Overall Organization of Behavior Generation

4.1 The main concept of behavior generation

The main concept of behavior generation presumes that to achieve some goal (a state in the future) the controlled system should execute some behavior. The module of Behavior Generation consists of two submodules: for behavior planning and for behavior execution. These two sub-submodules are not merely postulated: they perform two major functions of a control system at a level which include feedforward and feedback control. The structure of BG-module follows from the definition of its function.

Let us consider a more informative version of Figure 3-4 (see Figure 4-1). This little segment of ELF (WM-BG-A-W) can be characterized by the following information flows between the subsystems of ELF:

- a goal is submitted externally, e.g. from the adjacent level of NIST-RCS above the level under consideration; the Goal is shown as a desired value of the output vector at the time T_{ph} which will be called later "the horizon of planning"

- some models are submitted from WM to BG; these models will be used to generate the required BG output

- based upon this goal, a set of commands (or a time function $U_v(t)$ of the command vector) should be generated at the output of BG; it is shown as a string (c_1, c_2, c_3, c_4)

- the changes in the world will be sampled by sensor functioning: the latter will generate a string of signals

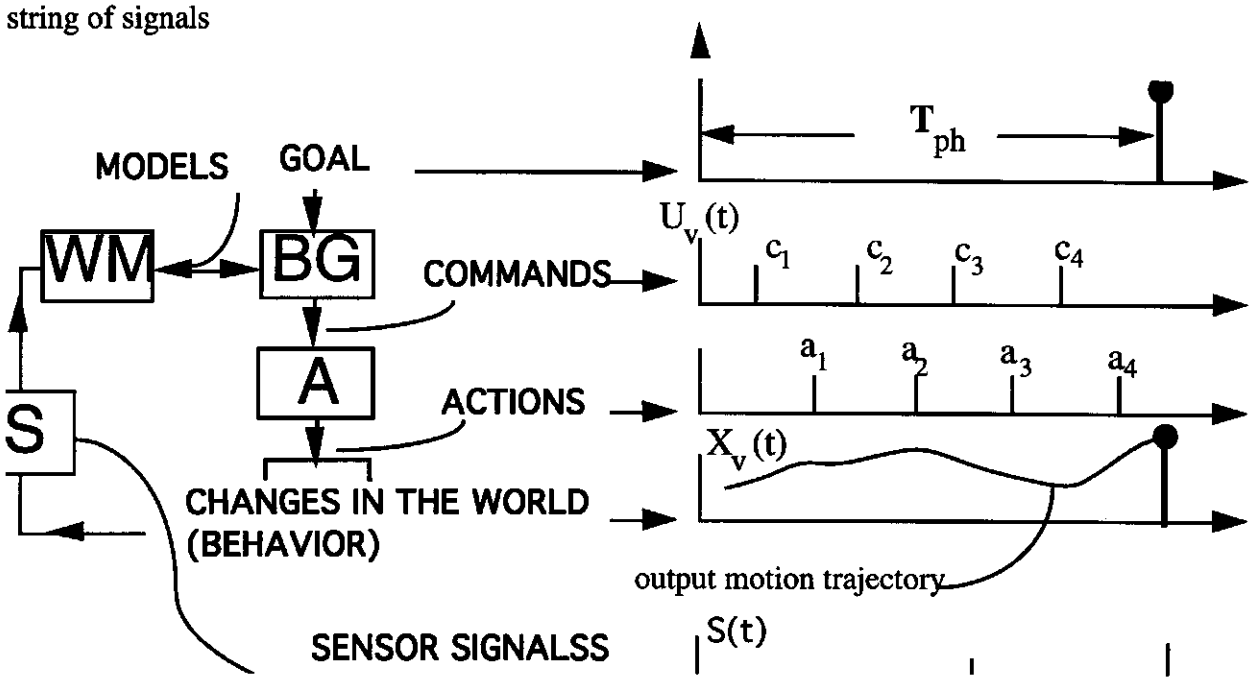


Figure 4-1 A detailed version of Figure 3-4

- as a result of this string of commands the set of actions (or a time function of the action vector) will be generated at the output of A; it is shown as a string (a_1, a_2, a_3, a_4)

- the output motion evolves within the world W and results in achieving the goal at the time Tph as was prescribed by the level above.

Even if all these variables are determined, the process of behavior generation cannot be considered finished. The following additional activities should be performed:

a) Because we are talking about virtual actuators and virtual world, the first command c_1, or a string of commands usually shorter than the full string of them determined by BG, should be submitted to the BG-module of the adjacent level below. A new behavior generation should be initiated with higher temporal and spatial resolution. This is separate from the different groups of components which exist within the output vector X_v and presume various virtual actuators listed at the level below. The commands should allow for distribution among the virtual actuators existing at the adjacent level below as shown in Figure 2-6.

b) The models obtained from WM have a limited accuracy. They occur even though they were accurate in the beginning. They become inaccurate in time because of yet unknown changes in the World. The string of commands computed is inaccurate. It will be quickly discovered that the virtual trajectory which is obtained as a result of functioning, differs from the virtual trajectory expected. It would be prudent to introduce corrections to the set of commands computed in expectation that these corrections will reduce the deviations from the output motion which are observed.

Two types of activities exist which are supposed to be performed by the behavior generation subsystem. The set of time functions should be determined, which allow for achieving the goal. They include the output motion trajectory, as well as the string of actions which produces this motion trajectory and the set of commands which generates these actions. This set of time functions will be called *a plan*. It consists of decisions about job distribution and scheduling together, it includes all package of coordinated schedules. It should allow for decomposition into components required for the virtual actuators of the adjacent level below. The process of finding these decomposable time functions will be called *planning* and the sub-subsystem that produces it will be called PLANNER.

Next, the process of transforming plans into the set of output commands to the adjacent level below will be called "execution". This process should include an opportunity to actively correct the plan online (compensate for the deviations, perform local predictions). The sub-subsystem performing this function is called *executor*.

A plan of the level should exist before any execution starts. This plan is considered a program of functioning. Then, measures are taken to make the executed behavior as close to the plan as possible so that the Goal could be achieved with the lowest cost possible. Cost is understood as the value of losses that is required to achieve the Goal. This should include losses

of energy, losses of time, expenses linked with manpower, cost of the tools and material, wear of the devices required. Goal is presumed to be generated externally[38], and the Goal entails the substance of cost functions to be applied in each particular case.

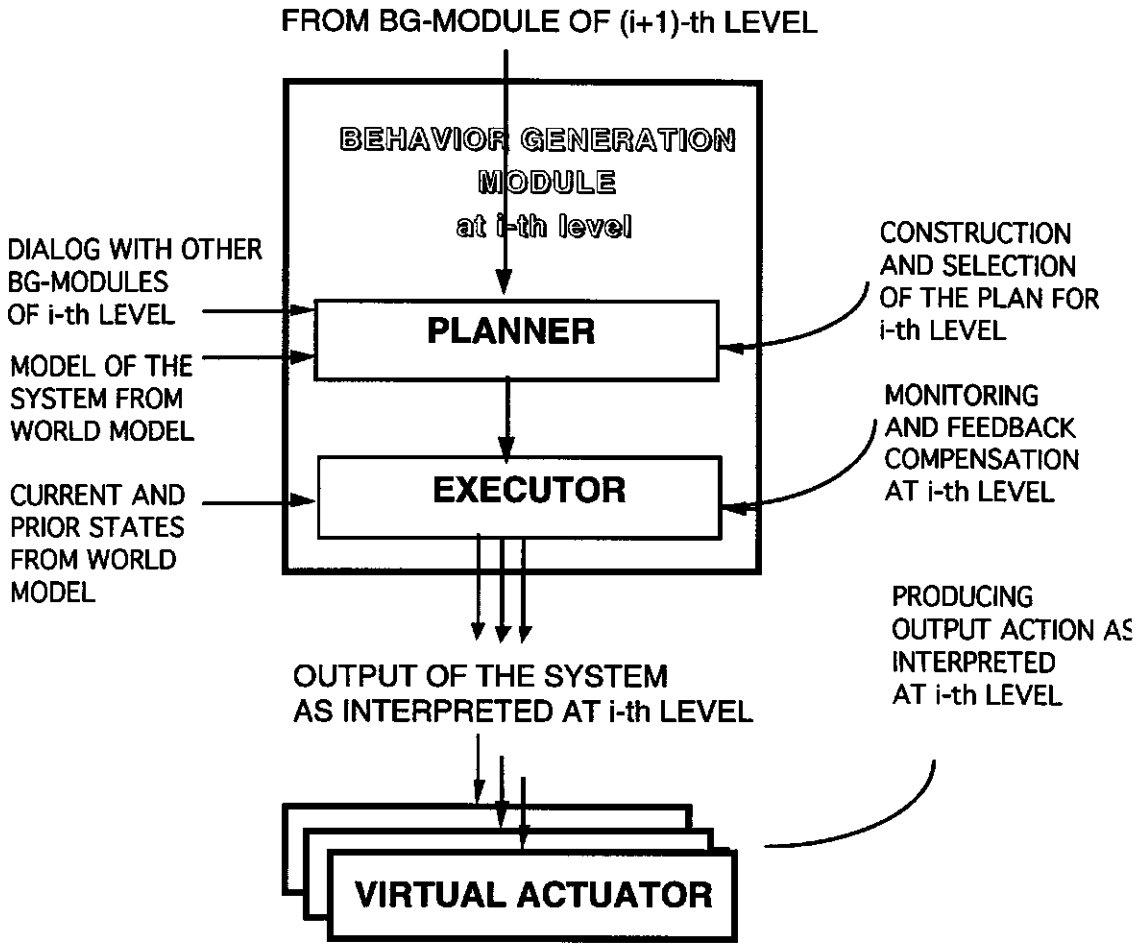


Figure 4-2. Behavior Generating Module of the i-th Level

It is a part of the Loop of Functioning determined for the i-th level. The output of BG_i is submitted to the Virtual Actuators as determined for this level.

Each ELF uses its BG to feed the Actuators (A) with its output. However, the output of A arrives to the world (W), where the motion is supposed to occur. This transition from the input commands to the motion of the World is the most fundamental process in the

[38] i.e. the lower level of resolution submits the Goal to our level, but it also received its Goal from even lower level of resolution; if this chain is not endless the highest level of resolution is presumed where the very first Goal was created—we do not know how, we do not ask this question in this book; the mystery of fully autonomous system is temporarily out of our scope and concern.

system because this is the main resource-consuming process where the resource can be interpreted as time, energy, or human power, etc. The need in consumption of these resources can be explained by the resistance to the action to be produced. This is why in Figure 3-1 we allocated a separate coordinate system for the action and the output trajectory. Actuators have commands as the input, actions as the output, and the desired motion in the external world as a result of this output[39].

The input to virtual actuators demonstrated in Figure 4-1 is an input to the BG-module of the adjacent level below. However, BG-module of the i-th level does not realize that the command is submitted to the BG-module of the (i-1)-th level. EXECUTOR of the i-th level does what it is supposed to do. It submits its output to the actuator. It has the parameters of this actuator (the set of virtual actuator parameters) and receives the information about the execution process at this virtual level.

4.2 Realistic examples of behavior generation

At some level of aggregation this is a turning machine tool that cuts the piece of metal to properly shape it, which is our goal. The relative motion of the cutter and the piece of metal is the output motion which leads us to the Goal (consider this to be Example 1). One can interpret Actuator at another level of resolution as assembly activities where different parts are to be put together. Then, the ordered attachment of the parts will be the desired motion, and the assembled object is the Goal (consider this Example 2). Finally, one can interpret Actuator as a Manufacturing Shop which gives as output several sets of manufactured parts according to a time schedule. The Goal is to have these parts at the output of this Shop according to the required schedule (consider this Example 3). In all three examples, the Goals and the output motions are different from the input commands introduced at the input of Actuators.

• **In Example 1**, the Goal is the trajectory of relative motion of the blank[40] and the cutter. Several alternatives can be considered typical for the metalcutting machines: the spindle and the turret, or the cross feed carriage (in the turning machine), the mill and the table (in the milling machine). This relative motion is a shape forming factor. The spindle rotates with a particular speed, which is determined by a spindle motor equipped with mechanical devices. This motor should receive a command "on" in the beginning and "off" in the end, which is the feedforward control, FFC: it can be interpreted as feedforward schedule of commands. Because the blank metal resists cutting, resistance forces emerge on the cutting surface.

The force developed by the actuator should overcome the resistance force. As a result of these forces, deformations emerge: the blank bends and the cutter wears. The torque on the shaft

[39] The term "motion" is understood in a general sense: as a set of time-tagged states (or vectors).

[40] Blank - a piece of material prepared to be made into something by a further operation, e.g. cutting.

of each electrical motor emerges, the current in its windings grows. The first order differential equation of the electrical circuit equilibrium for dc motor under load is written as follows:

$$u(t)—e(t)=i(t)+L\frac{di}{dt} \tag{4-1}$$

where u(t)-is voltage applied

e(t)-is value of the counter-EMF equal to $k_1\omega$

which gives $$e(t)=k_1\omega \tag{4–2}$$

k_1-is a constant value

ω(t)-is velocity measured on the shaft of the motor.

The first order differential equation also holds for the equilibrium of mechanical system:

$$T_m(t)—T_c(t)=J\frac{d\omega}{dt} \tag{4-3}$$

where T_m(t)-is the value of torque which is developed by the motor as a result of current i(t) so that

$$T_m(t)=k_2i(t) \tag{4-4}$$

k_1-is a constant value

J-is the value of inertia on the shaft.

T_c(t)-is the value of torque developed by the blank resisting to the process of cutting.

The set of equations (4-2)-(4-4) is obtained for the spindle actuator. Similar system of four equations can be written for the feed actuator.

The value of torques T_{cs}(t) and T_{cf}(t) for the spindle (s) and feed (f) actuators correspondingly depends on many variables such as the depth of cutting (d), the type of the cutter (c), the velocity of spindle actuator ω_s(t) and the velocity of the feed actuator ω_f(t). The World can be characterized by the function F of cutting process which can be written in the form of an equation of equilibrium of cutting. The possible form of this equation is illustrated by the following expression

$$F[d, c, T_{cs}(t), \frac{dT_{cs}(t)}{dt}, T_{fs}(t), \frac{dT_{fs}(t)}{dt}, \omega_c(t), \frac{d\omega_c(t)}{dt}, \omega_f(t), \frac{d\omega_f(t)}{dt}]=0 \quad (4\text{-}5)$$

The system of equations (4-1)-(4-5) has an infinite number of possible solutions. In order to perform planning, a set of constraints should be added and a condition demanding to minimize (or constrain) the cost function. The cost function C for the electrical motor can be interpreted as time, energy, materials, etc., separately or in a combination:

$$C\text{-->}\min \quad (4\text{-}6)$$

To find the process of output motion, these equations (4-1)-(4-6) should be solved jointly or simulated. The simulation is to be organized so that the overall effort could be distributed between the actuators.

We have intentionally selected the simplest case with only two equations which are linear and first order. In reality, we have more equations which are often nonlinear and have higher order. But, three requirements remain unchanged in all cases: the equation of equilibrium of processes in the Actuator should hold, the equation of equilibrium of processes in the World should hold, and the condition of cost function minimization (or constrain) should be added.

By solving the equations under different conditions, or by simulating the process of cutting under different conditions, the process of search is actually performed. This allows us to determine the command sequence called the feedforward control, that is supposed to generate a "reference trajectory" for subsequent tracking. When applying this control, the motion deviates from the required (reference) trajectory unless the speed (and/or torque) is measured and its deviations are compensated by proper varying of the input voltage (feedback compensation control, FBC).

Note that during the planning process we have determined the time functions for all components of the plan at once: the output motion, the current trajectories (actions), and the set of control commands. In this comparatively simple problem this was possible.

Electrical motor represented by equations (4-1)-(4-6) is a virtual one: all parameters and dependencies are generalized and hide behind them a layer of higher resolution. This does not violate the generality of approach we have introduced here. We can envision that after this stage of planning is performed, the actuators we dealt with could turn out to be virtual actuators. The next stage of planning (with higher resolution) should be performed when the next decomposition is done. Indeed, the turret should follow the prescribed trajectory. The latter is performed in a plane by motion of 2 electrical motors. Their trajectories are assigned by a constantly varying voltage (feedforward control, FFC), and the trajectories deviations should be compensated for (feedback compensation control, FBC).

• **In Example 2**, the Goal is to manufacture an object by assembling it from a set of parts.

The output trajectory is a best set of the parallel/sequential strings of activities that are required to achieve the "assembly" of the output trajectory with minimum cost (the output schedule).The input vector trajectory is contained within the schedule of commands which prescribes the activities to be performed.

Several alternatives that relate to the assembly operation can be considered: manual assembly, robotic assembly, partially automated assembly, etc. Within each of these alternatives, the relative motion of the parts is to be performed so that they match and attach to each other. Although the rules of attachment are prescribed by the drawings of the assembly, the relative motion can be performed in many ways. The fitting couples should be determined and attached to each other if it does not contradict the subsequent formation of the more complex groups of parts.

Then, the performing actuators should receive proper commands in the order determined by the plan of assembly. The plan can be interpreted as a feedforward control, FFC; it consists of the reference trajectory to be tracked at the output and the schedule of commands to be applied at the input. Each of the relative motions linked with the search of matching positions and subsequent fastening procedures requires maneuvering the parts in their relative motion. The process resists our effort to speed it up in the same way as the difficulty increases of maneuvering the moving device in the cluttered environment when we try to increase the speed of its motion. For example, because of the inertia one has to spend more energy in avoiding the collision, and the accuracy of motion reduces which leads to increase of time of machining instead of its reduction.

The actuator force should overcome the force required for quick maneuvering. As a result of these forces, motion errors emerge. The oscillatory component grows in the torque on the shaft of each electrical motor, heating of its windings grows. Considering that the assembly is performed by a single gripper equipped by three (X,Y, Z) actuators. First order differential equations of the electrical circuit equilibrium for dc motor under load is written as follows:

$$\{[u(t)-e(t)=i(t)+L\frac{di}{dt}],[e(t)=k_1 w]\}_j \ , \quad j=1, 2, 3 \qquad (4\text{-}7,4\text{-}8)$$

where j is the number of the actuator.

The first order differential equation holds also for the equilibrium of mechanical system:

$$\{[T_m(t)-T_{fr}(t)=J\frac{d\omega}{dt}],[T_m(t)=k_2 i(t)]\}_i \ , \quad i=1, 2, 3 \qquad (4\text{-}9,4\text{-}10)$$

where $T_{fr}(t)$-is the value of torque which is developed by the friction and weight of the part.

The set of equations (4-7)-(4-10) is obtained for the X-actuator. Similar system of four equations can be written for the Y and Z actuators. Let us denote this additional set $(4\text{-}7)^i$-$(4\text{-}10)^i$. For the

overall set of actuators we will receive a set of equations (4-7)-(4-10)^i.

The World is characterized by the initial configuration of parts and the configuration space where all possible trajectories of motion can be found by consecutive *simulation of elementary moves to form pairs*, i. e. by *searching*.

The system of equations (4-7)-(4-10)^i has an infinite number of possible solutions. To perform simulation and/or search, a set of constraints should be added and an equation demanding to minimize (or constrain) the sum of all cost functions C_i , i=1, 2, . . . computed for all actuators involved. As in the previous case, it includes components that depend on time, energy, materials, etc., separately or combined:

$$\Sigma C_i \text{ -->min} \qquad (4\text{-}11)$$

To perform the search, these equations (4-1)-(4-11) should be jointly simulated: their analytical solution at each step of search is either not easily available or more expensive computationally than simulation. Organization of the simulation is to allow for the eventual distribution of the assembly operation between the actuators.

We have intentionally selected the simplest case with only two equations which are linear and first order. In reality we have more equations, they are often nonlinear and have higher order. But three things should hold: 1) the equation of equilibrium of processes in the Actuator, 2) the equation of equilibrium of processes in the World, and 3) the condition of cost function minimization (or constrain) which should be added.

Only for the simple assemblies, is this joint search-simulation computationally affordable. In most cases, the output motion is to be simulated without simultaneous simulation of the transient processes in the actuators. The search is performed by minimizing the cost function, which is artificially simplified by generalizing its components. After the output trajectories are planned, they are inverted to the input of the actuators by using different analytical or computational techniques.

The reality of the process of assembly differs from our expectations, and feedback compensation control is required. Indeed, as we apply the chosen plan of assembly, the motion might deviate from the required trajectory unless the speed (and/or torque) is measured and its deviations are compensated by properly varying the input voltage (feedback compensation control, FBC). These corrections might lead to a need to replan because under new circumstances, better results could be obtained as a result of repeating the process of search from an intermediate stage of the assembly.

• **In the Example 3**, the Goal coincides with the output motion; it is the output schedule of sets of manufactured parts. The only difference that might be envisioned is that: the goal might include the constraint on expenses allowed. The Goal will be achieved if the input schedule of assigning these parts to the particular machines and manpower is determined. Among the

multiplicity of possible time and space assignments, there should be a set of preferable assignments which provide the output schedule required with the best or assigned losses.

The output motion trajectory can be obtained if a best set of the parallel/sequential string of activities is assigned to the virtual actuators (cells, and/or machines, together with a particular manpower). Then, the goal will be achieved with minimum losses (the output schedule).The input vector trajectory is the input schedule of commands that allows these activities to be performed. An activity is understood as an assignment of a particular part to a particular machine operated by a particular person. Information is available about productivity of the machines with particular parts to be manufactured (sometimes the productivity is affected by a particular worker doing the job). Also, as in the case with assembly, a rule base that contains a set of constraints that demand particular sequencing of jobs (precedence rules) is available .

Several alternatives can be considered related to manufacturing shop functioning: prior grouping of the machines, parts, and manpower which would provide reduction of the expected volume of search during the process of planning. Proper grouping creates intermediate levels of hierarchy in manufacturing so that in distributing part groups among the machines, general matches can be attained. This can ensure that more general cost functions are minimized. Although the rules of precedence are prescribed, the assignment can be performed in a vast multiplicity of ways. These ways of improving the process of planning become more complicated because the number of cost functions is usually large and not properly ordered.

The role of actuators is played by the machines with the manpower distributed among them. The World can be characterized by the initial configuration of machines, and the totality of parts to be manufactured. All possible trajectories of motion can be found by consecutive *simulation of elementary assignments to form non-contradictory strings*, i. e. by *searching* in the configuration space. The cost function C which can be interpreted separately or as combination of time, energy, materials, etc. The search is to be organized so that the overall effort allows for the eventual distribution of the assembly operation between the actuators.

The reality of the Manufacturing Shop can differ from our planning expectations. The feedback compensation control should be introduced as the schedule corrections.

4.3 Generalization upon realistic examples: a sketch of the theory

We can see that after our image of the output motion in the World has been found, as a result of the search, it should be inverted to the input of the Actuator to determine the required commands.

At each level of resolution of the NIST-RCS hierarchy, the image of the virtual output motion should be determined (the motion $X_V(t)$ in the virtual World, W_V). Then this motion should be inverted to the input of the virtual Actuator (A_V) and the function $F(u_V)$ of the virtual

input control should be computed. If the virtual Actuator has a transform function T_{av} then the following holds for the input trajectory $U_v(t)$:

$$U_v(t)=X_v(t)*T_{av}^{-1} \qquad (4\text{-}12)$$

With given T_{av}^{-1}, we worry about receiving both the desired virtual output motion $F(u_v)$ and the required input control commands $X_v(t)$. These two functions in most cases are not the same. Goal arrives to the level of NIST-RCS in a form of the set of states $\{X_{v1}, X_{v2}, ..., X_{vn}\}$ which should be achieved at the moments of time $t_1, t_2, ..., t_n$ where "n" is the "horizon of planning" and the minimum difference between two consecutive milestones is never less than Δt—the time-scale interval at the level. We can see that the Goal at a level is assigned as a set of the time-tagged milestones of the output trajectory (of the motion) within the state space.

If this trajectory is found in the form $X_v(t)$, and the properties of the system are known in the form of transformation function T_{av}, the input trajectory (program of control) can be found $U_v(t)$ by inverting the output to the input terminals of the system to be controlled. (Eventually, $U_v(t)$ should be transformed into the discrete string of commands $\{U_{v1}, U_{v2}, ..., U_{vn}\}$; the minimum difference in time should be equal Δt— the time-scale interval at the level.) Therefore, *plan* can be understood as a symbolic and/or numerical description for a couple $\{X_v(t), U_v(t)\}$ which includes both the desirable (output) behavior of the system, and the input control trajectory which is required to obtain this output trajectory.

Consequently, plan can be found by performing two steps of operation:

Step 1. Given the output milestones, simulate the processes and search for possible combinations of the output functioning which form the admissible set of virtual output trajectories of motion.

Step 2. Find the input commands to be submitted to the virtual actuator at the level.

In some realistic cases, Step 1 can be skipped over, and only Step 2 is required. Often $T_{av}=I$ and to find output is the same as to find the input. Other variations are also possible, however, the complete and adequate description of the operations required for planning includes both steps of the operation. However, in most of the NIST-RCS systems, finding the desirable output functioning is the main problem. This problem is solved by using the main NIST-RCS algorithm of planning (see sub-subsection 4.4).

We tag the output trajectory of the motion and the input commands that are supposed to produce this motion in elementary units of time pertaining to the particular level of resolution under consideration.

It is typical to decompose a schedule into a consecutive set of milestones (intermediate goals) on the way to the final goal. Task decomposition transforms the goal of the system into a hierarchy of the subgoals (which are the goals for the subsystems). The subgoals contain the set of

assignments including determining inputs to the subsystems, feedback control laws and gains of the compensation controller.

In this subsection, we have described how the goal state stimulates planning the output behavior which generates the goal-states for subsystems. In the same way, the initial input goal submitted to our system has emerged as a result of BG-process at a level above (a lower resolution level.) This concept has a broad basis and many examples can be given that in practice goal-creation and behavior generation happen in such a way.

4.4 Algorithm of Multiresolutional Hierarchical Planning (NIST-RCS PLANNER)

Let us consider a space Ω_i in which the World Model (WM) is represented at a particular level i with resolution ρ_i (for both space and time). From the (i-1)-th level, the goal

$$G_i(T_i,\ SP_i,\ FP_i,\ J_{i+1,1}<J_{i+1}<J_{i+1,2},\ J_{i,1}<J_i<J_{i,2}) \qquad (4\text{-}13)$$

arrives that demands for the task T_i to be performed having the start and final points SP_i and FP_i given as a part of this task. It contains a description of job to be done which can be performed at the i-th level of NIST-RCS given its particular set of virtual actuators. The motion trajectory from SP_i to FP_i is to be be found with the value of final accuracy ρ_i and the cost constraints of the upper level $J_{i+1,1}<J_{i+1}<J_{i+1,2}$ are added to the cost constraints of the i-th level $J_{i,1}<J_i<J_{i,2}$. The condition of job constraints for the i-th level can be substituted by the condition of minimizing some of the components of the J_i vector.

We will introduce the following three operators.

I. Operator of Job Assignment (**JA**) which determines the virtual actuators of the (i+1)-th level of resolution to be involved in behavior generation. This determines groups of coordinates in which Jobs to be performed are represented at the particular i-th level of resolution. **JA** assigns coordinates to these groups of Job labels. In other words, **JA** operator introduces the alternatives of spaces in which the output motion should be described at this level of resolution in order to choose one of them. This operator maps the task (T_{i+1}) that has arrived from the (i+1)-th level of NIST-RCS into the space of output covered by the virtual actuators of the i-th level of NIST-RCS. Computationally, this is done via combinatorial mapping from the set of tasks into the set of virtual actuators under constraints introduced from the prior experiences.

$$\mathbf{JA}:(T_{i+1},\ \Omega_i,\ \rho_i) \rightarrow \{\{A_v\}_{i,j}\}_k,\ \text{or}\ \{A_v\}_{i,j}=\mathbf{JA}(T_{i+1},\ \Omega_i,\ \rho_i), \qquad (4\text{-}14)$$

$j=1, 2, \ldots, n;\ k=1,\ldots, m$

where **JA** - is the combinatorial grouping operator that performs synthesis of the groups to perform the job assignment. It does it by mapping from the task description into a possible combination of the coordinates of the state-space Ω_i at the resolution ρ_i.

$\{\{A_v\}_{i,j}\}_k$ - is the set of meaningful distributions of the task among the n virtual actuators of the i-th level of resolution, j is the number of an actuator, and k is the number of alternatives of job assignment,

The result of this search gives a map which determines the density of the subsequent search-graph. This map will be called "Alternatives of job assignment" and the best time-schedule should be computed for each of these alternatives.

II. Operator of state space search for the admissible set of time-schedules of the output trajectory (**SC**). This operator provides for a combinatorial grouping of the points of the output space into a set of strings for the subsequent consideration of these strings as alternatives of the output trajectory. The input trajectory can also be $U_v(t)$ obtained during the process of planning (scheduling). Otherwise, it a separate computation (4-12) can be performed to find the input commands from the description of the required output results.

$$\mathbf{SC}: (\{\{A_v\}_{i,j}\}_k, \text{SP, FP, J}) \rightarrow \{X(t)\}_l \text{ , or } \{X(t)\}_l = \mathbf{SC}(\{\{A_v\}_{i,j}\}_l, \text{SP, FP, J}) \quad (4\text{-}15)$$

where **SC** -is the string concatenating, or scheduling operator

l=1, 2, ..., a is the number of alternatives of the time-schedules for the output trajectory retained for the subsequent comparison,

$\{X(t)\}_l$ - is the set of admissible strings (output motion trajectories) connecting the start point SP and the finish point FP and providing the string belonging to an interval delineated by the cost constraints of the upper level $J_{i+1,1}<J_{i+1}<J_{i+1,2}$ added to the cost constraints of the i-th level $J_{i,1}<J_i<J_{i,2}$.correspondingly, where indices 1 and 2 denote the lower and upper bounds of the desirable cost.

III. Operator of selection and focusing attention (**PS**), which determines the only trajectory (the "best" one), together with the the subset of the output space (the latter is supposed to be considered at the higher level of resolution as the subset of the output space for the particular solution refinement)

$$\mathbf{PS}:(\{X(t)\}_l \{J_p\}) \rightarrow \text{ENV}_{i-1}(X^*_i, w), \text{ or } \text{ENV}_{i-1}(X^*_i) = \mathbf{PS}(\{X(t)\}_l \{J_p\}), \quad (4\text{-}16)$$

where $\{X(t)\}_l$ -a set of all admissible output trajectories,

X^*_i - the best output trajectory (minimizing the sum of costs under consideration)

$\{J_p\}$- the totality of all cost measures including not only those mentioned above, but also the results of simulation if the latter are available, w- is the parameter of the envelope, (e.g. the "width" of the envelope), ENV_{i-1}-is the result of surrounding the best trajectory by a zone of the space called "envelope" for the subsequent search procedures at the next (i-1)-st level of resolution.

Joint functioning of these three operators gives the algorithm of planning: PLANNER=JA*SC*PS. Clearly, by selection of the X^*_i, we simultaneously determine the best schedule, the required set of actuators, the optimum set of inputs. PLANNER=JA*SC*PS is equivalent to a single operator of multiresolutional searching for the best output trajectory.

The multiresolutional search for the best output trajectory can be concisely described as follows: for k=1, ..., m do the following string of procedures:

a) $\mathbf{JA}(T_{i+1}, ENV_{i-1}(X^*_i), \rho_i)$,

b) $\mathbf{SC}(\{\{A_v\}_{i,j}\}_l, SP, FP, J)$,

c) $\mathbf{PS}(\{X(t)\}_l \{J_p\})$

The algorithm of control can be represented as a diagram

$$\begin{array}{l} T_{i+1},\rho_i \quad SP, FP, J \quad \{J_p\} \\ \downarrow \qquad \downarrow \qquad \downarrow \\ ENV_i(X^*_{i+1}, w_{i+1}) \rightarrow \mathbf{JA} \longrightarrow \mathbf{SC} \longrightarrow \mathbf{PS} \longrightarrow ENV_{i-1}(X^*_i, w_{i+1}) \\ \qquad\qquad \{\{A_v\}_{i,j}\}_l \quad \{X(t)\}_l \end{array} \qquad (4\text{-}17)$$

or a recursive expression

$$ENV_{i-1}(X^*_i) = \mathbf{PS}\left(\mathbf{SC}\left(\mathbf{JA}\left(ENV_i(X^*_{i+1}), w\right), \rho_k\right) SP, FP, J\right) \qquad (4\text{-}18)$$

The forms (4-17) or (4-18) can be considered the NIST-RCS Algorithm of planning for BG. Planning is not sufficient for Behavior Generation: feedforward control should be supplemented by feedback compensation. Thus, PLANNER in BG-module is later supplemented by a feedback compensation sub-subsystem called EXECUTOR.

4.5 BG-Module: An Overview

BG-module is considered in two interrelated aspects: as a part of the overall generic processing node of the NIST-RCS, and internally as a *generic behavior generating module*.

4.5.1 A Generic Processing Node[41]

From Sections 1 and 2 we learned that all systems allow for representation in the form of ELF-hierarchies, which can be considered as systems of nested ELFs (see Figure 1-1). A couple of additional comments can be made now in the view of our close interest to the processes within BG-module. In the set of ELF's subsystems we are especially interested in its processing part: SP-WM-(VJ)-BG[42] . We will call this part of an ELF, a Generic Processing Node (of NIST-RCS.)

The relationships and interactions between the BG, WM, SP, VJ, and KD modules in a generic node of the NIST-RCS architecture are shown in Figure 3-1 as a part of the overall ELF. Now we will discuss these subsystems in more detail.

The Sensory Processing (SP) module contains filtering, detecting, and estimating algorithms, plus mechanisms for comparing predictions generated by the WM module with observations from sensors. The SP has algorithms for recognizing entities and clustering entities into higher level entities. The Value Judgment (VJ) module evaluates plans and computes confidence factors based on the variance between observed and predicted sensory input. SP communicates actively with WM (and its knowledge storage) in order to be able to hypothesize about clusters detected and to make converging the process of multiresolutional image recognition[43].

The World Modeling (WM) module contains, or has an access to, the Knowledge/Database (KD), with both long-term and short-term symbolic representations and short-term iconic images[44]. In addition, WM contains a Simulator for testing the hypothesis that emerges during the process of knowledge organization. This simulator is employed by BG for testing the alternatives generated by JA and SC search algorithms for the subsequent comparison in the Plan Selector (PS).

Finally, the Behavior Generating (BG) module contains submodules of Planner (PL) and Executor (EX) as shown in Figure 3-2. According to sub-subsection 3.4, Planner has sub-submodules of Job Assignment (JA), Scheduling (SC) and Plan Selector (PS). Before the final selection of the best PLAN happens, Planner sends the alternatives to WM for modeling of the alternatives within a larger picture. One can argue whether or not Planner should send the alternatives for simulation back to the WM-module, or it should request the model from WM and

---

[41] Most of this subsection is an excerpt from [91].

[42] VJ is put in parentheses because in different discussions, it is beneficial to consider a separate Value Judgment module, or to distribute the function of Value Judgment among SP, WM, and BG.

[43] More details are to appear in a separate NIST report "RCS: Sensory Processing."

[44] The details of knowledge organization in the form of symbolic (conceptual) vs iconic domains of knowledge/data base will be given in a special report "RCS: World Modeling" which is now in progress.

perform Simulation within BG. It seems that in different systems this can be done in a different way.

Each node of the NIST-RCS hierarchy is a control system and it closes a control loop (ELF). Functioning of this ELF is demonstrated in Figure 3-3. Input from sensors is processed through sensory processing (SP) modules and used by the world modeling (WM) modules to update the knowledge database (KD). This provides a current best estimate (x-hat, see Figure 4-3) of the state of the world to be used as a feedback signal to the executor (EX) submodule. The EX submodule computes the compensation required to minimize the difference between the planned reference trajectory and the current state of the world.

The estimate x^ also is used by the JA and SC functions and by the WM plan simulator to perform their respective planning computations. The vector x^ is also used in generating a short term iconic image that forms the basis for comparison, recognition, and recursive estimation in the image domain in the sensory processing SP module.

The structure demonstrated in Figure 4-3 was built for a system when it is convenient to search for the alternatives of the schedule of commands by performing a specific search. First, introduce the alternatives of job assignment and then compute schedules for these particular alternatives of job assignment. We will later introduce a more general structure of a PLANNER to which the structure from Figure 4-3 is just a particular case (see Section 5). At this stage, we intend to discuss functioning a BG subsystem as a part of a generic node of the NIST-RCS system.

Figure 4-3 is presented with a level of detail which allows for applying it both for a single level of control (a single loop ELF), or for interpreting it as an arbitrary level of the multiresolutional hierarchy of NIST-RCS. In the latter case, the existence of lower resolution levels can be seen in the "Goal" arriving from above, while all lower levels are represented by an imaginary (virtual) controlled system .

4.5.2 Internal Description of BG-module

Let us start with description of full BG-module functioning within the ELF of a particular level. It is realistic to assume that we have a variety of sensors including those measuring values of some particular parameter (and using a transducer "physical variable-symbolic variable") and those dealing with visual images. It is convenient to deal with these two groups of sensors separately. Let us also assume that ^the WM-module is equipped with a rich knowledge database that allows for full support all sensory processing and decision making as well.

The module of Behavior Generation (BG-module) at each level of NIST-RCS hierarchy is performing such cognitive activities as construction of the alternatives of solutions in time and in space. For example, it suggests trajectories of the output motion, suggests schedules and or trajectories of the input commands, it performs selection of the preferable alternative of the plan,

BG-module communicates with World Model from which it receives the state information, the relevant mappings of the system to be controlled (computational model), and the results of simulation. It also communicates with all other BG-modules at its level of resolution.

A particular solution of the PLANNER shown in BG (Figure 4-3) consists of the sub-submodules JA, SC, and PS which operate as follows:

• JA forms tentative alternatives of distribution for the jobs and resources to subagents, and transforms coordinate systems from task to subtask coordinates (e.g. from end-point, or tool, coordinates to joint actuator coordinates.)

• SC computes a temporal schedule of subtasks for each alternative of job distribution contemplated by JA and coordinates the schedules between cooperating subagents (e.g. coordinate joint actuator trajectories to generate desired end-point trajectories.)

• PS accounts for the cost of the alternatives, for the results of their simulation by WM and evaluation of these results by VJ, and for the number of replannings in a particular situation. PS selects a subset of the plan for sending it to the higher resolution adjacent level of NIST-RCS hierarchy. Since JA and SC operate in the language of the next adjacent higher level of resolution, *the task turns out already to be decomposed.*

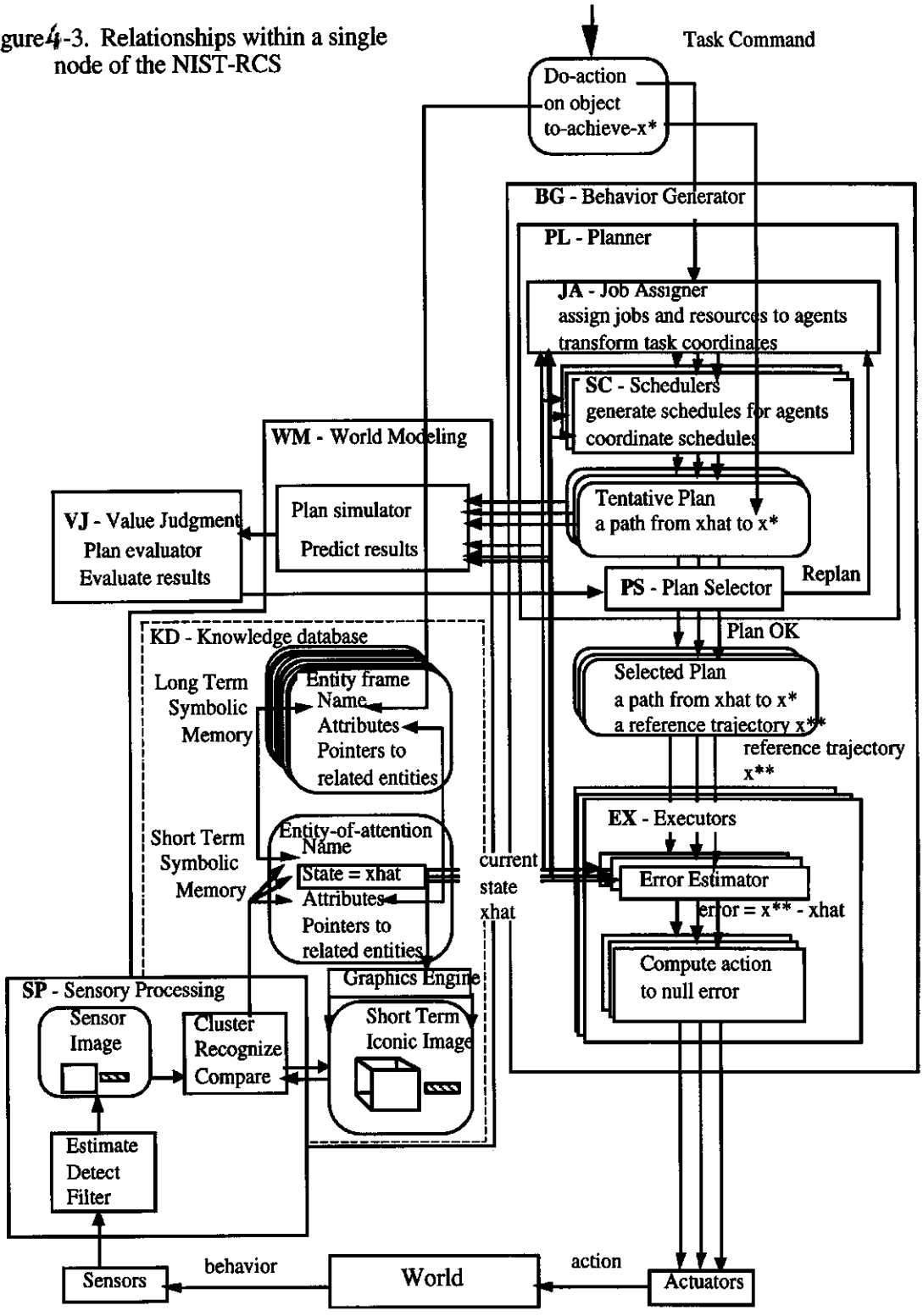
Together, the assignment of jobs and resources to subagents, both the transformation of coordinates, and the development of a coordinated input command schedule, as well as the output motion trajectory for each subagent, constitute the synthesis of a plan. Therefore, output from the JA and SC is a set of tentative alternatives of a plan. JA and SC may generate several tentative plans. Each of these are sent to the World Model where expected results are simulated. The results of simulation are sent to the Value Judgment (VJ) module where the cost/benefit evaluation for each alternative is performed. The evaluation is returned to the Plan Selector (PS) sub-submodule for a decision as to the best plan of action (see Figure 4-4).

This process can be iteratively repeated for different zones of the state space where possible plans are generated. PS performs its function of selecting one plan with the best results of VJ evaluation. With the proper harware architecture, plans can be developed and evaluated in parallel. From the set of tentative alternative plans, the best plan is submitted for execution. The detailed description of the PLANNER is given in Section 5.

At each hierarchical level, plans are expressed in a vocabulary of subtask commands that can be accepted as input by the executor (EX) submodules at that level. For each executor, the string of planned subtask commands constitutes a reference trajectory through the subtask space. Since the real trajectory will differ from the planned one, the value of the error should be estimated. There are many algorithms of estimation which will be discussed in Section 5. However, one thing is important: regardless how the estimation is performed, the action should be computed that will null the error. This process is called "compensation."

The output of BG-module is applied to the input of the Virtual Actuator (VA). At each level,

Figure 4-3. Relationships within a single node of the NIST-RCS



the results of planning have to be transformed into Action. However, the phenomenon of "action" is interpreted at each level in a level-specific way. After the process of decision-making is finished, the "decision" is formulated in the form of Plan. "Plan" contains information of the Goal to be achieved, Virtual Actuators to participate, and Motions they should execute. This Plan is applied to the System as it is visualized at this particular level and the result of planning should be obtained in the form understood at this particular level.

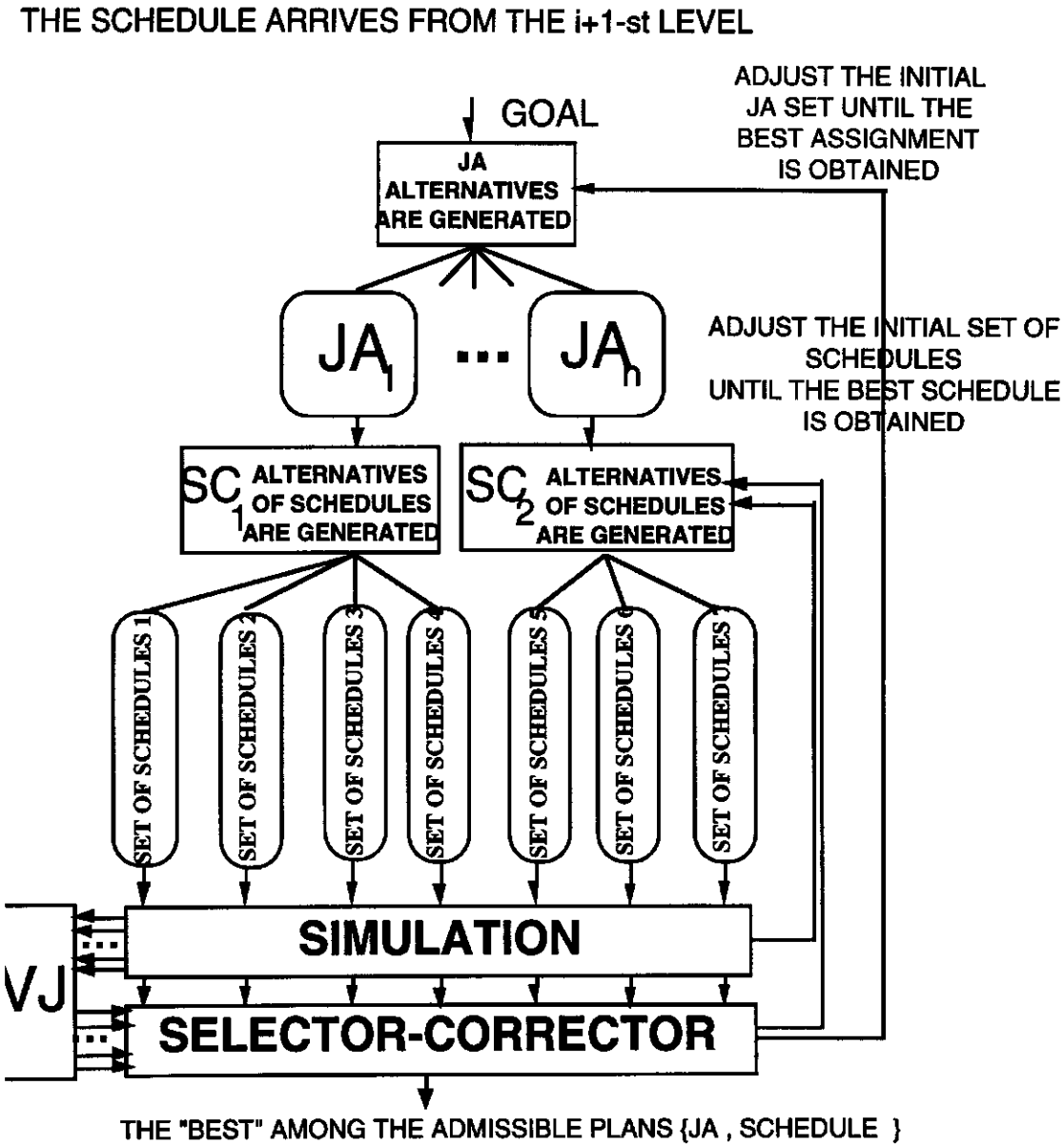


Figure 4-4. The diagram of Planning process

4.6 How the BG-module operates?

In Figure 4.4 a menu of operations and an arsenal of available techniques are presented. These can be applied in various technical solutions of BG-module.

BG-module receives information necessary for planning from the World Model and from other BG-modules at the level of consideration. World Model updates regularly the state information (or the string of states if the algorithm of planning requires). World Model provides the computational support of the search procedures performed by BG-module if there is no appropriate plans in the storage.

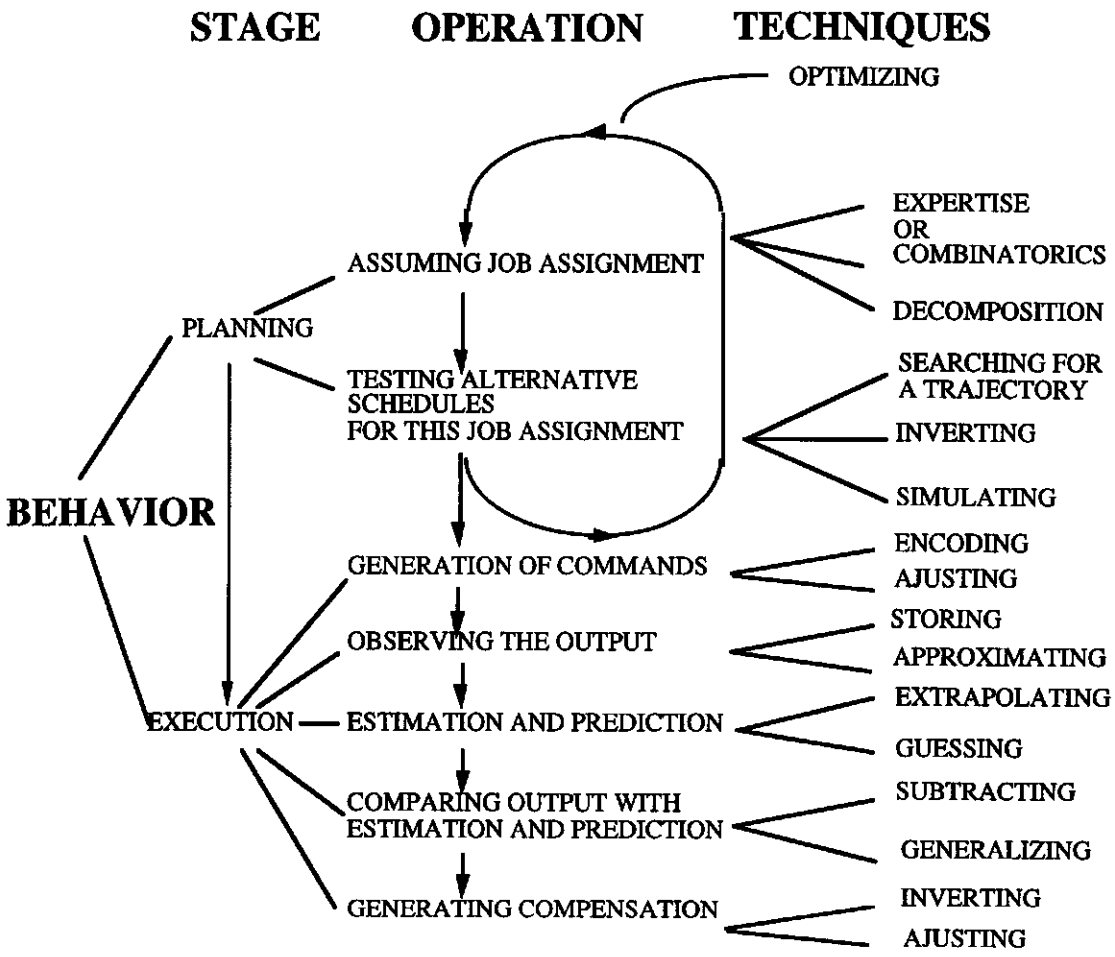


Figure 4-5 Stages, Operations, and Techniques Applied in BG-module

These two consecutive stages (PLANNING and EXECUTION) are performed using the following operations:

A) At the stage of PLANNING

- the set of participating agents is selected
- the schedules of motion are designed for the participating agents
- the input signal (command) strings are computed for Feedforward Control (FFC).

B) At the stage of EXECUTION

- the sequence of commands for FFC is encoded and adjusted
- the error of functioning is evaluated
- the compensation component of the input is computed (feedback control FBC).

From subsection 4-3 one can see that in performing these jobs the PLANNER should perform combinatorial optimization and search for the best combination of agents and best trajectory of motion. Because in most cases the systems are defined via their experimental data, selection of both the set of agents and the best trajectory descends to a recursive procedure of joint search and simulation. This can be performed off-line.

In the meantime, EXECUTOR works primarily online in real time. The error (deviation of the real trajectory from the plan) is measured and compensated for. Both PLANNER and EXECUTOR are discussed in the subsequent sections in more detail.

Functioning of BG-module is illustrated in Figure 4-3. We start with assuming different alternatives of possible Job Assignment (or distribution of the work to be done among the available agents). The alternatives of Job Assignment are chosen for the subsequent analysis and comparison based on available combinations of feasible assignments. The list of feasible assignments is based upon experience of generating similar behavior in the past.

Assume we have two proposed alternatives of Job Assignment. Each of them generates different alternative Schedules (for each of the Agents taking part in this particular combination of Job Assignment). The schedules are received after the desirable motion for each of the Agents is obtained and the output can be inverted to determine the input Schedule. Then, each of the Schedules should be checked in Simulation. The Simulator is contained within WM-module. The schedule enters the simulated string of Actuator-World-Sensors. The result of simulated sensing are processed in the Perception-module. The quality of the particular schedule is evaluated within the VJ-module. After testing all schedules, the result of this testing determines the "best plan" which is selected for the subsequent execution.

4.7 Nested Coordination of Concurrent Processes within BG-module

At a particular level of resolution, we can have more than one virtual actuator. The concept of integrating modules has been already introduced. But, let us imagine that we do not want to apply it. We can consider a "vector virtual actuator" which receives its "vector schedules of commands" from an integrated (vector) EXECUTOR connected to its PLANNER (which can also be treated in an integrated, or vector manner). Now, let us consider a case in which this integrated

actuator (or vector-actuator) is installed at (i.e. provides motion to) a particular subsystem of a system, while another integrated actuator provides motion to another subsystem of the same system (see Figure 4-6).

This will produce coordination of two (or more) concurrently functioning BG-modules for subsystems which can belong to a higher level BG-module of the whole system. The conditions of concurrency cannot (and should not) be prescribed by the BG of the upper level. They should be provided in the course of their PLANNERS concurrently functioning.

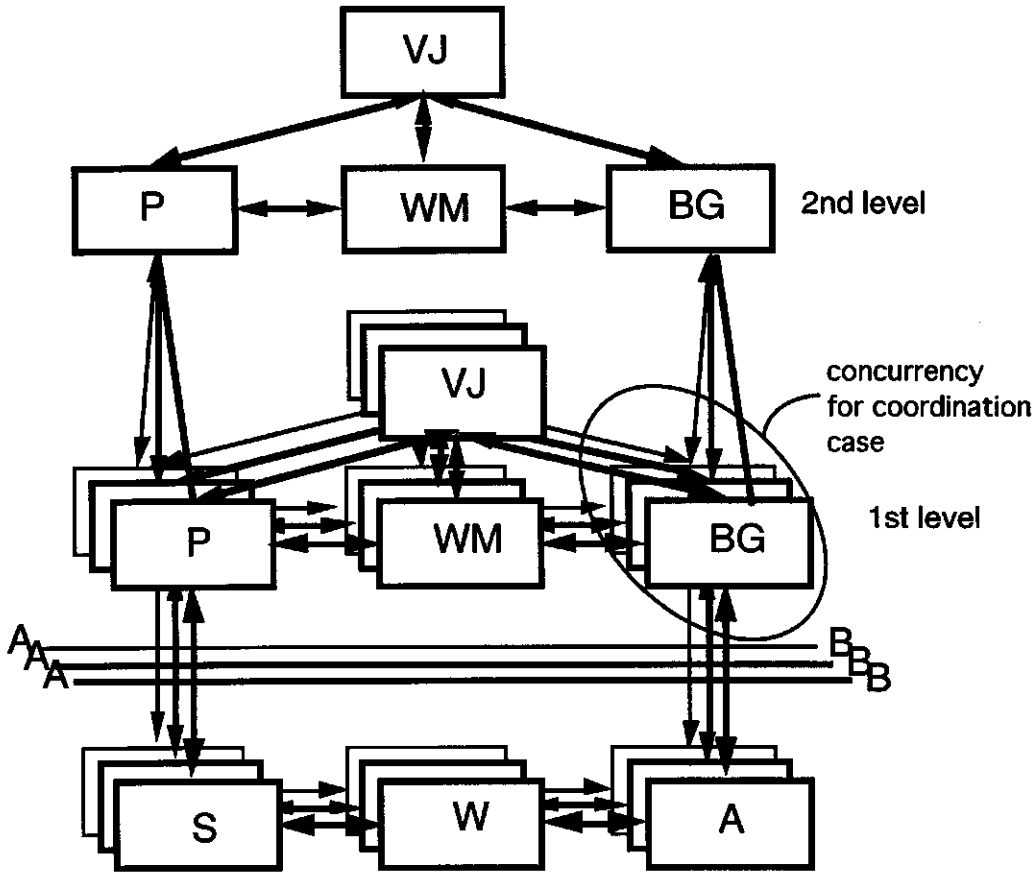


Figure 4-6. NIST-RCS system in which BG concurrency for coordination can be observed.

The PL submodules must communicate with each other if the results of their computation need to be coordinated. They also receive from the World Model a model of the system at a particular level of resolution which can be used for joint searches. The EX submodules at a level also communicate with each other. They do not need to have a model of the system, but they need to know each other's to synchronize their execution.

All this process of top-down planning/job-assignment is performed before the actual

execution starts. This can happen only if we have concurrent operation of all PL submodules which can be represented as a nested system as shown in Figure 4-6.

We would like to specifically focus upon the phenomenon of "nesting" which is an important component of the BG-module functioning. Indeed, the plan at each level cannot be considered completed without incorporating the results of planning from the higher resolution levels. The procedures of high resolution planning can be considered a part of the planning procedure at the level of consideration. This phenomenon spreads both top-down and bottom-up.

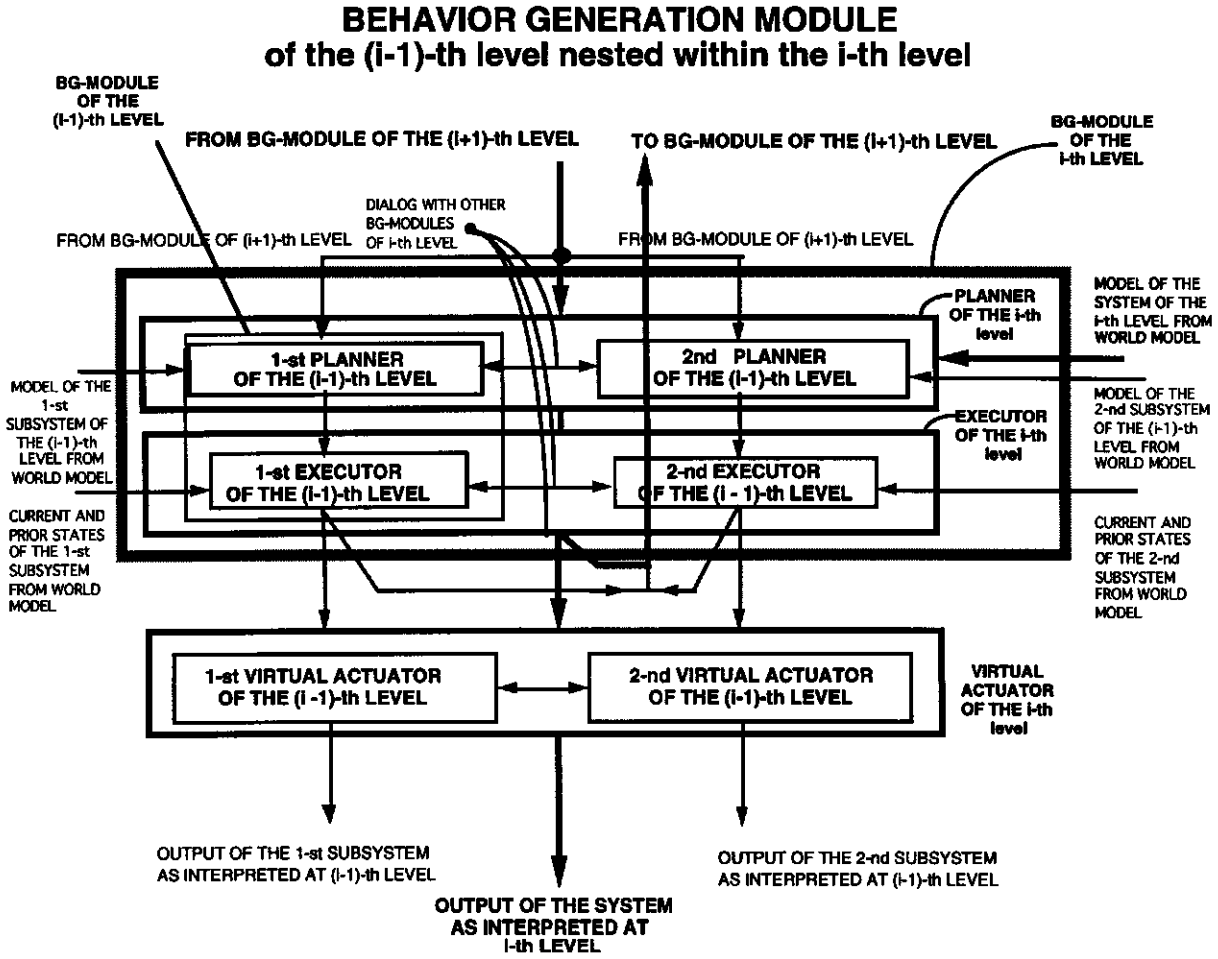


Figure 4-7. More that one BG-module at a level should work simultaneously (both nesting and concurrency phenomena are observed)

As Figure 4-7 shows, the PLAN is coming as a sequence of commands from (i+1)-th level down to the i-th level. This is performed in a twofold manner. Before the execution started, this command string is coming with no changes introduced by the EXECUTOR since no compensation is introduced. During this period, the compensation function are temporarily disabled. As soon as

the PLANNER of the i-th level has accomplished its operation, it submits the results both to the (i-1)-th level and to the (i+1)-th level.

It is important to notice — PLANNERS communicate as independent units. Their communication encompasses their inputs and their outputs. However, their inner (sub-subsystems) functioning is conducted in a totally independent way.

Sending it to the (i-1)-th level allows the (i-1)-th level to look ahead: because the PLANNER module of the (i-1)-th module is at the input of the VIRTUAL ACTUATOR of the (i-1)-th level, as it starts functioning for the level above it means functioning of the Actuator. Indeed, without a feedback from the PLANNER module of the (i-1)-th level, the PLANNER module of the i-th level cannot complete its mission. Sending the results from PLANNER of the i-th level to the PLANNER of the (i+1)-th level plays the same role — without this communications its operation could not be completed. Indeed, it confirms to the (i+1)-th level that its plan is admissible, or that it must be corrected and replanning is required.

Here we return to the concept of nestedness explored earlier. In the same way as each BG-module of the lower level is nested within its adjacent level from above, each PLANNER of the i-th level is nested too within the PLANNER of the (i+1)-th level[45]. Similar phenomenon should be expected for their sub-submodules too.

[45] see subsections 1.4 and 2.3

5. PLANNER

Theoretical discussion of PLANNER was presented in sub-subsections 3.3 and 3.4. In this section, we delineate specific details of PLANNER as a subsystem of BG-module. Planning is a popular topic in the area of intelligent systems. However, in AI-papers, planning was treated in a lopsided way. It was either considered a technique of the knowledge-based task formation, or a part of general process of decision-making—never as a part of a control process. In control theory, planning became a subject of discussion only recently, after the supervisory control became a legitimate theoretical topic. PLANNER as a part of a multiresolutional control hierarchy, is a new topic, and many details of it will be discussed in this Section.

5.1 General discussion of planning as a process

In the subsequent discussion, PLAN is understood as the set of data which includes the following components:

- The output time-trajectory of motion which can be represented for example, as time schedules[46] for the components of the output vector; generation of this output trajectory is assigned to the set of virtual actuators of the virtual control system at the output of the node under consideration. This trajectory is considered the "best" one out of all possible set of alternatives. It is denoted X^*_i in Section 4.
- The trajectory of the action vector which can be represented for example, as time schedules of actions. It is denoted $\{\{A_v\}_{i,j}\}_k$ in Section 4, p. 125[47].
- The time-trajectory of the input control vector which can be represented for example, as time schedules of control vector (or the vector of control commands). It is denoted in Section 4, p.123.

These components of the plan cover the control problem almost exhaustively. The feedforward control solution is satisfied by the PLAN. However, if the model we use for planning differs from the reality, the output computed as a part of PLAN will differ from the really desirable output. The input prescribed by the PLAN, will entail even larger mistakes in the output. This is why WM is supposed to constantly update the set of models employed for planning. This is why BG-module is equipped by EXECUTOR which performs the on-line feedback compensation (see Section 6).

It is presumed (and it can be proven for many practical cases) that there exists "an optimum" PLAN which maximizes the cost-function for a particular case. Certainly, there exists

[46] Time schedule is a couple of ordered lists with links of correspondence between them. The first list is a list of vectors, and the second one is a list of time instances.

[47] Notations: i is the level of resolution, j is the number of an actuator, k is the number of the alternative of job assignment.

"the best" PLAN which provides for a sufficient value of the cost-function for a particular case, or which is enclosed in the "envelope of desirability"[48].

PLAN can be decomposed in the same way as the objects, actions, and tasks. A multiplicity of SUBPLANs is possible in the general case when different combinations of actuators are available. Each has its own plan with its different schedules of performing particular tasks in a concrete time (each as an individual plan), which can be assigned to the actuators. It is presumed (and it can be proven for many practical cases) that there exists "the best" PLAN which maximizes the cost-function for a particular case.

From this section, one will find that there is no difference between planning and feedforward control. Planning and control problems are interrelated and do not have much meaning one without another. This is why we use a term planning/control where possible to underline inseparable character of these two operations. Planning is equivalent to feedforward control at each level of resolution, although for the adjacent level of the higher resolution (the adjacent level below) plan is associated with the goal, which came from above. It differs from the control computed at the level by some inherent *lookahead* connotation.

5.1.1 Epistemology of PLANNER

PLANNING is interpreted as a design of required activities and as a design system it functions in the domain of knowledge representation. If the System is built (and this was our assumption in the beginning, for simplification of the presentation) this design is done for the System which has already been manufactured and exists. However, for the much higher levels of generalization (lower levels of resolution), planning is a design both of the required activities and a system which should be a platform for realization of these activities[49].

Planner has at the input the string of commands (the assignment) from the upper level, which entails the results of using models obtained from the World Representation of the upper level. It also has models obtained from WM at the level under consideration, and it has information about current states, and information about planning activities of other PLANNER submodules of other BG-modules at this particular level of resolution. PLAN is the output of PLANNER and is a string of commands to obtain the desirable motion of all of the subsystems under its control at a particular resolution level, which is submitted to the higher resolution levels together with the description of the desirable motion at the output.

PLANNING has its own time constraints. We will try to avoid unnecessary search and reduce the space search as much as possible. Any reduction of the search envelope after some particular limit can lead to a loss of the "very best" plan. We will consider the envelope reduction

---

[48] H. Simon call these PLANs to be "satisficing" ones.

[49] The design as a part of NIST-RCS architecture as a part of integrated manufacturing is described in a separate paper.

as increasing the efficiency of planning and reducing its reliability. This is the trade-off between the replanning frequency and optimality. If the plan is recomputed frequently enough, it needs only to direct the process properly at the point of consideration while the future details are less important.

PLANNER uses knowledge of the Loop of Functioning as this knowledge is represented in the World Model (WM). The latter is the result of learning and reflects our long term WM knowledge of the system. This knowledge has two types of errors:

a) Large error — due to unmodeled variability of the objects represented in the World. These are produced by the prior operations of generalization which synthesized these objects from the higher resolution objects of the level below. There, errors determine the interval of certainty about each parameter.

b) Small errors— due to the minimum discrete increments of representation at the particular level of resolution.

Large errors emerge during the process of learning and determine the variability of costs for the alternatives of the plan. Small errors determine the lower bound of the plan's error. Small errors can be reduced only at the higher resolution levels (if they exist)[50]. Large errors affect forming the alternatives of the trajectories by PLANNER. Small errors affect functioning of EXECUTOR.

It is important to realize that PLAN is developed based upon the model WM_{i+1} while it should be applied at the i-th level which uses a different World Model (WM_i)— more narrow and more precise.

5.1.2 Functions of PLANNER

Functions of PLANNER are listed as follows:

• The main function of PLANNER of the level is to find "the best" set of PLAN at the level. The main function of PLANNER as a multiresolutional nested system is to find the multiresolutional planning/control system of input commands[51] .

• PLANNER of the i-th level receives the assigned subset of a PLAN from the (i+1)-th level (an adjacent level from above) which is called the TASK. This TASK is finalized by the BG-module of the (i+1)-th level only after corrections introduced by EXECUTOR online ("online" in the sense of the virtual ELF of the (i+1)-th level). PLANNER of the i-th level uses this TASK and the model submitted by WM to create tentative combinations of the plan distribution JA among the virtual actuators[52] (VA), and then contemplates which of all possible output trajectories performing

[50] At the level with real actuators world and sensors there is no higher resolution.

[51] In the subsequent material we use the term PLANNER for a subsystem of BG-module at the i-th level.

[52] PLAN is determined for the Virtual Actuator, not for an Agent. We use the term Agent as an equivalent for ELF because in the literature, Agents have elements of intelligence like ELF has.

the TASK, is "the best" one.

• PLANNER determines the appropriate alternatives (hypotheses) of the desirable combinations of VAs to perform the task. These combinations can be known in advance for simple case. However, in general, they should be found by making all possible combinations of VAs and testing all of them ("combinatorial search".)

• PLANNER computes the best trajectory of motion for each hypothesis of the combination of VAs. In some cases, this trajectory can be computed analytically. However, generally, the output trajectory can be obtained also as a result of search. All meaningful strings should be created ("combinatorial search"). One or more strings regarded as "the best" are to be inverted into the input commands space (all this is done for each particular combination of VAs, which has been determined by selecting the coordinates of the input space). So, two consecutive search operations are supposed to be performed. Otherwise, the PLAN cannot be obtained.

• After this double search is done, PLANNER distributes the best trajectory of motion among the VAs, and determines the best trajectory of each VA's motion.

• PLANNER transforms the best trajectory of the VA's motion into the VA schedule (the task's time distribution, the input commands time distribution, the actions to be done time distribution). Note that at the output of PLANNER, we receive a set of schedules which when selected are already distributed among the virtual actuators.

• PLANNER transfers the set of schedules to the EXECUTOR (see Section 6).

• PLANNER under consideration of the higher resolution level, or HRL reports inconsistencies and singularities to the BG of the lower resolution level (LRL.)

• PLANNERs of the HRL are considered a nested component of the PLANNER of the level under consideration. PLANNER corrects its PLANs after receiving from the HRL PLANNER its responses concerning the inconsistencies and singularities. However, it does not communicate with PLANNERs of other levels of resolution (only with the adjacent level below).

• PLANNER communicates with other planners of the same resolution level to coordinate working processes, e.g. sharing resources in situations that arise due to uncertainties at the stage of planning.

• PLANNER of the i-th level receives from the PLANNERs of the submodules of the (i-1)-th level their final results of refined planning and evaluates the overall performance; if replanning is required it replans and resubmit the new plans to the levels of higher resolution.

• PLANNER requests for the new planning alternative as its adjacent LRL PLANNER responded with a result which is not satisfactory.

• PLANNER makes the final decision on the final alternative of the package of schedules.

At the highest level of resolution no job distribution is required: the ACTUATOR is not a "virtual actuator" anymore. It introduces changes in the real world. (Of course, the metaphor of "virtual actuator" could be continued into the domain of reality but we are not going to do this).

5.1.3 Planning in multiresolutional space vs. planning in abstraction spaces

In this book, we depart from the usual planning paradigm as treated in Artificial Intelligence. The problem of planning was traditionally treated in the AI area following the STRIPS, ABSTRIPS, NOAH, MOLGEN, and SIPE paradigms. Planning was considered to be a theory of "reasoning about actions" and meta-planning was introduced to make this process more efficient (PANDORA paradigm[53][94].) Of course, anything can be regarded as reasoning. But, the AI reasoning presumes boils down to manipulating with collections of the schemata-like stimulus-response couples[54] [95]. The conventional AI paradigm of planning can be presented formally with the help of automata theory[55][96]. Our paradigm is broader and cannot be satisfied by a simple automata-formalism.

In all of the prior planning paradigms (including AI), the advantages of multiresolutional (multigranular, multiscale) systems such as NIST-RCS were not fully appreciated[56][97]. The commonalities of all proposed techniques of planning were not understood as they can be visualized within NIST-RCS paradigm. The commonalities among the planning and feedforward control (FFC) have not been noticed at all. It was especially clear from the way multigranular (multiresolutional) planning was treated in the literature. In this book we affirm the similarities between planning and FFC. The difference among the levels of different granularity is simply in the length of lead time and the frequency component in plan.

Although many of results are related to "planning in abstraction spaces," the AI authors convey totally different meaning than the one we convey in the NIST-RCS[57][98]. Indeed, the levels of multiple resolution employed in NIST-RCS are not the "levels of abstraction" associated by many AI researchers with hierarchical systems. The processes of aggregation/decomposition used in NIST/RCS are rather associated with generalization/instantiation and not with abstraction/specialization as AI authors often assume.

---

[53] Reasoning implies using logic for inference. The planning paradigm proposed in this book seems to follow logic of multiresolutional control systems. In this, we supplement the way of reasoning typical for the predicate calculus of the first order (without contradicting it.)

[54] M. Arbib's theory of schemata is a powerful tool of applying concepts automata theory to the domain of intelligent systems.

[55] A reminder: at each level of resolution, the automata representation has an equivalent representation presented in the language of differential and integral calculi (and vice versa.)

[56] E. Sacerdoti uses multilevel planning. However, generalization supplemented by an explicit change of granularity could be of high advantage to the planning process.

[57] "All classical AI planners have distinct and well define planning levels. In these systems, a new planning level is created by expanding each node in the plan with one of the operators that describe actions. In the literature, these levels are often referred to as hierarchical, which implicitly associates them with abstraction levels. In fact, they are independent of abstraction level; a new planning level may or may not result in a new abstraction level depending on which operators are applied" (see [98])

Unlike "abstraction spaces," our levels of multiple resolution can be legitimately associated with frequency or scale decomposition similar to those known from the fractal or wavelet theories [112, 113]. The meaning of planning in NIST-RCS is solving the control problem within separate "frequency domains." What we call "planning in generalization levels," or in "multiscale levels," or in "levels of different resolution" could be called rather "finding controller for spectral regions." Indeed, if the spectral density of the system looks as shown in Figure 5.1.

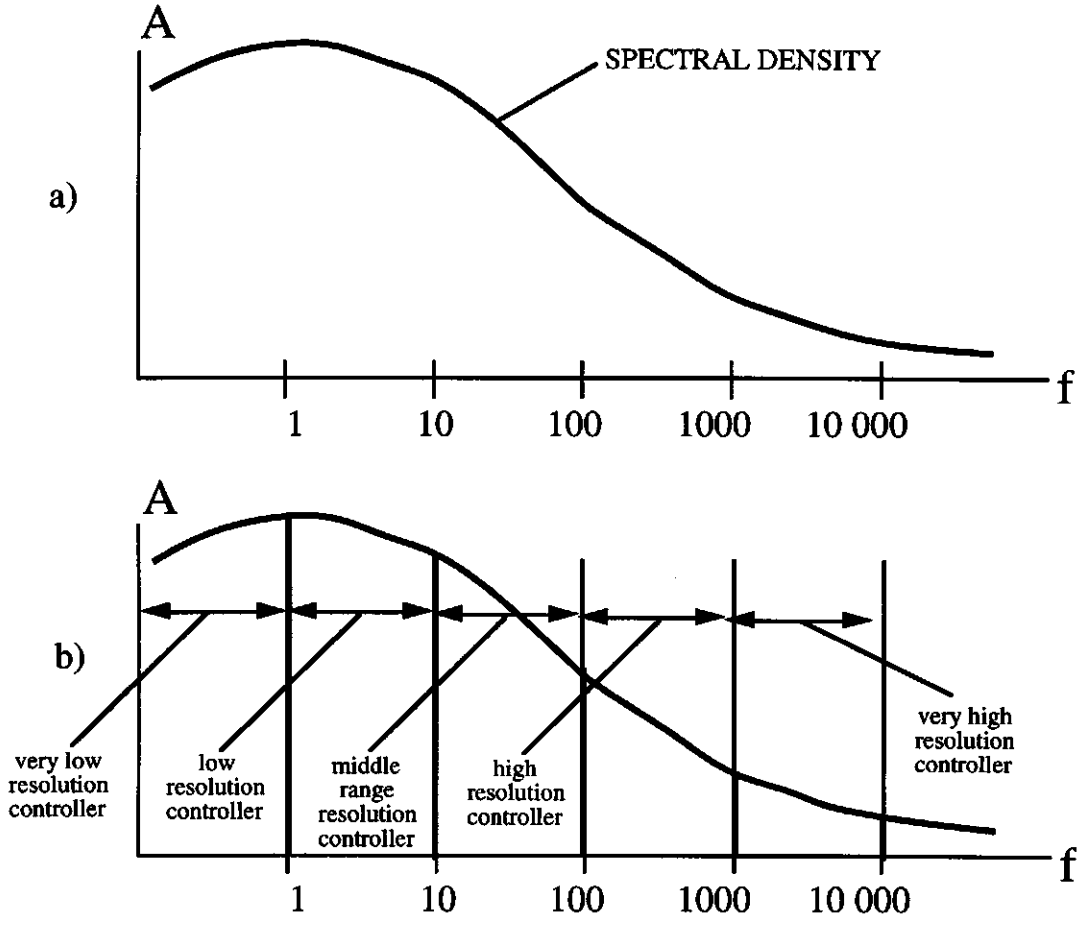


Figure 5-1 Spectral region controllers concept

If planning is a feedforward control, why not treat it as a control and why not to look for an analytical solution? This question is based upon an apparent confusion. In fact, there is no analytical methodology of finding the output trajectory for a feedforward control which satisfies some conditions, such as a set of constraints and/or delivering minimum to a cost-function. Solution is known only for extremely simple, trivial cases. In control theory, searching for a feedforward control is frequently avoided by determining inputs online as a function of outputs or other state variables of the system. This strategy is accepted based upon an existing opinion that

this is better than determining inputs as functions of time. This is true in many cases. But it is also true that combining these two principles will give even better results. This is exactly what is employed in NIST-RCS: determining the most significant part of the input as a function of time (PLANNER) and determining the compensatory part of the input taking in account the direct measure of the output (or/and other variables). There is a lot of evidence that if this principle is applied consistently throughout the entire hierarchical system, it can give the best result in comparison with all other possible decisions.

For most of cases, feedforward control requires search. We treat planning at each level as a procedure of search (first, in the storage, then in the state space.) Other authors do not visualize the uniformity of the search processes and address the issues of motion planning as if they are different from the "task planning"[58][99]. We do not distinguish between these because we address planning at all levels as a search in the state space.

5.1.4 Planning in the Task Space vs. Motion Planning

The term "planning" is equivalent to the term "feedforward control". Planning means finding a desirable trajectory of motion (including both control and output variables) using the available knowledge at the time of planning. The controversy emerges because "task space planning" is associated with *discrete events* while "motion planning" is usually linked with *continuous motion.* We see the difference between these two planning problems only in the resolution by which the state space is discretized[59] . The world models, the state information, and the control sequences in NIST-RCS are discretized anyway. In the "task space" the space is discretized in the natural manner because of topological discontinuities, which are also called morphological discontinuities and catastrophes. These are the places and the moments of time when the cutting tool enters the workpiece, finishes the process of cutting or when the gripper of a manipulator leaves the initial position, arrives at the final point or when the process of obstacle avoidance starts and ends.

From the last example with the obstacle one can see that the event of "starting the process of obstacle avoidance" is a fuzzy event. Its discrete time cannot be assigned with precision. One can easily deduce that between the two domains — one is discrete events and another is the domain of continuous motion, there is an area of fuzzy transition. Between the domain of "task planning" and "motion planning" there is a continuum of different state space tessellations.

This is why in NIST-RCS, we will not distinguish between the principles and techniques of planning which will be applied at different levels of resolution.

---

[58] The conceptual barrier existing between continuous systems and DES can be easily avoided in the case of "planning".

[59] The principles of representation including choice of the combination "scope of representation" and "minimal distinguishability zone" in space and in time are described in a separate report "NIST-RCS: World Model".

As the hypotheses about available JOB ASSIGNMENTs (and the initial decomposition of the work to be done) are completed in JA, the motion trajectory (including its output and input time-trajectories) should be found in SC which allows for the ACTION REQUIRED. This is the first stage of PLANNING. Preplanned trajectories can be stored[60] and browsed through when the need arise, or a search for the desirable trajectory of motion can be conducted.

Because the result will be assigned to the agents in a form of a final state required under conditions to be held (cost-function and constraints), PLANNING proceeds with developing schedules, or which is the same, finding the output time-profiles for the agents. The results of planning include the trajectories <OUTPUT> to be executed (FFC), and the time-profiles or trajectories <INPUT> to be applied to execute the output trajectories. The results of planning can be interpreted as time-tagged strings of commands <TASK TO THE NEXT LEVEL>. The time-profile <OUTPUT> is found by solving the optimization problem, i.e. by minimizing the cost function S:

$$<\text{OUTPUT}><\text{---------}S_{\min F}[T_k,\ R_k(\text{WM, JA}),\ \rho_{ik}\ ,\ \Delta t_k] \qquad (5\text{-}1)$$

T_k - task submitted to the BG-module at the k-th level of NIST-RCS

R_k(WM, JA)- representation admitted within the paradigm of the WM and JA functioning

ρ_{ik} - minimal distinguishability zone admitted at the particular spatial resolution in the i-th coordinate at the k-th level of resolution

Δt_k - minimal time interval admitted at the particular temporal resolution at the k-th level of resolution.

As the PLANNING of motion trajectory is occurring for each of the subsystems at a level, the cooperation of these subsystems is negotiated, and the FINAL JOB DISTRIBUTION is performed.

To execute the desired motion trajectory, the second part of PLANNING should be done. The time profile of the input to the lower (higher resolution) level should be found. The latter is done by an inverse procedure which results in the sequence of the output EXECUTION commands. If the level operator is presented in the form

$$<\text{OUTPUT}> <\text{---------}\ T(<\text{INPUT}>) \qquad (5\text{-}2)$$

where **T**-is the transformation function of the controlled system (a plant operator, a model of the system), then the inverse procedure can be expected (the "input" is computed from the "output" prescribed)

[60] The double-searching during planning activities is not only the source of a particular result, but also a source of meaningful alternatives which can be stored in the library of "alternative solutions.' This is one of the first entrances into the future RCS system with learning. The system can learn not only from its real experiences but also from the imaginary situations it has encountered during search procedures in its "imagination."

$$\text{<INPUT>} \leftarrow\text{---------}\ T^{-1}(\text{<OUTPUT>}) \qquad (5\text{-}3)$$

and substituting for <OUTPUT> from (5-1) we receive

$$\text{<INPUT>} \leftarrow\text{---------}\ T^{-1}*S_{minF}[T_k,\ R_k(WM,\ JA),\ \rho_{ik}\ ,\ \Delta t_k] \qquad (5\text{-}4)$$

where $P=T^{-1}*S_{minF}$ is the planning function.

The activities of BG-module include planning and execution with corresponding cooperation among the agents, and this should be reflected in the interfaces of all modules. The OUTPUT trajectory and the corresponding INPUT, as well as COMPENSATION, can be based on different premises linked primarily with the existing models within the system of representation in the WM-module (entity-relational graph, object-oriented representation) of the machine.

5.1.5 Reactive vs. deliberative decision-making

Theses two types of decision-making reflect the epistemological characteristics of the system. Deliberative decision-making is possible if the model exists and there is sufficient time to simulate different hypotheses and select one of them. However, when the models are not available and/or there is no time to build the hypotheses and test them, the reactive decision-making is the only choice. We have to have a menu of predetermined responses which are supposed to be beneficial in most cases.

The issue of reactive planning has arisen during research at SRI on combining SIPE [101] System for Interactive Planning and Execution Monitoring) and PRS [102] (Procedural Reasoning System) to control the indoor mobile robot. In the AI community, there is an impression that robots should be controlled in a reactive manner. We concluded that this conviction is wrong, and no intelligent operation can be done without online (or offline) planning.

The deliberate planning is considered as a different issue from the *reactivity* property: "The ability to act appropriately over a broad range of situations without deliberation is called reactivity, and is an important measure of competence for robots controlling dynamic and unpredictable processes" [103]. In the literature, there are many myths and fantasies related to reactivity. Many authors believe that all agents should be reactive agents (apparently due to the lack of knowledge of predictive control systems). Also, many authors do not realize that reactive control is a synonym to the "feedback control"; some authors propose "slow planners embedded into fast real time control systems" [104].

NIST-RCS is based on the concept that the goal-oriented system must be both deliberative and reactive[61] in generating its behavior that includes planning/control activities at each level of resolution. It should be deliberative to the degree of existing knowledge of the model and expected situation. It should be reactive to the unpredictable and unexpected factors, i.e. the factor of "unexpectedness" should be handled well. This combination of deliberative/reactive (or reflective/reactive) strategy of behavior generation holds at each level of resolution. Another interpretation of the behavior generation strategies and processes can be formulated as follows: each level can be considered as goal-deliberative and unknown-circumstances-reactive, while it is known-circumstances-deliberative and subordinate-deliberative. Most of this terminological discussion would be unnecessary if we represent the operation of NIST-RCS in the terms of feedforward/feedback control which are correspondingly computed using known and unexpected information.

5.2 What is inside the PLANNER[62]

PLANNER is effectively a feedforward control (FFC) submodule. Its function is to find the optimal FFC function which presumes both spatial/temporal motion design and motion distribution among the agents. This submodule performs the following three operations:

1) Spatial Planning including Job Assignment among the Virtual Actuators

2) Coordinate transformation from from task to subtask

3) Temporal Planning (Scheduling)

The overall structure of the PLANNER-submodule contains the following set of submodules (see Figure 5-2.) The operations of spatial and temporal planning are not separable in principle. They can be considered separately only for the sake of presentation and/or for very simplistic cases of the NIST-RCS functioning. More importantly, they are executed separately by existing computational systems because it is convenient algorithmically to interlace sequentially the procedures of spatial and temporal searching. On the contrary, the procedures of comparison and decision making are combined in a stand-alone set of operations that should handle a set of

[61] Reactive - based upon response to a stimulus, usually alludes to responding to something not expected and/or planned in a deliberative manner. This definition contains some circling because if one knows how to respond to a particular stimulus, this means that one have deliberated and prepared this response. An example of reactive response: avoidance maneuver when the obstacle emerges. The word "reactive is often substituted by "reflexive" because "reflex" is regarded as "action in response to the stimulus".

[62] This subsection is an excerpt from [105].

results obtained from the several cycles of spatial and temporal search for alternatives.

5.2.1 The sub-subsystems

PLANNER does its job by generating and contemplating different alternatives (hypotheses) of distributing the future activities among the potential subsystems at the higher resolution level. PLANNER determines them both in Space and Time. Combining hypothetical alternatives and analyzing them amounts for simulation. Planning presumes *simulation* of multiple hypotheses of the future.

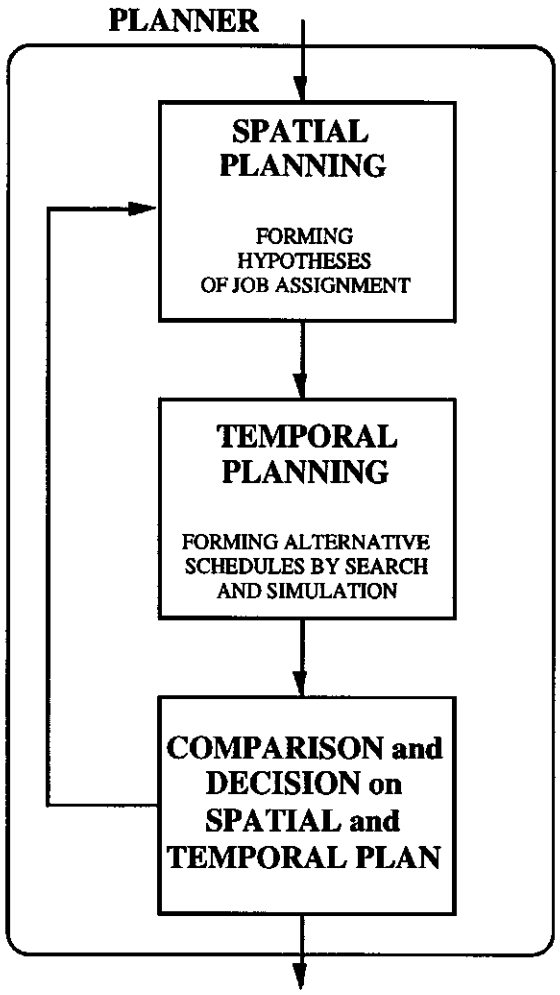


Figure 5-2. A Sequential Structure of Search for Spatial and Temporal Plan.

Interestingly enough, all algorithms of search contain simulation of the processes that has not yet happened. However, in planning we are interested in knowing "what if". This immediately creates a link between the concept of planning and the following concepts of intelligent information

processing:

1) **Grouping (G)** of the units of knowledge based on their similarity. We are interested in patterns of the state that emerge within the state when we contemplate its evolution into the future. Building of patterns demands for grouping of entities and/or relations by similarity and recognizing patterns within the emerging patterns.

2) **Focusing Attention (FA)** is required otherwise the abundance of computational procedures to be performed creates substantial predicament of computational complexity.

3) **Combinatorial Synthesis (CS)** is creation of imaginary "possible worlds", or plausible alternatives of the state evolution.

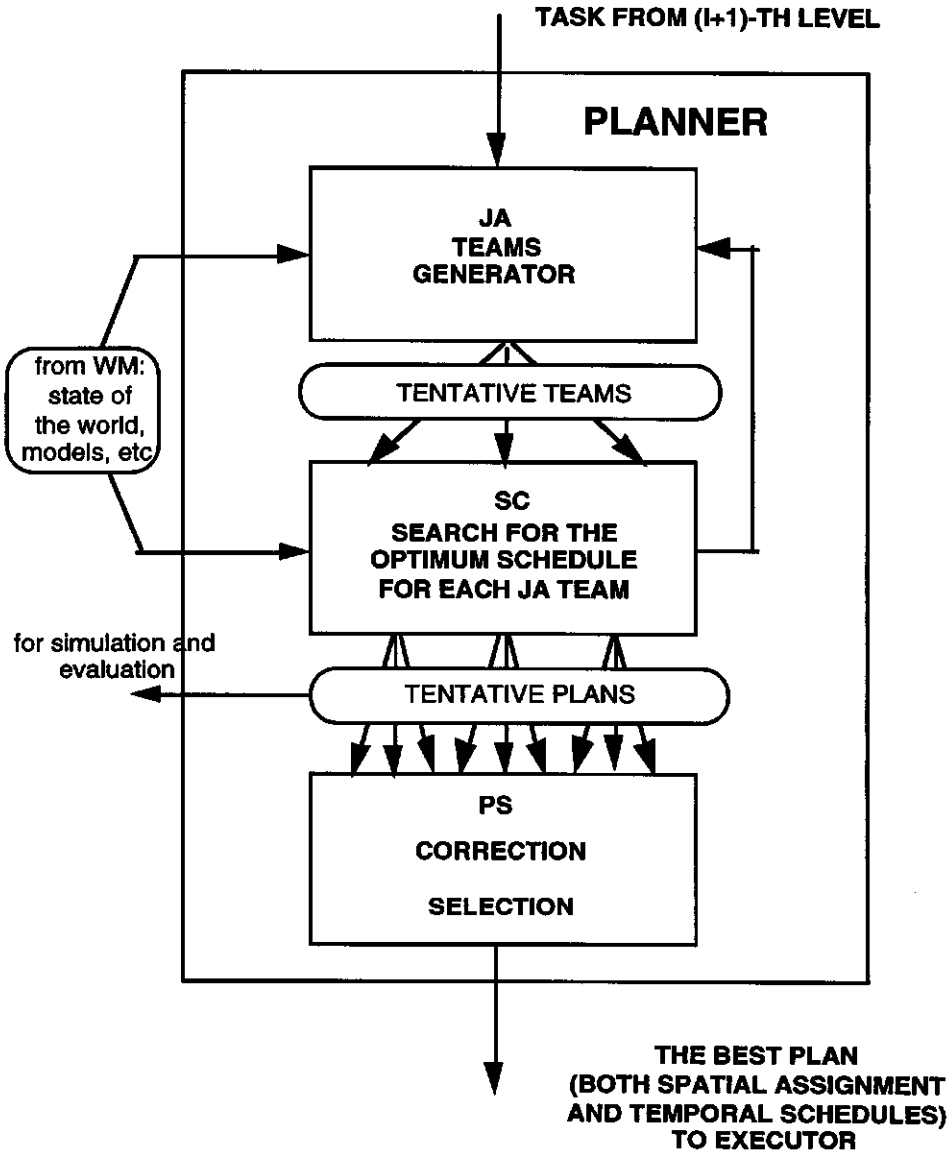


Figure 5-3. The inner structure of PLANNER

This triplet of concepts (GFACS) [78] is a basis for computing *plans*.

The inner structure of PLANNER is built according to the concept of its functioning described in Figures 4-2 and 4-3. This structure is shown in Figure 5-2 for a sequential processing (other versions are also possible.)

PLANNER comprises three compartments (Figure 5-3):

1) JA generates tentative assignments of consecutive elements of behavior (jobs, operations, contemplated strings of input commands, other decomposition in the space representation) subject to subsequent simulation as a part of searching the best plan. These activities can be also interpreted as "team generation" resulting from the search/simulation. JA modifies the teams to correct the assumptions and improve the results of subsequent simulation. JA cannot make any corrections unless each of the "teams" generated a search for the best functioning in time. This search is performed by the scheduler (SC). Search procedures performed by JA and SC can be unified into a joint process of generating planning alternatives.

2) SC is searching for the optimum schedule for each of the teams postulated by JA. It performs the time decomposition of system functioning. Searching for a schedule which can satisfy a set of requirements, means simulating the process of functioning. Each search algorithm is doing this abstractly. Sometimes, it is desirable to repeat the simulation in a less abstract manner with more detail in a broader paradigm. Then, the limited number of alternatives is sent to WM for simulating them in more detail.

As the results of search/simulation are arriving, SC modifies the time distribution of jobs, and JA corrects the teams to improve the results of the subsequently repeated simulation.

3) CORRECTION/SELECTION. This subsystem is supposed to make a final choice.There is a feedback loop from the results of search to the combinatorial process within the JA and SC. It allows for refining the prior coarse results of combinatorial synthesis, and searching for a more accurate fitting within the condition of "satisficing" and/or "optimum".

5.2.2 The process of planning[63]

PL submodules accommodate a variety of planning algorithms. These range from a simple table look-up of precomputed plans (or "scripts") to the real-time search in the state space or configuration space, or even game-theoretic algorithms for multi-agent cooperating or competitive groups. However, regardless of how the plans are synthesized, a plan consists of a spatial decomposition of a task into a set of job assignments and resource allocations to virtual actuators, plus a schedule of subtasks for each actuator, ordered along the time line. In many

[63] This concept is similar to the CMAC concept (see J. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", *Trans. ASME, J. Dynam. Syst. Meas. Contr.*, vol. 97, pp. 220-227, Sept. 1975, and the recent CMAC applications: L. Kraft, D. Campagna, "A Comparison Between CMAC NN Control and Two Traditional Adaptive Control Systems", in Eds. E. Sanches-Sinencio, C. Lau, Artificial Neural Networks, IEEE Press, 1992, pp. 483-490).

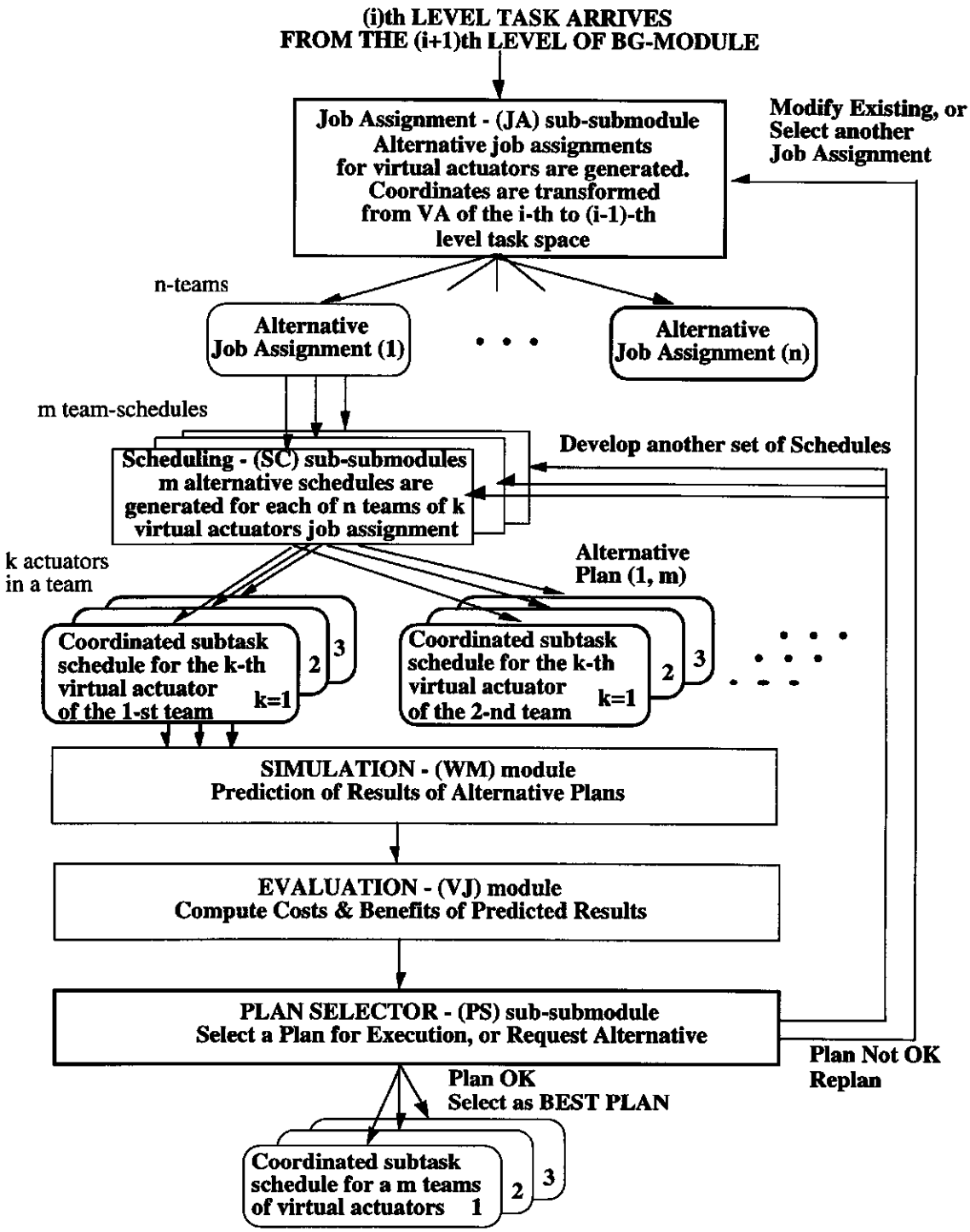


Figure 5-4 The diagram of Planning submodule.

Rectangular boxes are computational modules. Boxes with rounded corners are data structures.

cases, it is required that the schedules for the actuators should be coordinated to produce coordinated actions.

In general, planning consists of the following steps:

- generating a set of alternative plans including time and space distribution
- simulating the likely results of those plans by testing the alternatives[64]
- correcting the alternatives by using the results of testing
- evaluating those results according to some cost/benefit criteria, and
- selecting the tentative plan with the best evaluation for execution.

The functional structure of the Planner is shown in Figure 5-3. It is the function of the PL sub-submodules to generate and/or select a plan in response to the input of a task from the next higher level BG-module. Let us discuss it in more detail (see Figure 5.4).

It has been already introduced that PLANNER-submodules can be further decomposed into the sub-submodules of Job Assigner (JA), Scheduler (SC), and Plan Selector (PS). These sub-submodules function as follows. JA assignments are applied for tentative scheduling. The assignment can be considered a "tentative team formation." For each tentative team, a schedule is computed by SC. SC intermediate results are constantly challenged by introducing a new assignment from JA. So, the sequence of functioning looks like a string [JA-SC-JA-SC-...]. Therefore, JA and SC form a loop from which final versions of plans are chosen by the selector SC. Before the alternatives submitted to PS for final selection, an inquiry is possible to WM and VJ modules for simulating the meaningful set of alternatives.

JA generates the alternative job assignments for the virtual actuators. It transforms the Task from the level of virtual actuators of the (i+1)-th level of resolution (adjacent level from above) into the language of virtual actuators of i-th level (the level under consideration). After translation is done, JA applies the operator of "tentative teams formation" (which is the synthesis of combinations under constraints) and generates a set of n "actuator teams" of the i-th level of resolution. Each of the teams is considered an alternative of Job Assignment.

Each alternative of Job Assignment is considered an input to the scheduler SC together with the World Model pertaining to the case. SC generates m schedules for each of the n teams and for each of this schedule k schedules for each single actuator of the m schedules of the n teams. It is voluminous. Any opportunity to reduce ("prune") the search is considered.

These schedules are simulated in WM and evaluated in VJ outside of BG-module. In the well-structured cases with extensive prior experience the stage of external simulation can be skipped. All schedules are evaluated and compared in PS, after which one set of plans is selected as the

[64] In many cases, search for alternatives by JA and SC is performed on a sufficiently complete model of the system and can validate for being a "simulation". A separate stage of simulation is required only if the model employed by JA-SC is a simplified one and discrepancies are expected.

"best" PLAN. For this plan, the functions of "output," "action," and "input commands" are prepared for further communication considered with smaller horizon (for the (i-1)-th level of resolution) and then submitted for execution.

5.2.3 JA generates tentative teams

This subsystem is based on a mechanism of forming combinations among the available actuators and tentative distribution of their share in support of the overall motion to be provided. Most of the intelligent systems have redundancy, which allows for a multiplicity of possible combinations. The total job J_T should be a sum of the actuators jobs J_i (i=1,2,...,n; n-total number of available actuators) with different relative contribution determined by the share coefficient s_i as one can see from the "equation of job distribution":

$$J_T=s_1 J_1+s_2J_2+\ldots+s_n J_n, \qquad s_1+s_2+\ldots s_n=1 \tag{5-5}$$

Eventually, Job Assignment is an example of spatial planning. We talk about space in which the assignment is performed.) Spatial planning aims toward obtaining the best combination of the agents (virtual actuators) which are supposed to collectively perform their job.

This combination is supposed to be obtained by forming plausible hypotheses and testing them using the model of the system (by simulating motion of the system.)

Spatial planning starts with selection of the target-subgoal on the schedule submitted from above. After this, the plausible hypotheses are formed for job assignment. These hypotheses are evaluated and ranked (see Figure 5-5.)

JOB ASSIGNING (or "job distribution") starts with selecting the target point within the plan submitted from above. Then it forms hypotheses of job assignment. In order to do this it produces a number of tentative decompositions of the corrected plan (both before and after compensation) into the sequence of commands and verification of the results of planning at the next resolution level. The decomposition is done such that the total sequence could be resolved as a cooperative effort to be developed by several agents (subsystems of the adjacent level of higher resolution). After the higher levels of resolution are through with their BG process, PLANNER aggregates their results and decides whether they satisfy the constraints. If not, it starts the process of replanning.

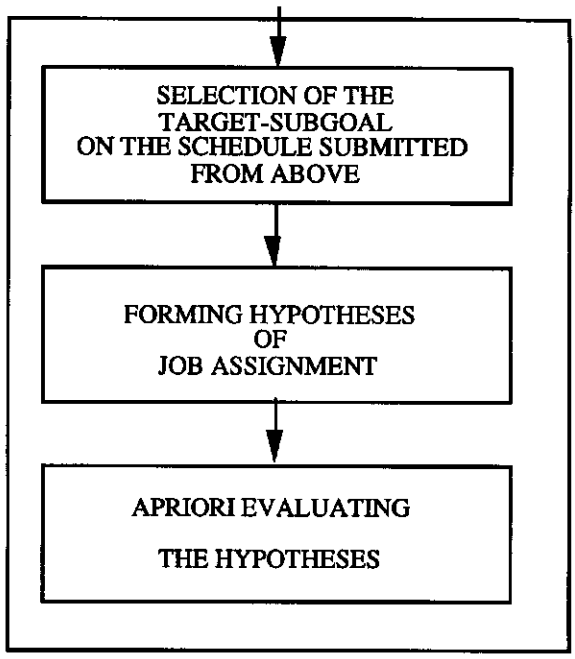


Figure 5-5. Spatial Planning (Job Assignment)

If one consider such examples as Integrated Manufacturing [105, 109, 111], RoboCrane operation [114] and/or Robot Manipulator [115], it becomes clear that the jobs cannot be distributed in advance "by decree." JA sub-submodule has to contemplate a variety of possible job distribution alternatives. Most of them based upon concurrent operation of agents. For example, machining in the RoboCrane case allows for multiple possibilities of job distribution among the winches, wheels, and positioning table motion. The acceptable solution requires minimization of the cost-function, which differs in different cases. It can be time, accuracy, losses of energy in the actuators, or cost of the operation. Any particular distribution of the job at a level that invokes processes of high resolution computation for verification of the rough alternatives proposed by the low resolution levels.

However, the external observer can judge the operation of this submodule by the set of temporal/spatial outputs of the BG-module that are issued to the subsystems (virtual actuators) as this diagram shows.

Assume that the output trajectory is obtained in the form of a piecewise curve in the multidimensional space with the forbidden zones (constraints).

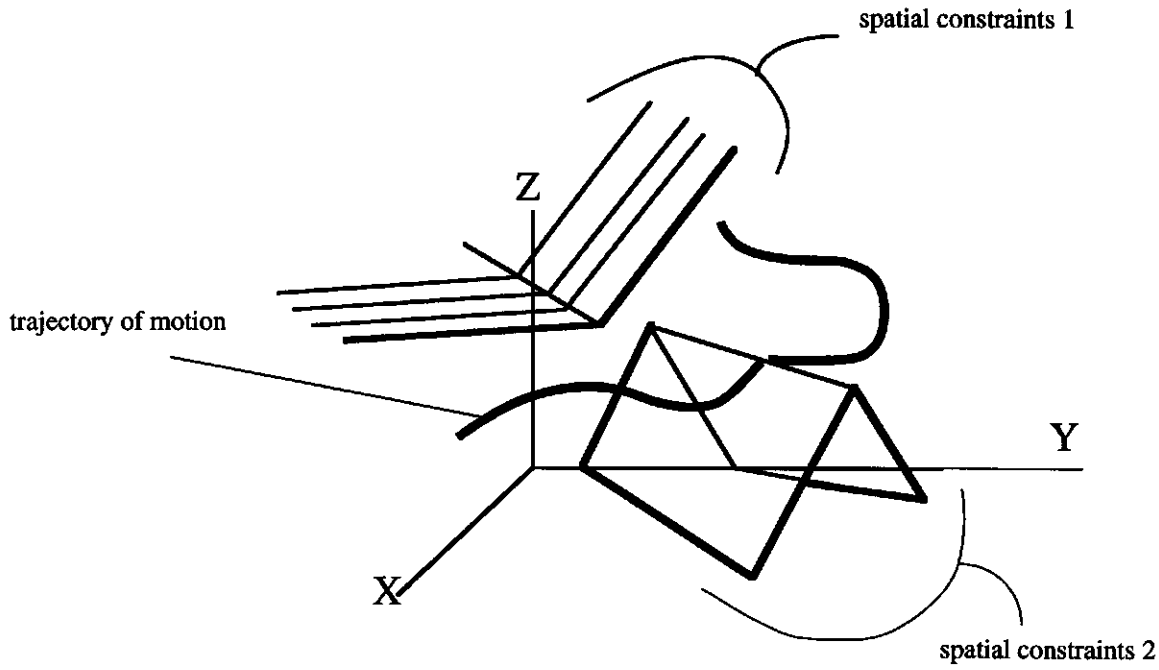


Figure 5-6. Geometry of Path Planning Among the Constraint Boundaries for the Robocrane case.

The output trajectory of motion eventually considered in the XYZ 3D physical space has its spatial constraints like obstacles shown in Figure 5-6. After mapping from XYZ space into the space of {six actuators}, we find that under the same 3D spatial constraints, the cost functions of time and electric motor losses, the trajectory turns out to be very different from the one "meaningful" one based on our human intuitions. However, the "right" trajectory can be obtained only by searching. The input search depends on the inputs we would be willing to use. In the case of NIST-RCS with automated planning, exploring all possible combinations of subsystems with their virtual actuators could be explored. The number of alternatives will be substantially reduced if the results of prior planning are known for the similar situations.

The operation of Job Assignment sub-submodule is the prerequisite for searching of the schedule alternatives. It presumes the existence of a limited number of discrete units at each of the output coordinates of the process and exploring all possible combinations. The degree of goodness for these alternatives can be determined by introducing a cost-function that is considered to be an additional output coordinate. To compute the cost function, the need in additional coordinates (states) can emerge, as shown in Figure 5-7, a. The objective, formulated as "searching for output

trajectories, and input command sequences, which perform the task (achieve the goal) with minimum cost-function (or cost-functional)," is similar to finding the optimum control process both at the input and the output. We will use search in the state space to solve the problem.

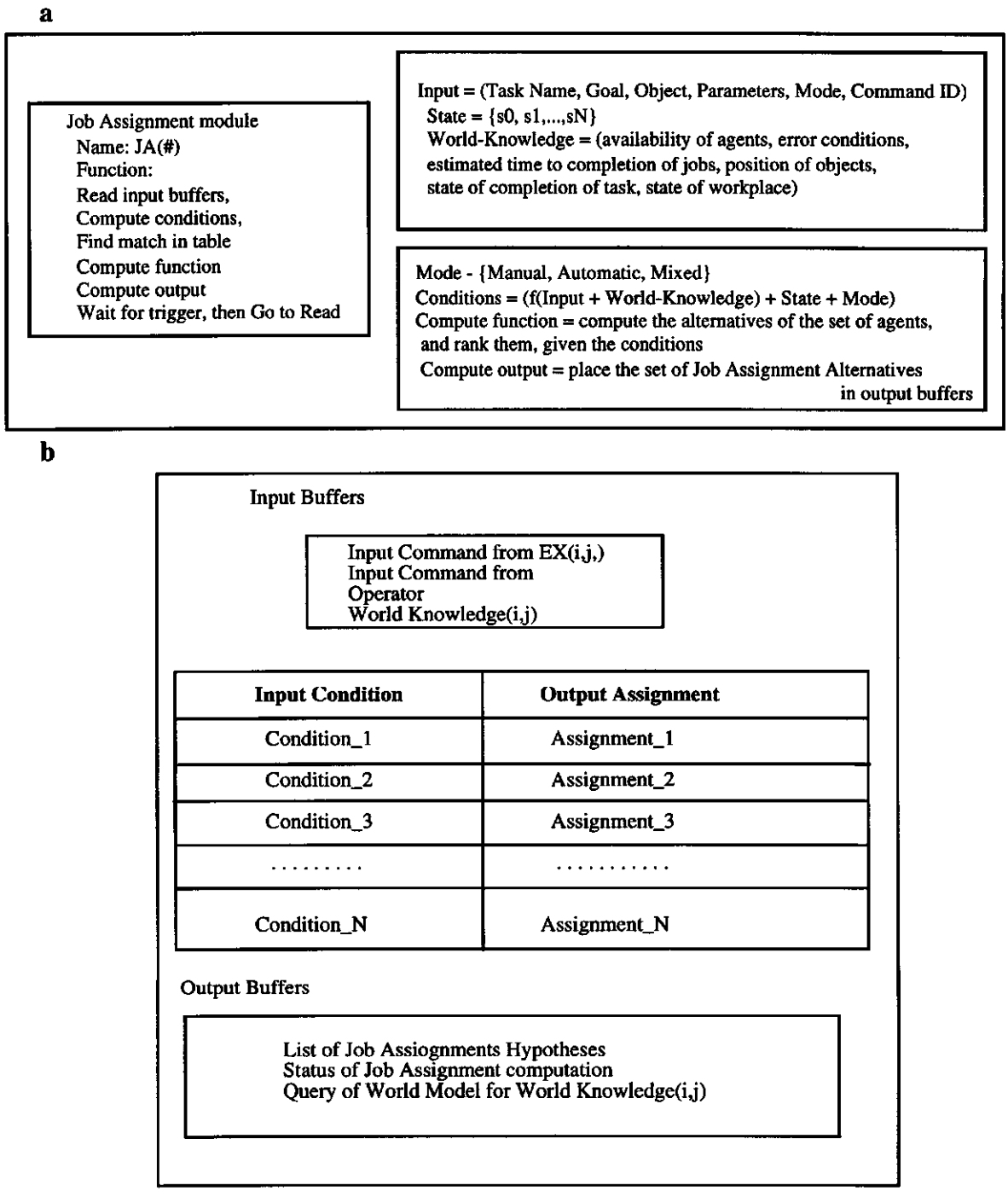


Figure 5-7 Job Assignment sub-submodules for spatial planning.

PLANNER determines the desirable output functioning. The input commands are not necessarily determined as a part of JA-SC joint searching process. In this case, the subsequent "inverse" algorithm should be used to compute the inputs. However, if the dynamic system is mathematically non-inverstible, the technique of "forward testing" is used to plan the input strings of commands. The essence of this technique is simulating the processes at the particular inputs. Instead of storing the analytical model, one can store a multiplicity of prior experiences which will be utilized as the results of forward testing[65] .

The algorithms of search by forward testing of the model provide the opportunity to eliminate several critical theoretical and computational problems. The following important advantages can be listed:

a) the need for the implementation of the inverse system for planning is removed;

b) the difficulties associated with the variational techniques are avoided (in most cases, these difficulties preclude the optimum planning);

c) the computations are simplified because approximations can be introduced instead of searching for precise solutions.

In using forward search, the first approximation, which must be used, is the conversion of a typically continuous time system into its discrete time equivalent. Because nearly all practical problems involve sampled data systems, this is not actually a restriction. By selecting an appropriate sampling period, it is possible to vary the structure of the discrete time equivalents of the continuous system and thereby change some of their properties. The effect of this approximation on the computed optimum state-space trajectory and its inverse will be that the desired function and the output due to the computed inputs will agree only at sampling instants. This approach requires that the invertible mappings among the states and the labels of tasks be explicated.

We do not differentiate between combinatorial search in the space of tasks and combinatorial search for the motion trajectory. We believe these processes are different only in the level of discretization and are unified under the label "searching for the output trajectory."

[65] SCRIPTS and SCHEMATA databases are obtained as a result of prior expertise. The technique of building these databases is outlined in two separate reports "Planning" and "Learning."

5.2.4 SC searches for the best schedule

Evolution of simulated systems functioning in time generates the schedules which are submitted subsequently to the agents at the higher levels of resolution.

Scheduling is performed by searching for the minimum-cost trajectory of motion in the state-space. The cost of all simulated schedules is computed by the submodule VJ. The search for the alternative schedules is performed via combinatorial construction (synthesis) of the output trajectories of motion at the selected level of discretization. This construction is equivalent to the process of simulating the system's motion under hypothetical conditions. As a result of this search, a set of 2-3 near-minimum-cost trajectories of the output motion are obtained.

After this, the set of near-minimum-cost output trajectories is being inverted using the dynamic model of the system. As a result, we receive a set of control schedules that allows us to check the convenience of the control to be applied. These schedules are simplified, linearized, and discretized according to the specifics of the system.

The set of schedules can be illustrated as shown in Figure 4-8. One can see that schedules of different levels of the hierarchy differ in the time discrete that separate consecutive events. This time interval is reduced when the resolution is higher. Schedules differ also in the overall horizon; the horizon grows when the resolution becomes lower.

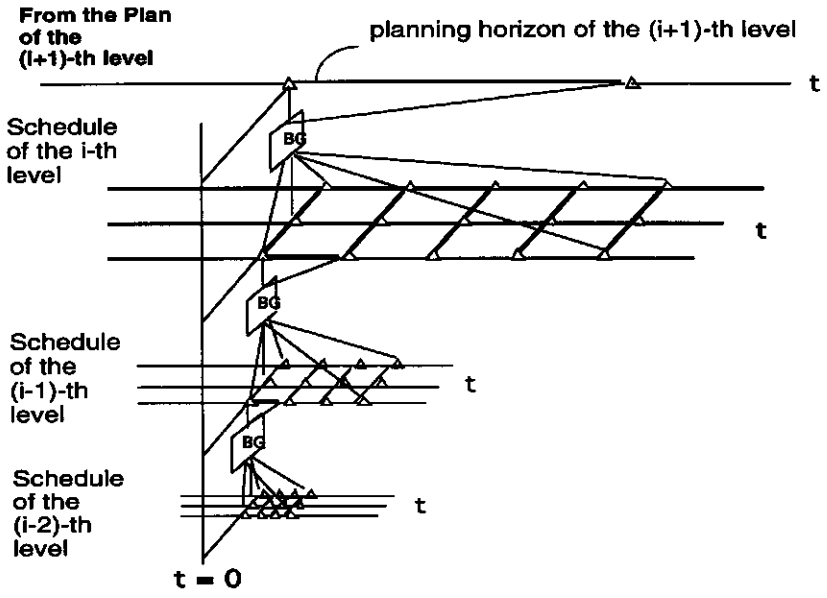


Figure 5-8 Scheduling as a Part of Planning

Notations: t-time axes, BG -behavior generating modules of three adjacent levels: (i-1)-th, i-th, and (i+1)-th ones

Certainly, all planning processes at all levels of resolution cannot be initiated simultaneously. Later, the "lead time" is introduced and demonstrated. The essence of JOB ASSIGNER/SCHEDULER functioning will be clear from the following description.

The state-space for the future searching of the best motion trajectory is bound from above and from below at each level of resolution by the scope of attention (any area beyond the scope of attention won't be required for searching) and by the smallest distinguishable element of the space (determined by the scale, granularity).

Synthesis of the State Space trajectory results in a "time-tagged" spatial curve which can be easily recomputed into the time axis for each subsystem of interest (scheduling). The upper bound of space will be translated into the planning horizon (in time). When the output trajectory is found, and the input command sequences are found, the Schedules can be obtained for both inputs and outputs. They must be made consistent with the list of precedence conditions which are supposed to be listed for any particular environment. If one or more conditions from this list are violated. the constraints are introduced into the state space of search and the search is repeated (see sub-subsection 5.3.2 Replanning).

5.2.5 PS makes the final choice or initiates correction

Regular updating of models and correction of parameters is required, since the state-space is discretized, parameters contain a substantial stochastic component (possible error), and the tree of search is frequently pruned to reduce complexity of computations. After 2-3 near-minimum-cost schedules are selected, the coefficients should be varied in the "equation of job distribution." This allows for refinement of the optimum trajectory. The refined command schedule, together with the output expected trajectory, is submitted to the output as the "best" plan which includes:

1) a set of time schedules (of feedforward control sequences of commands) for the selected set of actuators (virtual actuators)

2) and a set of corresponding anticipated output trajectories.

5.3 Functioning of the PLANNER-submodule

PLANNER shown in Figure 5-3 is not sufficient for practical cases. Two additional capabilities should be added using prior experience of planning and coordinating parallel plans.

After the Command arrives from the upper level, the submodule performs the following operations (See Figure 5-9):

a) BROWSING in the library of solutions (see SCRIPTS and SCHEMATA databases in Appendix);

If the results of browsing are unsatisfactory, the search in the state space should be initiated;

b) SEARCH in the state space (by the SEARCH-PLANNER[66])

As the desirable trajectory of motion is found at a level of the intelligent system it is considered a "PLAN" at this level. On the other hand, given all uncertainties of the initial information, newly arriving sensor data, intervals of discretization (inaccuracies inflicted by resolution), this PLAN is, actually, no more than a fuzzy suggestion for the adjacent level with higher resolution of the subspace where the next search should be conducted. This trajectory of motion should be regarded as a "pipe", as a "stripe," as a narrow enough space for the subsequent search. If the motion is performed within this stripe it is already not bad, it is already s "satisficing" solution. This is why we might refer to this plan as to a "stripe of satisficing motion."

The boundaries of this stripe are the constraints. If the real motion is executed within these constraints, no replanning would be required.

c) Coordination of the SCHEDULES which entail the results of Spatial Planning[67] .

The results of browsing are based on the existing solutions menu. If the appropriate solution does not exist, or has a low level of "goodness," the search is initiated. The results of the search are accepted if they have a higher value of goodness. In this case, they are also stored in the solutions library. The selected planning solution is submitted to the next level for exploration, where the similar sequence of activities is performed. The results from the next level should confirm, or reject the results of the planning at the level of consideration. Usually, the vocabulary of the lower level is richer and contains information about the agents performers. By selecting the best possible alternative of the plan, the initial stage of coordination is performed. The final stage of coordination is done during the stage of execution.

5.3.1 Lead time diagram

PLANNER receives the plan from the upper level (after its decomposition by the JA of the upper level) and determines the trajectory of desirable motion in the state space. This means that PLANNER has some "cost-function" and is capable of searching for the "best" alternative of motion to be executed. Because each level of the higher resolution is supposed to initiate its planning operation only after the upper level completed these procedures, the "starting points" of planning processes at all levels of resolution can be illustrated as shown in the Figure 5-9.

One can also see from this Figure that the process has two waves: top-down and bottom-up. The search is done by constructing combinations of possible strings with the consecutive comparison of the results of this synthesis. Because other planners at a level are doing a similar job for their part of the functioning, the search procedure should be done in cooperation with other PLANNERS. All PLANNERS exchange the current information about the intermediate results of

[66] See a separate report on "Coordination".

[67] Remember, we are talking about (i-1)-th resolution level components of the i-th resolution level system. Existence of these components nested within the i-th level of resolution system, is a part of the knowledge of each i-th resolution level.

planning among themselves. Of course, the parallel processing is presumed.

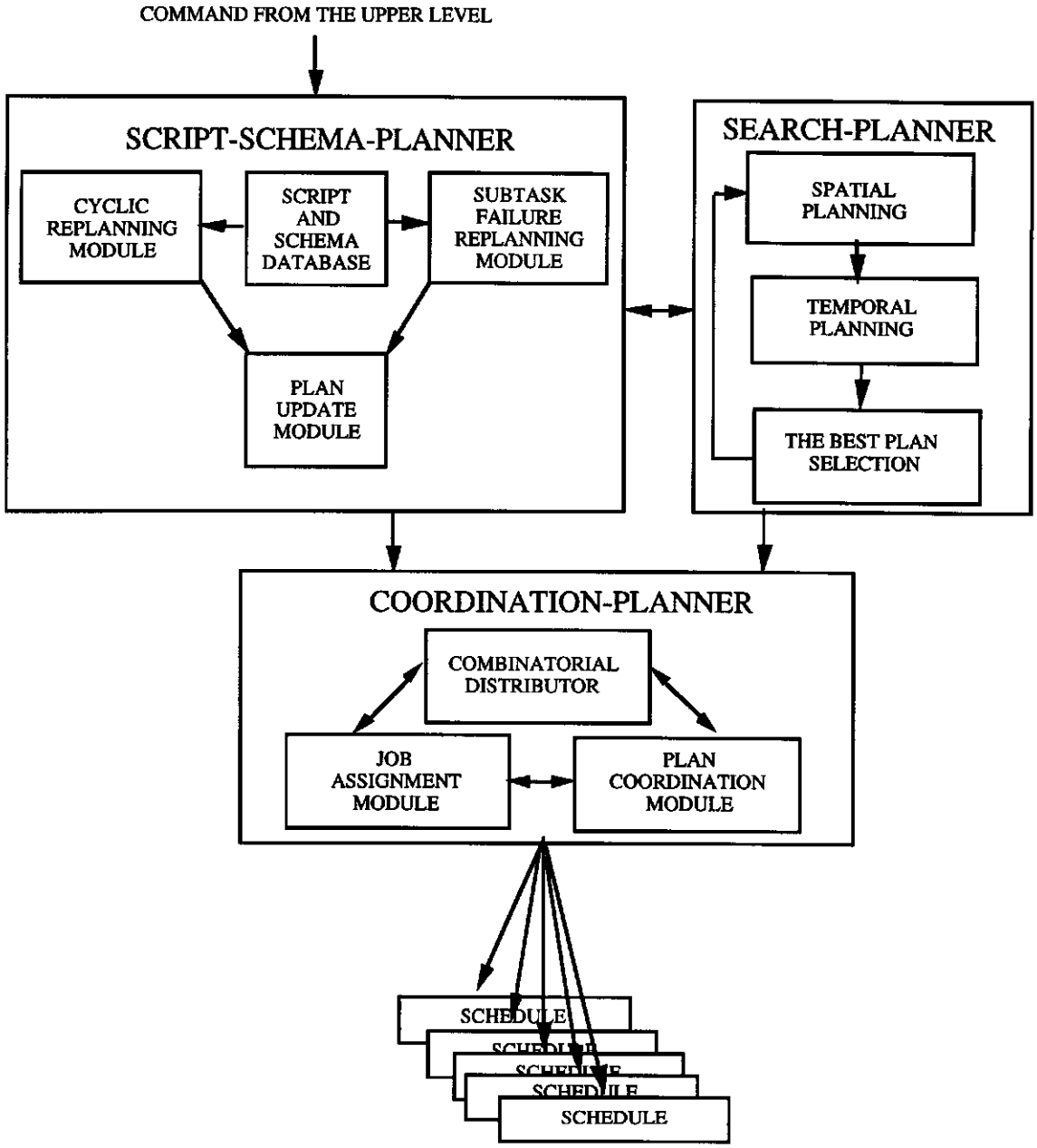


Figure 5-9 A Variety of Methods by which a Planner might be implemented

Notations: Script-schema is a package for the Linguistic Rule Based Planner, Search-planner is synthesizing the best trajectory in numerical representation, Coordination-planner negotiates possible alternatives

Completion of the lowest level planning initiates real time control execution, which

generates control assignments to the lowest end controller with the maximum frequency available. At each control level, new PLANNING is initiated. This occurs either immediately after the previous PLANNING is finished, or if the warning information is coming that the real motion at the level below is not executed as expected. The boundaries of the "stripe of satisficing motion" were crossed, and REPLANNING is required.

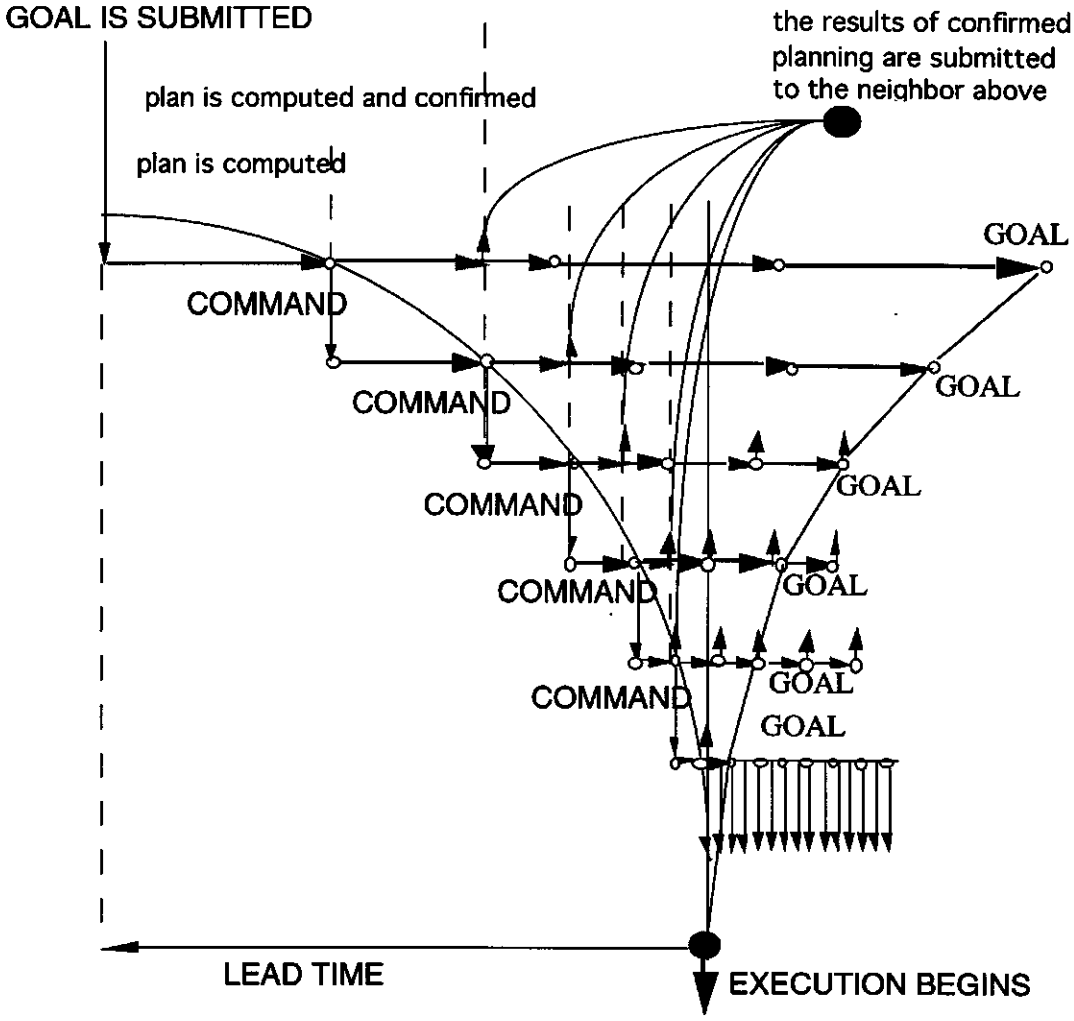


Figure 5-10 Evaluation of the Lead-time Required for Planning

5.3.2 Replanning

From the Lead Time Diagram, one can see that the process of Planning at the i-th level is completed after the plan is confirmed at the level below. While the level below is operating, a concurrent process of planning ahead can start since the fraction of rejected plans normally should not be too high. After the confirmation arrives, it should be submitted to the upper level. If the

confirmation is not coming, then the diagnosis should be performed to correct the processes of browsing or search in the state space.

Replanning should be done after these corrections are performed. It can be done in a smaller subset of the state space. Therefore, it is expected to be a faster process than the initial planning. However, if the initial plan was accepted, the search for the better alternatives should continue. The experience of search in the state space shows that the trade-off is admitted. Limited envelope of the state space speeds up the search, while the best solution might be missing. Therefore, if the plan is confirmed and the new command has not yet arrived, the process of replanning is being run for all zones of the trade-off.

5.3.3 Planning as a time related process

The process of PLANNING at the upper (low resolution) levels ends at a very early time, much before the actual motion should start. This can be explained by the need that all levels should complete their PLANNING processes before the motion starts. The real-time motion can start with a particular time delay equal to

$$\Delta t_d = \sum t_{pl.i} \quad i=1,2,\ldots,n$$

where $t_{pl.i}$ -is time of planning at the i-th level of the BG-hierarchy,

n - is the number of levels in the hierarchy of control.

As soon as the upper level planning is completed, the next level starts its process of PLANNING. Completion of the lowest level PLANNING initiates the real-time control, which generates control assignments to the lowest-end controller with the maximum frequency available. At each control level, new PLANNING is initiated. It may begin immediately after the previous PLANNING is finished. It may be initiated if the alarm comes that the real motion at the level below is executed not as expected, and the boundaries of the "stripe of satisficing motion", or the "stripe of reliable planning" were crossed.

The time span between the two curves characterizes the time uncertainty of the level of control. The flow of regularly arriving information about motion execution is being absorbed by the system only in its learning part. Otherwise, the regular PLANNING starts after the previous PLANNING is finished. Or REPLANNING starts as soon as the alarming information has arrived.

The curve of PLANNING HORIZON determines the time-tag of the outcome predicted by the algorithm of planning. The upper (lowest resolution) level of the NIST-RCS hierarchy has the largest horizon, the low end controller has the value of horizon determined by the frequency of control commands:

$$\text{horizon} = {}^{A}/_{\text{frequency}}$$

where A is a number depending on a particular domain of application.

5.4 The algorithm of planning

This algorithm specifies Search in the State Space as the method of finding the desirable state space trajectory. This entails the complete set of required results (spatial plans with job distribution among the agents and the temporal schedules). The fundamental issue relevant to the algorithm is how the goal is assigned and represented. In the vast multiplicity of cases, the goal is considered to be a known state in the future.

In many problems, this state is unknown before the vicinity of it has been reached. Then, this vicinity is regarded as a known state at the lower level of resolution. For example, the goal can be assigned as a "safe position on the northern slope of the mountain Z". After the slope is achieved, the procedure of planning should be performed again since the goal-state is supposed to be known.

Often, we are dealing with problems of control in which the goal is assigned not as a state but as a condition to be satisfied (and this condition can be satisfied in a variety of states). For example, the goal can be assigned as "being in the visibility range from the moving objects of a particular type (in a particular region)." Then, the low resolution goal is to achieve this "particular region." The high resolution position within this region should be defined based upon other criteria. Then, searching for moving objects should be initiated. Areas of high probability to detect moving objects should be recognized and found. The next high resolution planning will happen upon detection of a moving object.

An example of the algorithm is given in **Appendix IV.**

[After the present state is known, after the goal state is assigned, after the cost-function is clarified] **at the i-th level of resolution DO**:

Step 1. Find the alternatives plans recommended by the library of stored solutions. At this particular level,compare them under conditions of the assignment and select one that is "the best".

If the solution is found, exit successfully.

Otherwise **go to**:

Step 2. Search in the state space for the sets of feasible trajectories for achieving the goal at the i-th level of resolution.

The search should be done both in the state-space and in time within the search envelope of the i-th level. It should be determined whether achieving this goal requires any cooperation of the

subsystems existing at the adjacent level of higher resolution[68]. Different alternatives of combination of the subsystems of the (i-1)-th level of resolution are usually available. Judgment about the preferable combination of the systems component should be done based upon the results of their planning. If there exists a system (or systems) of the i-th resolution level that plan their operation for themselves, and our system should cooperate, it should interact with these systems during the search.

Implications of the Step 1:

a) The search procedure should be prepared and executed. This search will look for a trajectory in the state-space which leads to the Goal from the initial state. It minimizes a cost-function or keeps the process within some specified boundaries (instead of minimizing the cost-function) or both. Selection of the states is determined by mapping from the space of problem into the space of components of the solution. The latter also affects the alternatives related to the cooperative agents of the next level of higher resolution which should be selected for performing the job.

b) It would be inefficient to repeat the search for a particular set of conditions if they are encountered again. Storing the prior "good" solution is presumed. It is an example of learning from experience. Many of the situations can be simulated off-line and the results of simulation with a sufficient degree of belief can be stored. It is an example of learning from the simulation results. Finally, similar situations could be experienced in other systems of this type. Some of the results with a sufficient degree of belief might be included in the storage. It is an example of learning from an expert. The procedure of search in the state space can be supplemented by the procedure of searching in the library of successful solutions known from the prior searches and experience of functioning, simulation, or expertise.

Step 3. Evaluate these trajectories by their "goodness" and the probability of success."Goodness" evaluations should be based upon the cost-criteria important for the system including the processes of cooperation with other systems.

Step 4. Store the limited number of trajectories with the best "goodness values," in a list. The length of this list can vary depending on the environment, the system under consideration and the computing power available.

Step 5. Within the best trajectory, assign a limited scope of it (horizon in time and the envelope in the state variables) for the consecutive refinement (i.e. translating the results of planning into the language of the adjacent higher resolution level.) The stochastic reality factors lead to the phenomenon that the degree of belief for the results of search (that this is "the best"

[68] See J. Albus, "A Reference Model Architecture for Intelligent Systems Design," [107].

trajectory) gradually decreases along the time axis for the trajectory received as a result of planning, since our belief was based on an "open loop" execution during the selection process.

Step 6. Apply the subset of the planned trajectory (within the limited time horizon) to the virtual actuator and execute it using feedback information for on-line compensation. If this was the 1-st level of the NIST-RCS, consider the job finished.

Step 7. Submit the following information about the trajectory after performing the compensation to the adjacent level of higher resolution (or, which is the same, the (i-i)-th level below):

a) The final "point" of the trajectory designated by the limited scope assignment from the Step 5. (The point is actually described as an area of indistinguishability zone.)

b) The "stripe" of the state-space with boundaries confines the envelope for searching at the higher resolution level.

This is performed using translation of the results of planning into the language of the adjacent higher resolution level.

Finally: Execute at the (i-1)-th level **Steps 1 through 7.**
Continue until the schedule of operation demands execution.
Then, submit the output of level (i-1) to its actuator.

Continue computation at all levels simultaneously until the system shuts down.

5.5 Representing plan as a set of tasks

5.5.1 Planning as a generation of data structures

In most of the existing RCS materials, the results of planning are always being precomputed. This generates a great variety of useful, even adaptive behaviors. However, in general, precomputed planning can be done only in a limited set of sufficiently simple cases. Realistically, the hierarchy of task decomposition can be precomputed only in the epistemologically poor worlds. Yes, we know many examples of task decomposition in practice particularly in manufacturing. However, all these hierarchies are created ad hoc and contain many deficiencies. Apparently, we need a methodology of task decomposition that would be based upon a firm theoretical background. We believe that this book is a step toward a scientific methodology of task decomposition.

Generally, the task decomposition should be generated in real time by the process of planning within NIST-RCS architecture. The plan of a level (a set of tasks at this level) forms a solution for a problem formulated as its task at the upper level. But where are all these solutions

coming from? The mechanism that can be recommended throughout the NIST-RCS hierarchy starts with organization of a level via TASK FRAME which serves as a vocabulary for the search space[69]. It is illustrated in the diagram shown in Figure 5-11 (see also subsection 1.4 .)

The diagram illustrates how the process of browsing for SCRIPTS and SCHEMES develops. If the format of COMMANDS satisfies the Command Frame, then the Library of Task Frames will allow recovery for all available cases. Either the plan has been stored or the planning procedure has been recommended. The latter contains constraints and suggested subroutines for the algorithm of the state space search.

Task Frame defines the information required to plan and execute a task. Some of the parameters are assigned by the intervals. Their final selection is done immediately prior to execution. Some of the parameters are subject to be corrected as a result of Learning.

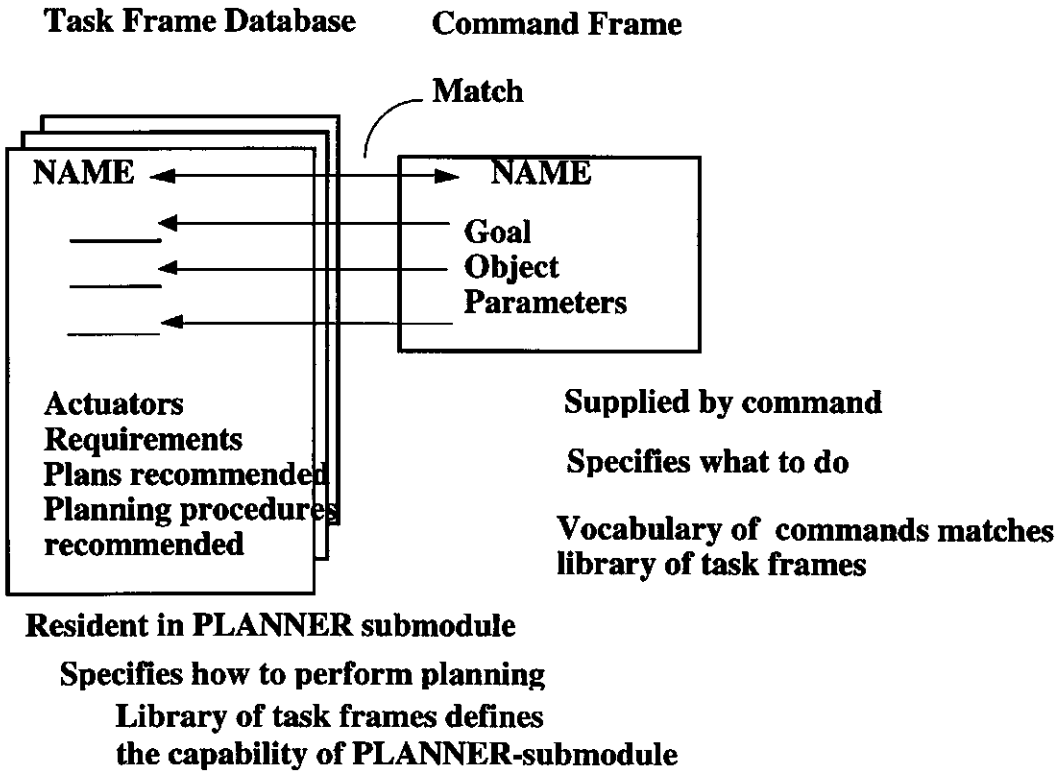


Figure 5-11. The Task-Frame

Definitions and explanations which are required for understanding the concepts of Task-Frame components are shown in the following table.

[69] Thus, we receive a new alternative definition of the plan (in addition to those found in Chapter 1 and in Appendix I): *plan* is a combination of *tasks* at the i-th level which are determined as a *solution* for the *problem* considered a *task* at the (i+1)-th level of the system's *multiresolutional hierarchy*.

Table

TASK NAME Name of the task

Goal..... Event or condition that successfully terminates the task
Object.....Identification of thing to be acted upon
Parameters.....Priority
Status (e.g. active, waiting, inactive)
Timing (e.g. speed, completion time)
Coordinate system in which task is expressed
Tolerances
Agents.....Identification of subsystems that will perform the task
Requirements.....Feedback information required from the world model during the task
Tools, time, resources, and materials needed to perform the task
Enabling conditions that must be satisfied to begin or continue the task
Disabling conditions that will interrupt or abort the task activity
Procedures.....Precomputed plans or scripts for executing the task
Planning algorithms
Functions that may be called
Planning Procedures.....Suggested envelope
Suggested method of successor generation
Suggested cost-functional
The law of compensation suggested.....Suggested error boundaries
Suggested gains
Suggested membership function

5.5.2 Two types of task decomposition.

In the programming paradigm, TASK becomes a command. It is defined as an instruction

DO<Task>AFTER<Start Event>UNTIL<Goal Event>,

where Goal is understood to be the assigned state ("result")

or

TASK COMMAND:= DO <TASK>
WHEN (START EVENT)
DO (TASK)
UNTIL (GOAL EVENT)
END-DO

The statement DO<Task> is supposed to be understood by the performer. It does so either by lower resolution level, which will DO the next decomposition of the TASK, or by the execution actuator, which transforms the label of TASK into ACTION. Strings of tasks can be visualized as control laws.

Command strings can be applied if the state is known ("start event",) and when the final state is determined for this particular control command ("goal event".) So, it is clear that the hierarchy of TASK DECOMPOSITION (H_T) can be generated by the BEHAVIOR GENERATION SYSTEM if two additional hierarchies are given: a hierarchy of CURRENT STATE DECOMPOSITION (H_S) and a hierarchy of FINAL STATE, or GOAL DECOMPOSITION (H_G).

Then the couple (H_G,H_S) constitute the problem to be solved and H_T is a solution for the problem. The planning/control algorithm is supposed to produce a transformation

$$P/C:(H_G,H_S)\text{-->}H_T$$

A set of n consecutive tasks (subtasks) for the i-th level represents a solution for the task T_{i+1} of the upper (i+1)-th level

$$T_{i+1}\text{-->}P_i\text{--> }\{T_i(t_0, t_1);T_i(t_1, t_2);..;T_i(t_n, t_{n+1});..;T_i(t_{f-1}, t_f)\}\ ,\ 1\le n \le f\ ,$$

is called PLAN[70] , or P_i to be performed at the i-th level when the task T_{i-1} should be executed at the (i-1)-th level of the hierarchy of task decomposition. The task of the (i-1)-th level under consideration T_{i-1} can generally be decomposed into a set of m parallel plans:

$$T_{i-1}\text{ --> }\{P_{i1}(t_0, t_f);\ P_{i2}(t_0, t_f);...;\ P_{im}(t_0, t_1)\}$$

Consider the functioning of the PLANNING subsystem at a level of comparatively low resolution where it functions often cannot be trivialized[71] .

5.5.3 Types of concurrent plans and consecutive tasks.

Two subsets of the concurrent plans can be considered:

a) plans that are undergoing further decomposition within the hierarchy under consideration (denoted $\mathbf{P_{im}}$ where m stands for "main"), and b) plans that are not decomposed within this hierarchy. The latter can be divided in two groups. The first one is decomposed as a part of

[70] See references [92, 93].

[71] The phenomenon of intersection of the task-hierarchies is a typical phenomenon in the systems of intelligent control. Clearly, the same task can have different cost-function in different hierarchies.

intrinsic activities of the resolution level under consideration. It is denoted $\mathbf{P_{i=1}}$, which means that for this branch of decomposition, the i-th level of the main hierarchy is the 1-st level of the branching hierarchy. The second one can be decomposed only within another hierarchy which **intersects** the hierarchy under consideration[72]. It is denoted $\mathbf{P_{ix}}$, where x should be selected depending on the particular name of the intersecting hierarchy. Intersection of the hierarchies is illustrated in a graphical form. The system under consideration N is part of two different hierarchies built upon different S-K-P/C sets. This is a very frequent situation.

Two subsets of the consecutive tasks can be found within a particular task-string: a) tasks that are being furtherly decomposed within the hierarchy under consideration (denoted $\mathbf{T_{im}}$ where m stands for "main"), and b) tasks that are not decomposed within this hierarchy. The latter can be divided in two groups: those decomposed as a part of intrinsic activities of the resolution level under consideration (denoted $\mathbf{T_{i=1}}$), and those that belong simultaneously to another hierarchy, which **intersects** the hierarchy under consideration (denoted $\mathbf{T_{ix}}$ where x should be selected depending on the particular "name" of the intersecting hierarchy). One can see from Figure 5-12 that a level can be a part of two additional hierarchies simultaneously, while other levels of both hierarchies have nothing in common. Many examples of intersecting hierarchies can be found in man-made and/or natural hierarchies.

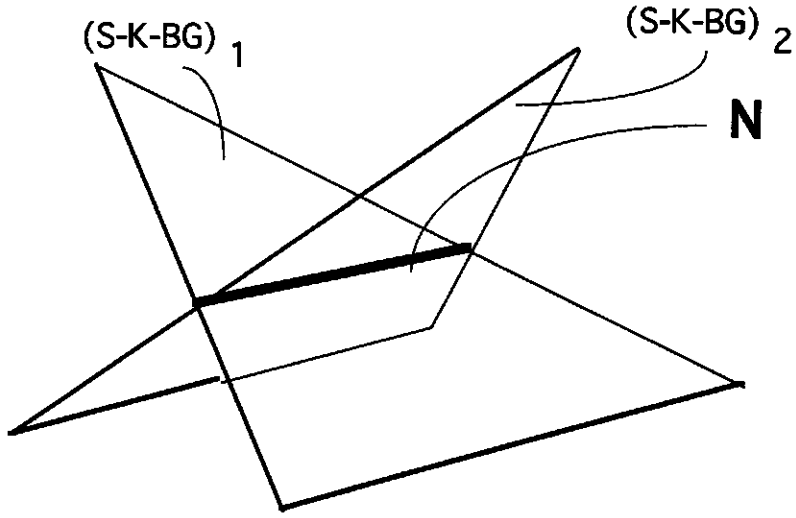


Figure 5-12. Hierarchies Which Have In Common Only A Limited Set Of Aspects Pertaining To A Particular Level (N - the aspect of commonality)

Parallel plans talk to each other via cooperating tasks. There are tasks where factors of cooperation should be taken in consideration as follows:

[72] Operator of cooperation should be considered in a broad sense, e.g. we can talk about "negative cooperation" in interaction with an adversary.

DO<Task>AFTER<Start Event>UNTIL<Goal Event>
AND SIMULTANEOUSLY WITH<Parallel Event(s)>

or

TASK COMMAND:= DO <TASK>
WHEN (START EVENT)
SIMULTANEOUSLY WITH (PARALLEL EVENT(S))
DO (TASK)
UNTIL (GOAL EVENT)
END-DO.

The sub-operator of cooperation can in turn be decomposed correspondingly .

Finally, each level of the hierarchy can be constructed, or can be processed in a hierarchical manner if it brings computational advantages within this particular level. In this case, we refer to computational hierarchies which are orthogonal to the hierarchy of the overall system architecture.

6. EXECUTOR: Its Structure and Functioning

6.1 Processing the Results of Planning

Planner is working under an assignment of remote goal. It aims to distribute the job among the virtual actuators (job assignment, or spatial planning) and scheduling (temporal planning). Together this gives the most efficient functioning of the system. It searches among all possible alternatives. This operation is done by using models from WM. However, these models become obsolete very quickly. Even if nothing changes, the models are erroneous due to the errors of discretization, generalization, assumptions.

Certainly, the models, together with discretization, contain errors due to the prior generalization and limited resolution which make all models flawed. The real situation (or "virtual situation" at intermediate levels) will differ from the expected one.

The results of spatial and temporal planning (job assignments and schedules) are formulated in the terms of the outputs. Task decomposition generates tasks. Each task is a description of the desirable result of functioning together with the suggestion, how this functioning should be activated. If the triplet "Actuators-World-Sensors" is a particular "machine," the results of planning describe what should happen in the World, rather than input of the Actuator should be. If we substitute the triplet "Actuators-World-Sensors" by a single unit of "Plant" (which can be considered a "virtual Plant"), the Plant input should be computed.

Clearly, a module should be introduced. This handles these two needs:

a) to compute the input to the virtual actuator at the level of consideration, and

b) to compensate for the imperfectness of the model at hand.

Plan inverse.

The first need is satisfied by inverting the results of planning (the output of Plant) to the input of Plant. Actually, this is one of the possible JA procedures, which uses either analytical operator inverse expression (P^{-1}), or forward searching (if the system is represented in a non-invertible form). The projected input might have been already submitted together with the desired output as a part of the Plan. However, if it is not submitted, the Plan should be inverted.

Figure 6-1 shows a submodule FFC: "feedforward controller". In this Figure, FBC takes place of the PLANNER (as described earlier in Section 5). Indeed, the decomposed and inverted trajectory of motion should be received as a result of joint computations [JA+SC]. The results of inverse and decomposition are sent a) to the plant model for simulating the process, and b) to the PLANT, or next level (higher level of resolution) for executing the process.

Error compensation.

The second need is satisfied by feedback compensation. The results of the actual motion should be compared with the expected results. The expected results are part of the Plan, or more accurately, are part of the simulation results obtained from WM after simulation (see Figure 6-1). Because the World Model is constantly updated, the Plant representation at the moment of "execution" might differ from the Plant representation during the process of planning. This is why one might be willing to simulate the output motion using the updated model of Plant and compare the results of simulation with the actual motion. The simplest possible mechanism of comparison is shown in Figure 6-1. The information about the state is arriving through one of the two alternative paths: either directly from the Plant (I), or from the Plant model which is constantly updated via loop II.

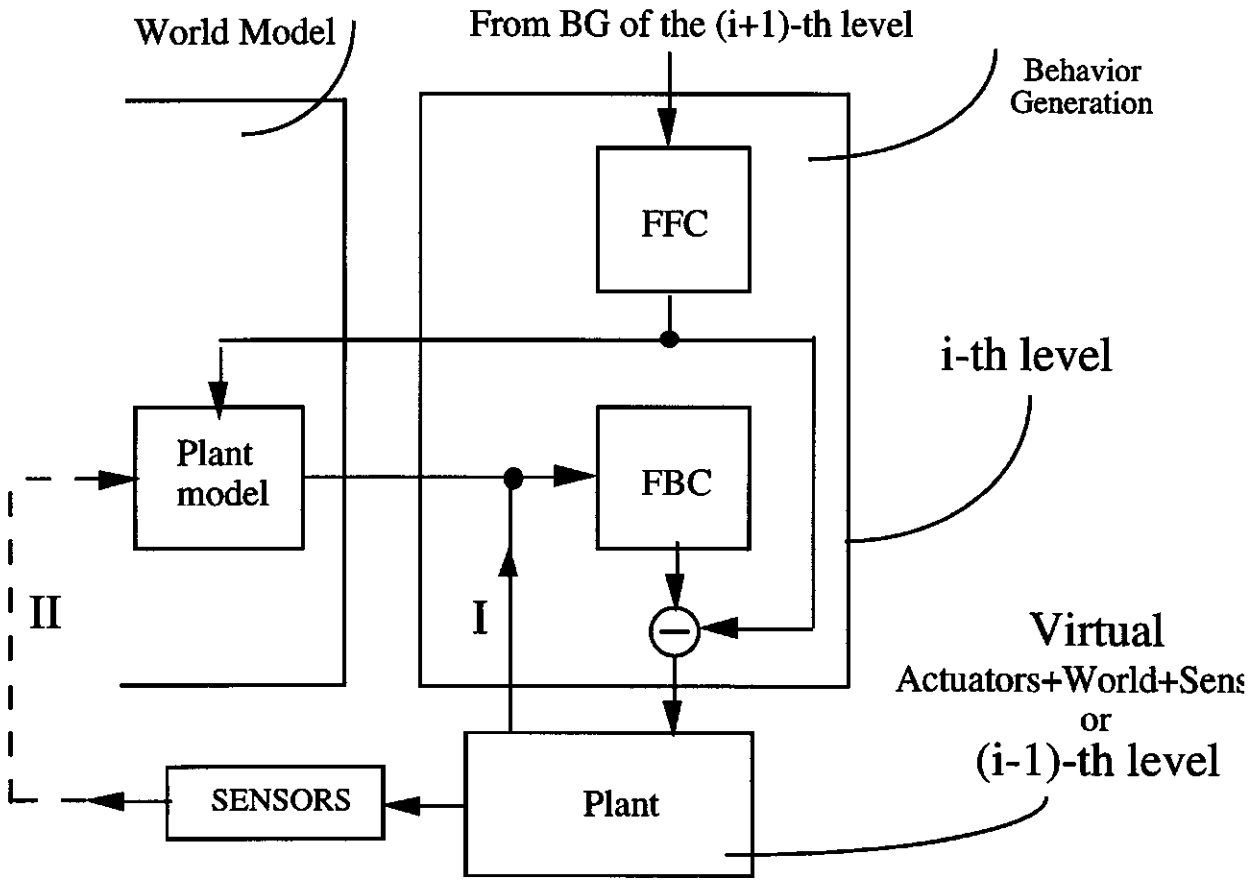


Figure 6-1. General structure of EXECUTOR

Both channels I and II should contain subsystem for state and output estimation (in the reality, levels of NIST-RCS are equipped with subsystems of "recursive estimation".

The difference between the Plan and the actual value of state variables ("error") is computed within the submodule FBC as "feedback compensation" and added or subtracted from the plan (the output of FFC ("feedforward controller".) The law of compensation admitted depends on the nature of the system and the level of resolution at which this compensation is performed.

One can see that EXECUTOR should adjust the results of planning to the reality of system functioning *on-line*. This why the need for feedback compensation arises. One should correct the control command quickly, even in advance. No off-line activities that can reduce the deviation is possible, when the reality of deviation intrudes unavoidably. EXECUTOR is the system which role is to deal with these predicaments. Some of the possible solutions employ the concept of "prediction." Evolution of errors is analyzed, and the future values of error are forecasted. In many cases, this can be performed with high reliability.

EXECUTOR interprets the PLAN into the sequence of commands that are submitted to the actuators, or to the *virtual* actuators for the level under consideration, and introduces a compensation command. It plays a role of the feedback controller (a **reactive on-line** controller typical for the conventional control theory and representing a pre-selected control law).

EXECUTOR contains a menu of control laws applicable in different cases. This menu allows for the most desirable compensation processes for the system at hand. The choice of the law of compensation is supposed to arrive from the Planner together with the Plan. However, one can arrange for a separate submodule as a part of Executor which would recognize the required law of compensation depending on the Plan and the real conditions of the World. The process of compensation requires proper distribution among the agents although the complexity of this is not as high as in the case of feedforward plan development.

Figure 6-1 can be better understood if we compare it with the representation that is common in the area of control theory as shown in Figure 6-2. The differences arise because we try to take care of phenomena that are beyond the attention of theory, such as aging of the model. The quality of this model (the measure of adequacy of the model, how well it represents the system)

Some information for the EXECUTOR is contained in the TASK FRAME. As a result, EXECUTOR compensates for the errors of planning if the sensor information demonstrates undesirable deviations. When the deviation exceeds the predetermined boundaries, EXECUTOR informs PLANNER that the procedure of replanning should be performed. Finally, to introduce compensation timely, EXECUTOR can be equipped with a built-in predictor of the process of deviation development. This will reduce the final error of the ELF functioning.

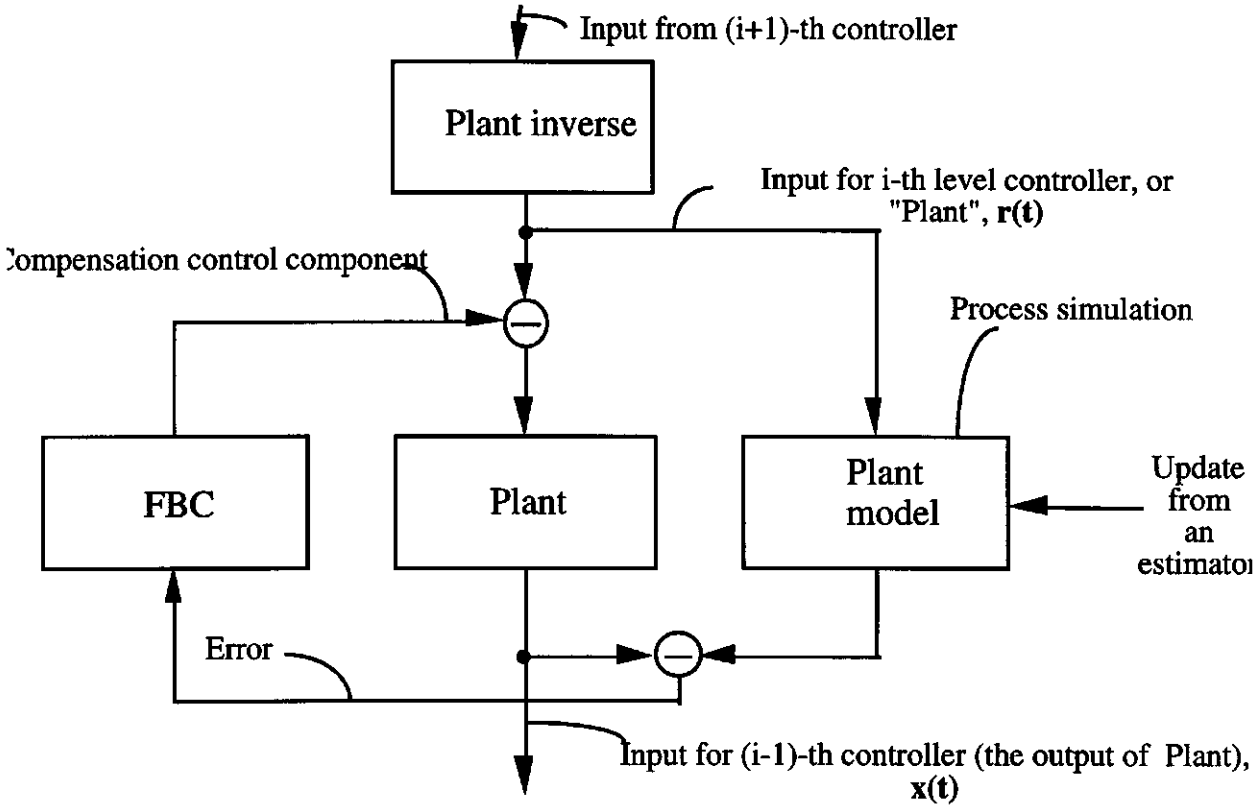


Figure 6-2. Interpretation of PLANNER and EXECUTOR in terms of control theory

6.2 The Structure of EXECUTOR

The function of EXECUTOR is to provide for the on-line support of the system operation by introducing corrections into schedules. The inner structure of EXECUTOR is built according to the account of its functioning as it is described in Figures 6-1 and 6-2. Its structure is shown in Figure 6-3. This diagram presumes that PLANNER has submitted both the set of required output trajectories, as well as the set of recommended input schedules (input commands). Current results of process simulation have arrived from WM.

Then, the anticipated output trajectory is compared with the current functioning, and the difference is estimated ("error".) "Error" should not be interpreted as a difference between the anticipated trajectory and the current measurements. The submodule of error estimation might include various methods of prediction.

Based upon the "menu," a control law is chosen, and the correction to the predicted feedforward control command is computed. This correction is used to supplement the control commands that arrive from PLANNER together with anticipated output trajectories, or can be computed from the anticipated output trajectories (the "plan") by using the "inverse" submodule

(shown in Figure 6-2).

Among the existing alternatives, EXECUTORS based upon multiresolutional variable structure controllers and multiresolutional Kalman filters have good perspective for the future in the industrial applications of NIST-RCS systems.

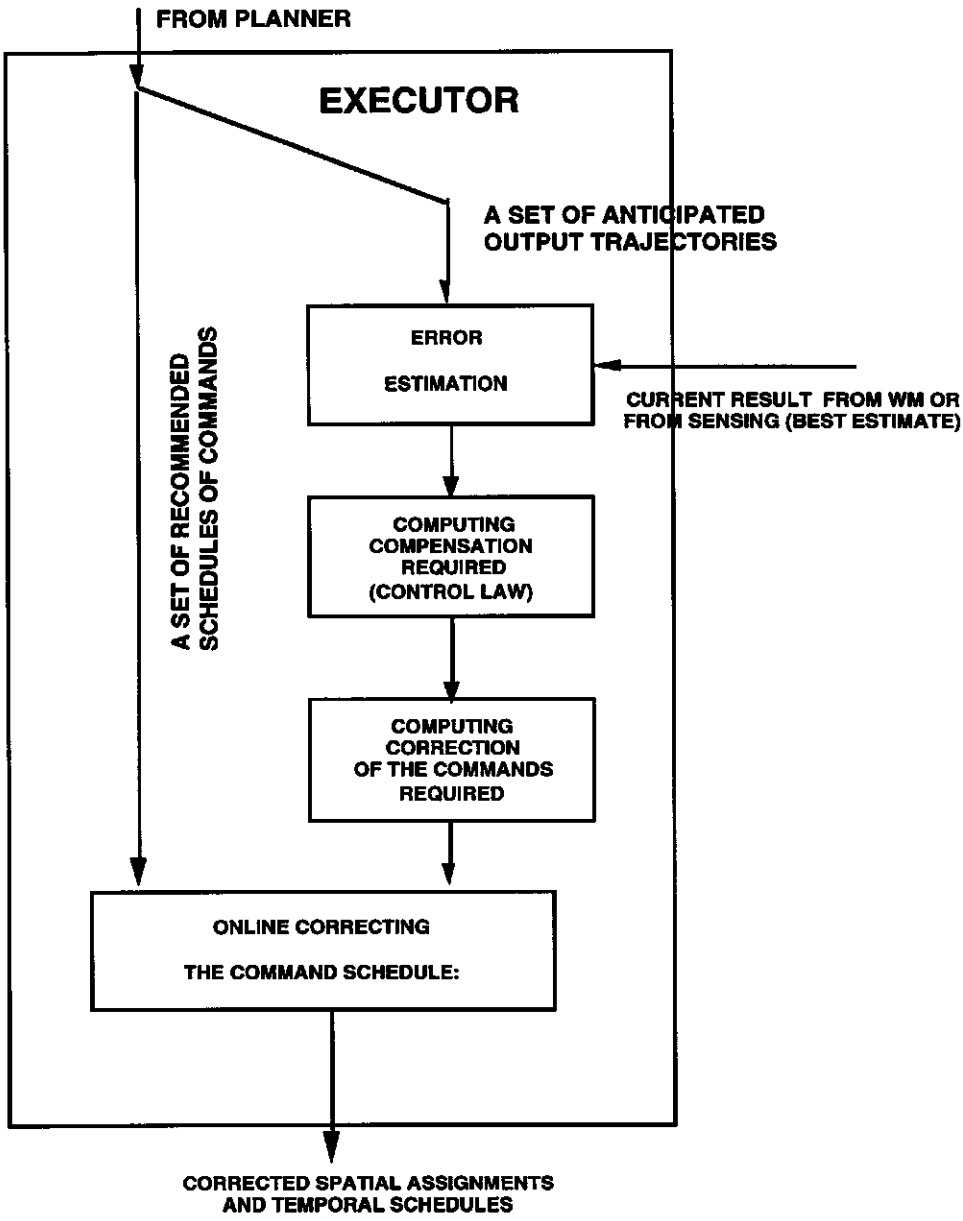


Figure 6-3. The inner structure of EXECUTOR, Type 1

The alternative of EXECUTOR including the "inverse" submodule is shown in Figure 6-4. Here,

the input commands are not obtained from PLANNER. They should be restored in EXECUTOR by using the inverse operator. Then, both the error estimate and the control sequence arrive to the "control law" submodule where the corrected control sequence is computed.

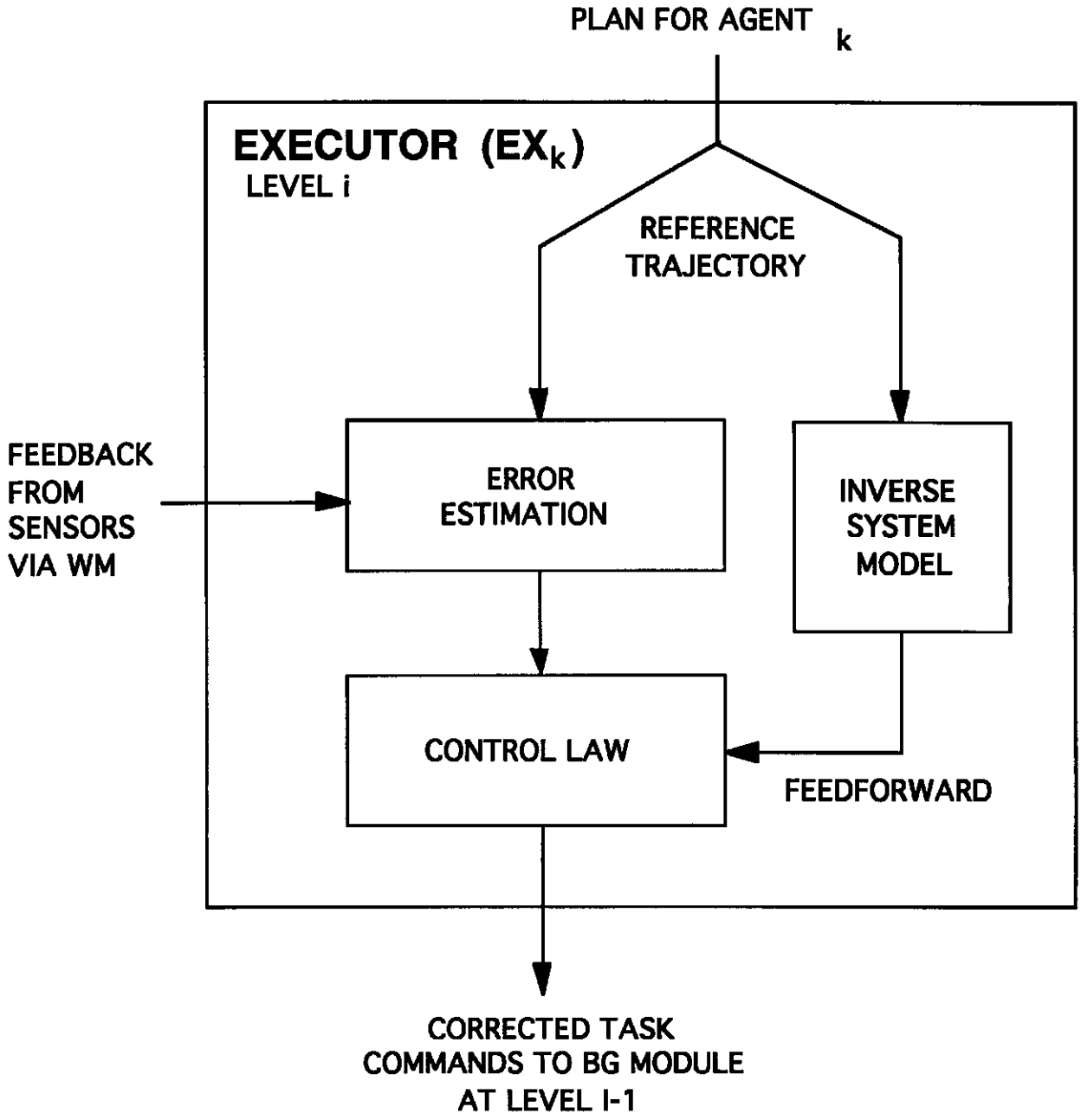


Figure 6-4. The inner structure of EXECUTOR, Type 2

Other types of EXECUTORS can be found in practice too.

6.3 Operations of EXECUTOR

In this subsection, we will discuss the functioning of Executor in more detail. This is how the subsystems of EXECUTOR operate:

Error Estimation.

The anticipated output trajectory (the "plan") is compared with the actual result of sensing. There are many techniques of estimation, and under different circumstances, some of them are more preferable than others. The sequence of the estimation process is illustrated in Figure 6-5. Estimator gives an *anticipated error* at its output.

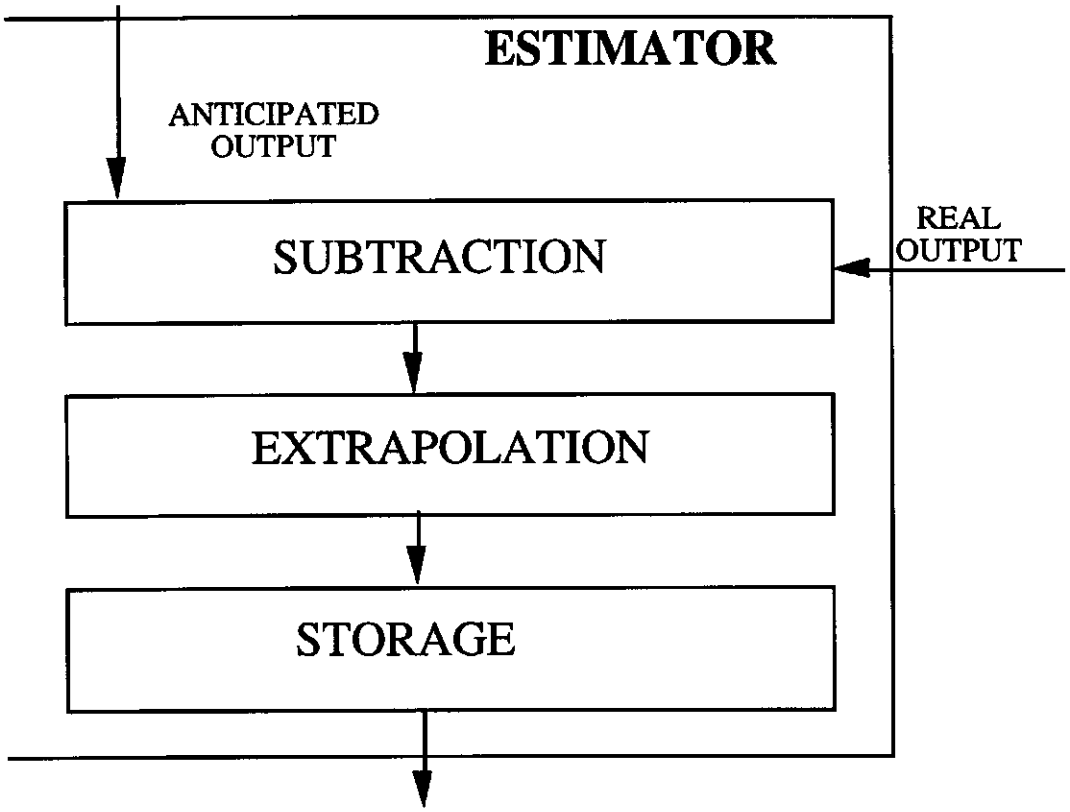


Figure 6-5. The structure of ESTIMATOR

The key element of ESTIMATOR is the sub-module EXTRAPOLATOR. In all cases of estimation, the measurements are compared with their expectations based on a particular theory of the process development or algorithm of prediction. The expectation of the computed error depends on the concrete system. The required correction of the control command is predicted based upon statistics of the measurement and assumptions about particular properties of the

system: its ability to respond to particular statistical characteristics of the error function. Since all estimation results can be assigned some degree of belief, the predicted control command should be trusted with a particular degree of belief also. It has been demonstrated that in a hierarchical system, the total prediction is more adequate than if estimation is done at a single resolution level. Various schemes of recursive estimation seem to be an appropriate solution for many resolution levels.

Computing the compensation required.

Depending on our anticipation of the error of our system working in a particular environment, a particular control law can be suggested. This control law allows us to properly compute the value of control compensation. (There are many reasons to expect that in the case of RCS, the *variable structure equivalent control law* has substantial preferences before other alternatives). Variable Structure Controllers can be interpreted in many applicable ways: as a "bang-bang" controllers for minimum-time systems, as fuzzy control system with a hierarchy of fuzzy convergence to the assignment. Estimator is not involved in computing the complete control command. It should compute only a part of it, the error compensation.

On-line correction of the command schedules.

In this sub-subsystem of the subsystem of EXECUTOR (see Figures 6-3 and 6-4) the on-line correction is introduced to the flow of command signals sent to the system of actuators, or (virtual actuators). As a result, the final version of the schedules arrives at the corresponding control system right-on-time, at the moment when it should already be applied.

This operation can be considered similar to a reactive response of the "intelligent agent" as presented in the literature on reactive control of robotic intelligent agents with preassigned "behavior." Thus, EXECUTOR with the ACTUATOR at its output can be considered an analog of intelligent agent.

6.4 Temporal functioning of EXECUTOR

Two cases of interest are shown in the Figure 6-6

a) when the new information arrives at a higher frequency of sampling than the frequency of the commands for the low end controller (Figure 6-6,a), and

b) when the new information arrives at a frequency of sampling lower than the frequency of the commands for the low end controller (Figure 6-6,b).

The difference in operation of planning for these two cases acn be demonstrated as

follows. First let us consider a case of the functioning of the 1-st (the highest resolution) level when the frequencies fit to each other. This case is shown in Figure 6-7.

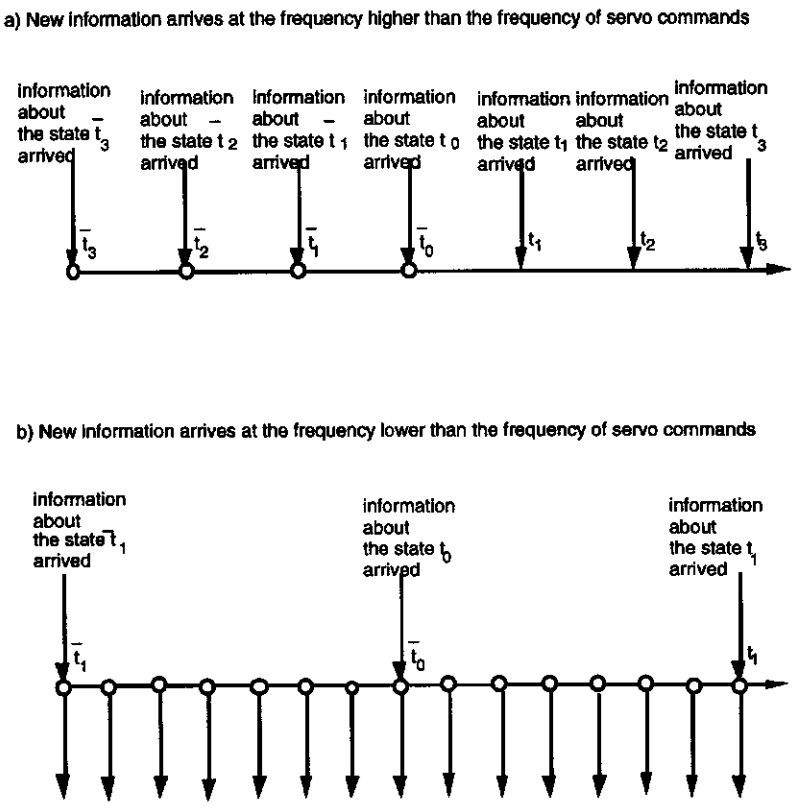


Figure 6-6. The algorithm of planning.

We can compare cases with too slow and too fast rates of new information arrival (see Figure 6-7). In the first case (Figure 6-8,a), the information about the state that has arrived at time $t_{(-1)}$ can be utilized only at the time of arrival of the next signal. This gives an opportunity to judge at least about the character of changes in the system (incremental, or decremental). In the second case (Figure 6-8,b), the multiplicity of input information signals can be used as a base for the learning process. The latter will allow for more accurate forecast computation.

In a number of practical cases related to metal-cutting machines controlled by a multiresolutional system, one can use from three to five consecutive signals to compute the forecast for one next interval of time. This makes the computed string of control commands more reliable, and the potential error smaller. However, to take advantage of this situation with the high rate of information arrival, the provisions should be made for using this information for the forecast computation.

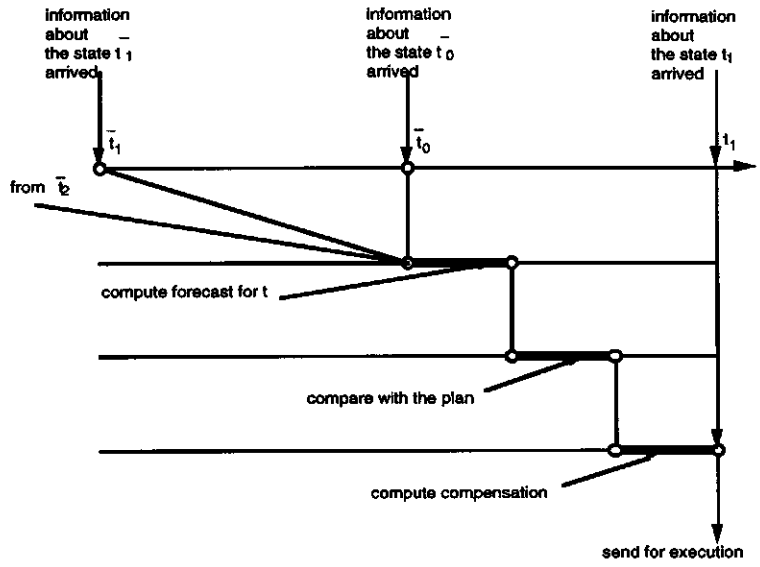


Figure 6-7 The case with frequencies that fit each other

Many of these facts are overlooked and/or neglected when new systems are being designed. Newly arrived information is considered to be a valid source for computing the next control command at all levels of resolution including the execution level (low end controller). This can be explained as a successful experience of using Kalman Filters in applications where there is no actual need in it. This type of a controller, by the very nature of its principle, builds up a base for the forecast and computes compensation based on using the forecast information. Figures 6-6 through 6-8 show that properties of information and the schedule of its use can be easily taken into account when the simplest and the most flexible PD and PID controllers are used.

a) Low frequency of new information arrival **b) High frequency of new information arrival**

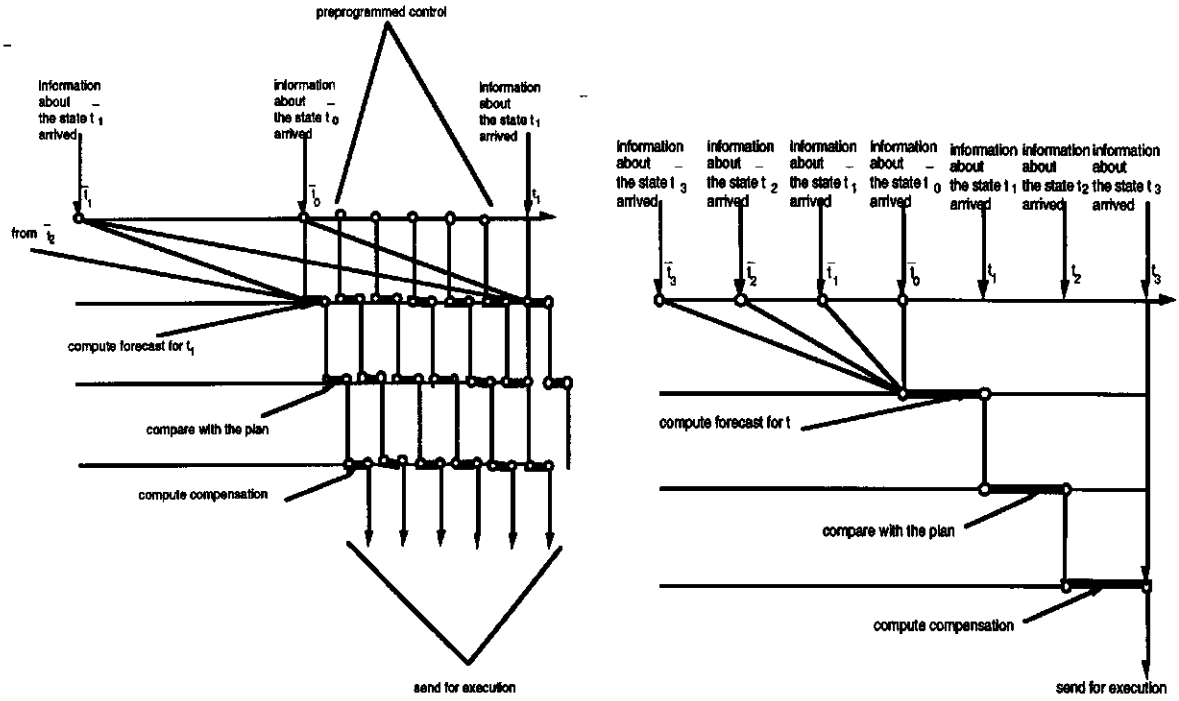


Figure 6-8. Comparison of execution at different rate of information arrival

6.5 EXECUTOR as a TASK GENERATOR

EXECUTOR is a submodule for accomodating PLANS for particular ACTUATORS. Central part of this process is the error COMPENSATION. As the real-time control is performed, the results of PLANNING can be corrected by using FEEDBACK COMPENSATION (FBC) which is found from the similar "inverse-like" operator as follows

$$<\text{COMPENSATION}> \leftarrow \text{----------} F(<\text{OUTPUT}^p> - <\text{OUTPUT}^r>)$$

where F-is a Feedback Operator which represents the accepted strategy of feedback,

$<\text{OUTPUT}^p>$ and $<\text{OUTPUT}^r>$ are planned and real (observed) output trajectories correspondingly.

Selection of the concrete strategy F of feedback control depends on the context and the requirements of the user. There are many laws of control that reflect different strategies. If the processes are fairly slow, proportional feedback can be good enough. It can be improved by

adding an operation of prediction, which helps in all other cases also. Often, a coarse operation of prediction can be done by a simple differentiation. Multiple experiences show that even for processes with substantial dynamics, taking into account a simple derivative of the process turns out to be sufficient, and this is how proportional-derivative strategy emerges (PD-control). To take into account a cumulative error, designers use an additional component of the feedback signal that is proportional to the integral of the error. Integration of all these strategies lead to the proportional-integral-derivative feedback. This is well known PID-controller. If the designer is concerned with the statistical evaluation of the error the strategy of control should include computation of the square of this error.

Finally, the equation of the real time operation of the i-th level will be

$$\text{TASK}\leftarrow\text{----------}C_{rt}(\langle\text{INPUT}\rangle, \langle\text{COMPENSATION}\rangle)$$

where C_{rt} - is the operator of real time control, using for example direct addition

$$C_{rt}(\langle\text{INPUT}\rangle, \langle\text{COMPENSATION}\rangle)=\langle\text{INPUT}\rangle + \langle\text{COMPENSATION}\rangle$$

Altogether, the ASSIGNMENT GENERATION can be considered a convolution

$$\text{ASSIGNMENT GENERATION}\leftarrow\text{---------}[S*P^{-1}]*F$$

that can be considered a KNOWLEDGE INVERSE OPERATOR (KIO) for the behavior generator.

This process has been discussed for a system as a whole. However, it looks the same for each level of resolution separately. Obviously, other subsystems are supposed to be attached to the fragment demonstrated in this picture. We focus upon this fragment only because BG-module is determining the whole functioning of the system, while other subsystems provide support of BG-module. The fundamentals of productivity, efficiency, and effectiveness are contained in this module. Its links to the rest of the system are not only in its connections at a level, but also in its connections to other levels.

7. Conclusions: Integrating BG in Intelligent System

- NIST-RCS is a nested multiresolutional system where each level of resolution can be considered and Elementary Functioning System (ELF) which includes World, Sensors, Perception (Sensory Processing), World Model, Behavior Generator, and Actuators.

- Behavior of Intelligent Systems is generated as a result of joint functioning of modules for Perception,World Modeling, Value Judgment and Behavior Generation which together perform Planning and Control of the system equivalent to its behavior.

- Behavior Generation as a subsystem of NIST-RCS is based upon functioning of its inner submodules responsible for Planning and for Control of the level of resolution.

- PLANNER consists of three sub-submodules for Job Assignment, Scheduling and Selection of the best PLAN). Planner plays a role of the Feedforward Controller: it computes the set of commands which are valid if the knowledge of the system is adequate. Planning starts with Job Assigning which outlines the alternative of spatial distribution of the job among the participating AGENTS. Scheduler searches for the minimum cost plan for all alternatives of job assignment. Selector chooses the best alternative of the schedule.

- EXECUTOR consists of the sub-submodules for inverse computation, error estimation, and error compensation. Executor plays a role of the Feedback Control System.

- In all cases, the process of developing the NIST-RCS Architecture should start with its Behavior Generation module including Planner and Executor since they affect performance

of the system the most.

- All other subsystems of the ELF are supportive for Behavior Generating subsystem. Development and functioning of other subsystems (including Sensors and Actuators) is determined by the way the Behavior Generating structure is designed and operate.

- Behavior can be generated only as a process in a loop (Elementary Loop of Functioning.) Analysis of systems starts with identification of existing loops. This is a non-trivial procedure since finding a loop can be done only by hypothesizing: constructing them tentatively and exploring whether the loop functioning does not contradict any experimental data. In turn, the hypotheses depend on our familiarity with the systems of signs that can be identified within the environment of interest.

- NIST-RCS System has as many loops as it has levels of resolution. The number of levels of resolution depends on providing for condition of minimum complexity of computations in addition to symbol grounding, i.e. finding a correspondence between the hypotheses and the existing experimental data.

- All loops should be treated separately. This means that their "languages" should be developed and utilized according to specifications written for this loop. Translation of a language of one level of resolution into a language for the adjacent level should be considered a part of the analysis of Behavior Generation processes. Each loop has to be checked for satisfactory conditions of inclusion (related to their Knowledge flows.)

- All loops have their own flow of Knowledge (Information.) The rules of consistency and the laws of conservation should be formulated and checked for each of the loops. All loops have to be checked for consistency by verifying conditions of nesting top-down and bottom-up in each module of the ELF.

- The core of the Behavior Generation subsystem is the concept of Recursive Nested Hierarchies. All NIST-RCS results and applications are determined by the concept and implicit formalisms of Recursive Nested Hierarchies. The concept of a hierarchy in RCS is different from the mathematical concept of a decision tree because of the nodes at a level are interrelated and form a network.

- The formalisms of Recursive Hierarchies are based on fundamental decision making procedure (determined by the triplet of intelligence with its components: generalization, focusing attention, combinatorial search) applied to the data (knowledge) structures typical for the area of application.

- The algorithms of Recursive Hierarchies are problem invariant. The results of NIST-RCS design are determined by the context information to which the formalisms of Recursive Hierarchies are applied.

- All Integrated Complex Systems allow for effective modeling by the formalisms of Recursive Hierarchies

- The property of nesting requires that the hierarchies satisfy additional rules of inclusion.

References

[1] L.D. Erman, F. Hayes-Roth, V.R.Lesser, and D.R. Reddy, D.R., "Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computer Survey,* vol. 23, June 1980, pp. 213-253.

[2] J.E. Laird, A. Newell, and P. Rosenbloom, "SOAR: An Architecture for General Intelligence", *Artificial Intelligence*, vol. 33, 1987, pp. 1-64.

[3] Honeywell Inc., "Intelligent Task Automation Interim Technical Report II-4", Dec 1987.

[4] J. Lowerie, et al. "Autonomous Land Vehicle", *Annual Report, ETL-0413*, Martin Marietta Denver Aerospace, July 1986.

[5] D. Smith and M. Broadwell, "Plan Coordination in Support of Expert Systems Integration", *Knowledge-Based Planning Workshop Proceedings*, Austin, TX, December, 1987.

[6] J.R. Greenwood, G. Stachnick, H.S. Kaye, "A Procedural Reasoning System for Army Maneuver Planning", *Knowledge-Based Planning Workshop Proceedings*, Austin, TX, December, 1987.

[7] A.J. Barbera, J.S. Albus, M.L. Fitzgerald, and L.S. Haynes, "RCS: The NBS Real-time Control System", *Proceedings Robots 8 Conference and Exposition*, Detroit, MI, June 1984

[8] R. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, vol. RA-2, 1, March, 1986

[9] G.N. Saridis, "Foundations of the Theory of Intelligent Controls", *IEEE Workshop on Intelligent Control,* 1985.

[10] A. Meystel, "Intelligent Control in Robotics", *Journal of Robotic Systems*, 1988.

[11] J.A. Simpson, R.J. Hocken, and J.S. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards", *Journal of Manufacturing Systems*, Vol. 1, No. 1, 1983.

[12] J.S. Albus, C. McLean, A.J. Barbera, and M.L. Fitzgerald, "An Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Environment", *4th IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology*, Gaithersburg, MD, October 1982.

[13] J.S. Albus, "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles", National Institute of Standards and Technology, Technical Report 1251, Gaithersburg, MD, September 1988.

[14] J.S. Albus, H.G. McCain, and R. Lumia, "NASA/NBS Standard Reference Model for

Telerobot Control System Architecture (NASREM)" National Institute of Standards and Technology, Technical Report 1235, Gaithersburg, MD 1989.

[15] A. Meystel, M. Montgomery, D. Gaw, "Navigation algorithm for a nested hierarchical system of robot path planning among polyhedral obstacles," Proc. IEEE Int'l Conf. on Robotics and Automation, Raleigh, NC,1987, pp. 1616-1622

[16] C. Isik, A. Meystel, "Pilot level of a hierarchical controller for an unmanned mobile robot," *IEEE J. of Robotics & Automation*, Vol. 4, No. 3, 1988 pp. 244-255

[17] A. Meystel, Autonomous Mobile Robots : Vehicles with Cognitive Control, World Scientific Publ., Singapore, 580 p., 1991

[18] D. Apelian, A. Meystel, "Knowledge Based Control of Material Processing: Challenges and Opportunities for the Third Millenium", Eds. H. Y. Sohn, E. S. Geskin, Metallurgical Processes for the Year 2000, TMS, Warrendale, PA, 1989

[19] C. Corson, A. Meystel, F. Otsu, S. Uzzaman, "Semiotic Multiresolutional Analysis of a Power Plant", ibid. in Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems, Proc. of the 1995 ISIC Workshop, Monterey, CA, 1995, pp. 401-405

[20] B. Hayes-Roth, "A Blackboard Architecture for Control", *Artificial Intelligence*, pp. 252-321, 1985.

[21] J.S. Albus, *Brains, Behavior, and Robotics*, BYTE/McGraw-Hill, Peterborough, N.H.,1981

[22] G. A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", *The Psychological Review*, 63, pp.71-97, 1956.

[23] A. Meystel, "Theoretical Foundations of Planning and Navigation for Autonomous Robots", *International Journal of Intelligent Systems*, 2, 73-128, 1987

[24] M. Minsky, "A Framework for Representing Knowledge", 211-277 in P. Winston (Ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975

[25] J. Albus, A. Meystel, "A Reference Model Architecture for Design and Implementation of Semiotic Control in Large and Complex Systems", in Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems, (eds. J. Albus, A. Meystel, D. Pospelov, T. Reader), Proc. of 1995 ISIC Workshop, Monterey, CA 1995, pp. 33-45

[26] E.D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier, New York, 1977

[27] R.C. Schank and Abelson, R.P., *Scripts Plans Goals and Understanding*, Hillsdale, NJ, Lawrence Erlbaum Associates, 1977.

[28] D.M. Lyons and Arbib, M.A., "Formal Model of Distributed Computation Sensory Based Robot Control", *IEEE Journal of Robotics and Automation in Review*, 1988

[29] D. W. Payton, "Internalized Plans: A Representation for Action Resources", *Robotics and Autonomous Systems*, 6, 89-103, 1990

[30] A. Sathi and M. Fox, "Constraint-Directed Negotiation of Resource Reallocations", CMU-RI-TR-89-12, Carnegie Mellon Robotics Institute Technical Report, March, 1989

[31] V.B. Brooks, *The Neural Basis of Motor Control.* Oxford University Press 1986

[32] J. Piaget, *The Origins of Intelligence in Children*, International Universities Press, New York, 1952

[33] J.S. Albus, "A Theory of Cerebellar Function", Mathematical Biosciences, Vol. 10, pp. 25-61, 1971.

[34] P.D. MacLean, *A Triune Concept of the Brain and Behavior* , University of Toronto Press, Toronto, 1973

[35] A. Schopenhauer, "The World As Will and Idea", *1883* , In *The Philosophy of Schopenhauer*, Edited by Irwin Edman, Random House, New York, NY, 1928

[36] J.J. Gibson, *The Ecological Approach to Visual Perception*, Cornell University Press, Ithaca, N.Y. 1966

[37] D.H. Hubel and T.N. Wiesel, "Ferrier Lecture: Functional architecture of macaque monkey visual cortex", *Proc. Roy. Soc. Lond. B. 198*, 1-59 (1977)

[38] H. Samet, "The Quadtree and Related Hierarchical Data Structures", *Computer Surveys*, 16-2, 1984

[39] P. Kinerva, *Sparse Distributed Memory*, MIT Press, Cambridge 1988.

[40] J.S. Albus, "A New Approach to Manipulator Control : The Cerebellar Model Articulation Controller (CMAC)", *Transactions ASME,* September 1975.

[41] J.S. Albus, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)", *Transactions ASME*, September 1975.

[42] M. Bradey, "Computational approaches to image understanding", *ACM Computing Surveys,* 14, March, 1982

[43] T. Binford, "Inferring surfaces from images", *Artif. Intell,* 17, 205-244, 1981

[44] D. Marr and H.K. Nishihara. "Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. Roy. Soc. Lond. B* 200, 269-294, 1978

[45] R.F. Riesenfeld, "Applications of B-spline Approximation to Geometric Problems of Computer Aided Design", Ph.D Thesis, Syracuse University (1973). Available at University of Utah, UTEC -CSc-73-126.

[46] J.J. Koenderink, "The Structure of Images", *Biological Cybernetics*, 50, 1984.

[47] J.C. Pearson, J. Gelfand, W.Sullivan, R. Peterson, and C. Spence, " A neural network approach to sensory fusion", *Proceedings SPIE Sensor Fusion Conference*, Orlando, 1988

[48] D.L. Sparks and M. Jay, "The Role of the Primate Superior Colliculus in Sensorimotor Integration", In *Vision, Brain, and Cooperative Computation* (Arbib and Hanson, Eds.) MIT Press, Cambridge, 1987

[49] R.A. Andersen and D. Zipser, "The role of the posterior parietal cortex in coordinate

transformations for visual-motor integration", *Can. J. Physiol. Pharmacol.,* 66, 488-501, 1988

[50] D. Marr, *Vision,* W.H. Freeman, San Francisco, 1982

[51] J.S. Albus and Hong, T.H., "Motion, Depth, and Image Flow", *Proceedings of the IEEE Robotics and Automation,* Cincinnati, OH, 1990 (in process)

[52] D. Raviv and J.S. Albus, "The computation of range from image flow given knowledge of eye translation and rotation", *IEEE Transactions on Robotics* (to be published)

[53] E. Kent and J.S. Albus, "Servoed World Models as Interfaces between Robot Control Systems and Sensory Data", *Robotica,* Vol. 2, pags. 17-25, 1984.

[54] E.D. Dickmanns and T.H. Christians, "Relative 3D-State Estimation for Autonomous Visual Guidance of Road Vehicles", Intelligent Autonomous System 2(IAS-2), Amsterdam, 11-14 December, 1989

[55] R. Bajcsy, "Passive perception vs. active perception" *Proc, IEEE Workshop on Computer Vision,* Ann Arbor, 1986

[56] K. Chaconas and M. Nashman, "Visual Perception Processing in a Hierarchical Control System: Level 1", National Institute of Standards and Technology Technical Note 1260, June 1989.

[57] Y.L. Grand, *Form and Space Vision,* Table 21, Indiana University Press, Bloomington, 1967

[58] A.L. Yarbus, *Eye Movements and Vision,* Plenum Press, 1967

[59] D.C. Van Essen, "Functional organization of primate visual cortex", A.Peters and E.G. Jones (eds.) *Cerebral Cortex* 3: 259-329, Plenum, New York, 1985

[60] J.H.R. Maunsell and W.T. Newsome, "Visual processing in monkey extrastriate cortex", Ann. Rev. Neurosci. 10: 363-401, 1987

[61] S. Grossberg, *Studies of Mind and Brain,* Reidel Publishing Co., Holland, 1982]

[62] G. E. Pugh, *The Biological Origin of Human Values,* Basic Books, New York, 1977

[63] A.C. Guyton, *Organ Physiology, Structure and Function of the Nervous System,* 2nd ed., Philadelphia: W.B. Saunders, 1976

[64] W. B. Scoville and B. Milner, "Loss of recent memory after bilateral hippocampal lesions", *Journal of Neurophysiology, Neurosurgery and Psychiatry,* 20, (11), p. 11-29, 1957

[65] J.S. Albus, "Mechanisms of Planning and Problem Solving in the Brain", *Math. Biosciences,* 45, p.247-293, 1979

[66] S. Grossberg (Ed.), *Neural Networks and Natural Intelligence,* Bradford Books, MIT Press, 1988

[67] J.S. Albus, "The Cerebellum: A Substrate for List-Processing in the Brain", In *Cybernetics, Artificial Intelligence and Ecology,* editors Robinson, H.W., and Knight,

D.E., Spartan Books, 1972.

[68] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings National Academy of Sciences,* 79: 2554-2558, 1982

[69] B.Widrow and Winter, R. "Neural Nets for Adpative Filtering and Adaptive Pattern Recognition", *Computer*, 21 #3, 1988.

[70] M. Minsky and Papert, S., *An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969

[71] M. Ito, *The Cerebellum and Neuronal Control*, Chapter 10, Raven Press, N.Y., 1984

[72] J. Albus, "A Canonical Architecture for Intelligent Machine System", NIST, Gaithersburg,

[73] J. Albus, "Outline for a Theory of Intelligence", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, May/June 1991, pp. 473-509

[74] J. Albus, A. Meystel, A. Lacaze, "Algorithm of Nested Clustering for Unsupervised Learning", Proc. of the 10th Int'l Symposium on Intelligent Control, Monterey, CA, 1995

[75] A. Meystel, "Multiscale Systems and Controllers", Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, Tuscon, AZ, pp. 13-26

[76] L. A. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press, Cambridge, UK, 1987, p. 203

[77] Y. Maximov, A. Meystel, "Optimum Design of Multiresolutional Hierarchical Control Systems", Proc. of the IEEE Int'l Symposium on Intelligent Control, 11-13 August, 1992, Glasgow, Scotland, UK, 1992

[78] A. Meystel, "Nested Hierarchical Control", Eds. P. Antsaklis, K. Passino, *An Introduction to Intelligent and Autonomous Control*, Kluwer, 1993

[79] C. Knoblock, Generating Abstraction Hierarchies, Kluwer, 1993

[80] T. Allen, T. Starr, Hierarchy, The University of Chicago Press, 1982

[81] S. Salthe, Evolving Hierarchical Systems, Columbia University Press, 1985

[82] S. McCormick (Ed.), Multigrid Methods, SIAM, 1987

[83] R. Graham, D. Knuth, O. Patashnik, Concrete mathematics, Addison-Wesley, 1989

[84] J. Sowa, Conceptual Structures, Addison-Wesley, 1984

[85] J. Mendel, Lessons in Digital Estimation Theory, Prentice Hall, 1987

[86] C. Therrien, Discrete Random Signals and Statistical Signal Processing, Prentice Hall, 1992

[87] G. Saridis, C. S. G. Lee, "Approximation of Optimal Control for Trainable Manipulators", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.8, No. 3, 1979, pp. 152-159

[88] A. Meystel, Semiotic Modeling and Situation Analysis: An Introduction, Publ. AdRem, Inc., Bala Cynwyd, Pa, 1995

[89] A. Meystel, Intelligent Systems: A Semiotic Perspective, *Int'l Journal of Intelligent Control*

and Systems, Vol. 1, No. 1, 1996, pp. 31-58

[90] *Architecture of Semiotic Modeling and Situation Analysis*, eds. J. Albus, A. Meystel, D. Pospelov, T. Reader, Proc. of the IEEE Workshop at the 10'th ISIC, Monterey, CA 1995

[91] J. Albus and A. Meystel, "A Reference Model Architecture for Design and Implementation of Semiotic Control in Large and Complex Systems", published in *Architecture of Semiotic Modeling and Situation Analysis*, eds. J. Albus, A. Meystel, D. Pospelov, T. Reader, Proc. of the IEEE Workshop at the 10'th ISIC, Monterey, CA 1995, pp. 33-45

[92] M. Herman, J. Albus, "Overview of MAVS: Multiple Autonomous Undersea Vehicles", *Unmanned Systems*, Winter 1988/89, pp. 36-52

[93] M. Herman, J. Albus, "Real-Time Hierarchical Planning for Multiple Mobile Robots", Knowledge Based Planning Workshop, Proceedings, December 1987, pp. 22-1—22-10

[94] R. Wilensky, Planning and Understanding, Addison-Wesley, 1983

[95] M. Arbib, The Metaphorical Brain, New York, Wiley, 1972

[96] E. Dynkin, A. Yushkevich, Controlled Markov Processes, Springer, 1975

[97] E. Sacerdoti, A structure for Plans and Behavior, Elsevier, 1977

[98] D. Wilkins, Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, 1988, pp. 46-50

[99] J.-C. Latombe, Robot Motion Planning, Kluwer, 1991

[100] A. Meystel, S. Uzzaman, "Planning Via Search In The Input-Output Space", Proc. of the 8-th IEEE Int'l Symposium on Intelligent Control, Chicago, IL, 1993

[101] D. Wilkins, Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, 1988, pp. 25-39

[102] M. Georgeff, et al, "Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot", Technical Note 380, SRI, Menlo Park, CA, 1986

[103] T. Dean, M. Wegman, Planning and Control, Morgan Kaufmann, 1991, p. 206

[104] L. Kaebling, "An Architecture for Intelligent Reactive Systems", eds. M. Georgeff, A. Lansky, Reasoning about Actions and Plans, Proc. of the1986 Workshop, Morgan Kaufmann, 1986, pp. 395-410

[105] J. Albus and A. Meystel, "A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems", *The Int'l Journal for Intelligent Control and Systems* , Vol.1, No.1, 1996, pp. 15-30

[106] L. Kraft, D. Campagna, "A Comparison Between CMAC NN Control and Two Traditional Adaptive Control Systems", in Eds. E. Sanches-Sinencio, C. Lau, Artificial Neural Networks, IEEE Press, 1992, pp. 483-490

[107] J. Albus, "A Reference Model Architecture for Intelligent Systems Design", Ed. by P. Antsaklis, K. Passino, An Introduction to Intelligent and Autonomous Control, Kluwer

Academic, 1993

[108] H. M. Huang, R. Hira, R. Quintero, "Submarine Maneuvering System Demonstration Based on the NIST Real-Time Control System Reference Model", Proc. of the 8th IEEE Int'l Symposium on Intelligent Control, Chicago, IL 1993

[109] F. Proctor, J. Michaloski, "Enhanced Machine Controller Architecture Overview," NISTIR 5331, NIST, Gaithersburg, MD, December 1993

[110] M. Nashman, W. Rippey, T. Hong, M. Herman, "An Integrated Vision-Touch Probe System for Dimensional Inspection Tasks," NISTIR 5678, June 1995

[111] K. Stouffer, J. Michaloski, R. Russell, F. Proctor, "ADACS - An Automated System for Part Finishing," NISTIR 5171, NIST, Gaithersburg, MD, April 1993

[112] R. Quintero, A. J. Barbera, "A Software Template Approach to Building Complex Large-Scale Intelligent Control Systems," Proc. of the 8th IEEE Int'l Symposium on Intelligent Control, Chicago, IL 1993

[113] Eds. Mary Beth Ruskai, et al., Wavelets and Their Applications, Jones and Bartlett Publishers, Boston, 1992

[114] B. Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman and Co., 1977

[115] R. Bostelman, J. Albus, N. Dagalakis, A. Jacoff, "Aplications of the NIST Robocrane", Proc. of the 5th International Symposium on Robotics and Manufacturing, Maui, HI, 1994

[116] L. Kelmar, Manipulator Servo Level World Modeling, NIST Technical Note 1258, NIST, Gaithersburg, MD 1989

Appendix I.

Definitions

Action- is an effort generated by the Actuator which produces changes in the World.

Actuator- is a subsystem which receives commands from the Behavior Generation module and produces the effort (Action) to create changes in the World.

Actuator, Real - is a physical system which responds to the string of commands by transforming the input energy into the output motion producing changes in the world.

Actuator, Virtual - is a set of all subsystems of the NIST-RCS structure which are situated below a BG module which submits to the Actuator the string of commands. At all levels of the NIST-RCS hierarchy, the virtual actuator is decomposed into a new set of the subsystem which continue the process of BEHAVIOR GENERATION. Only at the very bottom, the string of commands arrives at the input of a real actuator.

Agents- is a code word for the subsystems recognized at the adjacent higher level of resolution. Agents cooperate with each other by the virtue of communication.The results of their cooperation are to be evaluated externally. An agent can have its own decision making system. All levels of the NIST-RCS hierarchy are agents. If the agent cannot only decide for itself, but has a freedom in motion execution it is called an **Autonomous Agent**. The concept of Agent is very vague. A subsystem of a system can be called an agent; a module of the RCS controller can be called an agent.

Behavior- is the ordered set of consecutive/concurrent changes among the states (in a simple case, "the string of changes between the consecutive states") registered at the output of the system (subsystem). This is a unified property ("regular" behavior) if a "law" of the string formation is found. If the law is not found, we call it "stochastic" or "random" behavior. Therefore, any output of the system observed during some interval of time can be

considered "behavior" of this system. This means also that behavior of the system can be described as a time-tagged trajectory (motion) in the state space.

Command- is the assignment encoded in such way that the Task could be invoked within the module which receives the command. This is a code word for the assignment admissible within the particular level of the system. It is the output of the upper level presented in the form of the encoded task. The command by itself is insufficient to trigger the operation. The State and the Spatio-Temporal Model should be submitted by the World Model, as well as the information from the other Behavior Generating modules at this particular level of resolution. Command contains the name, goal, object, constraints to be satisfied and cost-function to be minimized.

Constraints- are the boundaries in the state space to which the process of the functioning of a system should be confined.

Control- the term "control" is used at higher resolution levels to describe the same phenomenon which is called, "plan," is at the lower level of resolution. The sequence of plans with gradually increased level of resolution ends up with control at the bottom of the NIST-RCS hierarchy. One can even talk about "planning/control continuum" top-down. (See *Feedforward Control*) and *Feedback Compensation*).

Cost-function- is the rate of losses (gains) of the resources during the process of goal achievement. It is typical to try to minimize (maximize) the cost-function.

Cost-functional- is a cumulative criterion which summarizes the total amount of losses (gains) during the process of goal achievement. It is typical to try to minimize(or maximize) the cost-functional.

Criterion of optimality- is a function which should be maximized (minimized) during the operation of a system.

Event- any change of the world for which the beginning and the end are defined is called an event (a discrete event). If the change cannot be detected at a particular level of resolution (because it is indistinguishable, i.e. is below the threshold of sensitivity of the sensors), then the event does not exist at this level as a phenomenon.

Feedback compensation- (FBC) a correction to the plan computed by comparing the outcome of the process with the planned trajectory without correcting the world.

Feedforward Control- (FFC) a string (time sequence) of input commands and/or input signal determined as a function of time (or another variable which can be assumed independent within the system under consideration). This is supposed to provide the desirable output motion trajectory. FFC can be computed in many cases when the conditions of functioning can be expected (and/or predicted). Real Motion will always differ from the expected when FFC is applied. However, the feedback controller will play role only of feedback compensation controller (FCC). The intensity of on-line control operations in most of the practical cases will be drastically reduced.

Goal- is the state to be achieved; it is represented as a data structure. Goal is determined at a level. Different approaches are acceptable in intelligent systems. One of them presumes that the goal for the next level of the hierarchy is selected by the upper level and submitted for performance. Another approach: goal is determined at the level under consideration after analysis of the long-term schedule submitted from the level above.

Indistinguishability zone- is the volume of the state space at a higher resolution level which is considered a point at the lower level of resolution.

Job- an elementary unit of behavior at a level of NIST-RCS

Level of a Hierarchy, (level of resolution, granularity, generalization, abstraction)- is a representation of the system with a particular level of detail. Level of a Hierarchy can be also called level of resolution, level of granularity, level of generalization, level of abstraction. Level of resolution, and level of granularity have the same meaning because both resolution and granularity refer to the same idea of indistinguishability zone. Level of generalization presumes that different resolution (granularity) of levels are obtained as a result of the properly performed generalization. The expression "level of abstraction" is often used instead of "level of generalization,"

although their meaning is not equivalent[73] . Abstraction means focusing upon some particular feature and/or property of an object. Generalization presumes unifying a set of features and/or property, object into one property, object (generalized property, object). There are many methods of generalization including generalization via approximation, via averaging, via integration, via aggregation and labeling based on recognition and detection. The goal of both generalization and abstraction is to increase the efficiency of knowledge manipulation.

Nesting- a property of *being contained in*. In this book we apply this term as it relates to knowledge and information. This means that nesting always realizes via interpretation. Nesting puts some conditions upon information processing in hierarchies.

Plan- is the set of schedules for the group of agents which are supposed to perform these schedules as a cooperative effort and accomplish the required job (achieve the goal) as a result of this effort. To find this set of schedules different combinations of agents should be tested and different schedules should be explored. Plan is also defined as the course of events determined within BG-module which is supposed to be reproduced in the World to achieve the Goal in the desirable fashion;

or— it is a description of the set of behaviors which lead to the Goal in the desirable fashion. This description is represented as a set of "schedules

or— it is a state space trajectory that describes the behavior of system leading to the goal and providing satisfaction of constraints and conditions on some cost-function, or cost-functional (these conditions might include: having the value of this cost-function/cost-functional within some interval, maximizing, or minimizing it).

Thus, *plan* controls the system. It consists of two major components: the final state which should be achieved in the end of the planning interval, and the string of the intermediate states which are often supplemented by their time-schedule.

Plan consists of task space/time decompositions, such that the subtasks are

---

[73] The examples of abstraction via decision making can be found in C. Knoblock, Generating Abstraction Hierarchies, Kluwer, 1993. The examples of generalization via averaging can be found in W. Gray, et al, Mathematical Tools for Changing Spatial Scales, CRC Press, 1993; J. Sanders, F. Verhulst, Averaging Methods in Nonlinear Dynamical Systems, Springer-Verlag, 1985

distributed in space and time. It may be represented as a PERT chart, Gant diagram, a state transition graph, a set of schedules complemented by the account of resources required (such as bill of materials, tools and manpower requirements, delivery schedules, and cost estimates.) Each Plan is characterized by its *goal, time horizon , set of agents (performers),* and *its envelope.*

Plan, Optimal - is the plan which leads to the goal achievement while minimizing (or maximizing) a particular cost-function, or a cost-functional. Optimal plan can be found (synthesized) only as a result of the comparison among all alternatives of feasible (admissible) plans.

Plan, Satisficing[74] - is one of the admissible plans which is within a narrowed set of constraints. It is one of the state space trajectories which is constructed within the desirable boundaries specified by a customer who does not want to determine the cost-function. In other words, this is a sufficient, satisfactory, but not necessarily "the best" plan.

Plan, Spatial- is the state space trajectory (in the enhanced state space which includes inputs, outputs, and states of the system.) The state space trajectory should be represented at the output of the planning submodule as the result of selection of agents and jobs assigned to them, their responsibilities and criteria of their performance.

Plan, Temporal- see *Schedule.*

Plans, Admissible - are all meaningful plans that can be built within the specified constraints.

Planning- is the design of the course of events determined within BG-module; design of the desirable state space trajectory; design of the feedforward function, and thus, the future for the system. Planning is performed in an assumption that we know the agents of the adjacent higher level of resolution which will cooperate in the process of the further delineation of the plan. This assumption corresponds to one particular alternative of the solution. Another alternative has another assumption about performing agents and leads to

---

[74] This term was introduced by H. Simon.

another plan. The design of the desirable motion of the system entails that many supportive components of operation also should be planned: the algorithms of feedback compensation, inputs to the energy converters, the scope of sensing (focus of attention), and others.

Planning envelope- is a subset of the state space with a corresponding world model which is submitted to BG at the higher level of resolution for refinement. Upon completion of the planning process at a level, a part of this plan should be refined by searching for a more precise solution in the limited envelope around the planned trajectory. A subset of the Plan (for a limited time $0<t\leq Dt_h$) is submitted to BG unit of the higher level of resolution for refinement.

Planning horizon- is the time interval within which a Plan is meaningful. The degree of belief for each future state of the plan falls off as time t grows large because the stochastic component of the operation affects the verifiability of the results. For some particular Dt_h in the future the degree of belief is lower than the degree required for the decision making process. This Dt_h is called "planning horizon".

Planning strategy- orientation toward receiving either the *optimal* or the *satisficing* Plan

Replanning- is the process of planning which is performed if the top-down and bottom-up processes of plan propagation did not converge. The need in replanning can emerge a) if the initially selected version of plan distribution failed, b) if the prescribed conditions of compensation fail to keep the process within the prescribed boundaries, c) if the World Model has changed, d) Goal has changed.

Resources- the following resources are usually taken into account: time, energy, materials, remaining life-span of the system, the degree of fault-tolerance, and money.

Resolution- is the property of the level of hierarchy which limits the distinguishability of details.

Schedule- is another term for the "temporal plan;" it is the description of the

development of the process in time. It obtained by computing the state space trajectory within the time domain. The schedule should focus upon the start and the end events and provide for coordination, reduced queues, and elimination of the "bottlenecks". Schedule can be also defined as a job-time event-gram.

Scheduling- is outlining the temporal development of the process.

Simultaneity, statements of- can be understood in a trivial way only for the events performed and observed at a particular level. Trivially, we mark a point at the time axis and all states corresponding to this point consider "simultaneous." Since all resolution levels have a different time scale, this trivial way cannot be applied. Events and processes belonging to different levels can be considered *simultaneous* if the time units of consideration overlap (fully, or partially).

State- is a data structure representing the "snapshot" of the World; a description of Reality (of interest) at a particular moment of time, e. g. the set of states together with the sets of inputs, outputs and cost-functions (including variables and if necessary, their derivatives too; a set of coordinates with their particular value (each value is given as an interval[75] ; the smallest possible interval is equal to indistinguishability zone at a particular level of resolution).

Subsystems- structural and/or functional parts of the system; the result of system decomposition; deeply related to the task decomposition; reflects both the statistics of all expected task decompositions and the way the system was manufactured. This means that decomposition can emerge as a result of the system to be naturally decomposable in a set of particular subsystems (determined at the stage of design). However, often the design decision is determined by the rational decomposition which is determined so that maximize the efficiency of the system. The system and all its subsystems (including the subsystems of the subsystems, etc.) determines the vocabularies which are used later for task decomposition purposes.

---

[75] At a particular level of resolution any numerical assignment can be considered either as an expectation with its $\pm 3\sigma$ dispersion, or as an interval between [min.value-max.value].

System- Any machine or aggregate including a robot, an autonomous vehicle, a robotic cell, a manufacturing floor, a factory, or others which are supposed to be controlled by NIST-RCS, are referred to as the *System*. The complete system consists of the plant and the NIST-RCS. Very often though, instead of using the term plant we refer to it as to a system.

Task- is the data structure representing the assignment i.e. specifying the goal to be achieved; the Command for the higher resolution level issued by the lower resolution level specifies the Goal for the higher resolution level. It is a description of the goal state and the alternatives of its achievement, together with the key parameters that should be maintained; all this is specified in a form of an abstract data structure (a frame); it can be also represented by the Action which is supposed to terminate at this state (in this case, the condition of termination should be specified).

Task Aggregation- is the bottom-up process opposite to the process of Task Decomposition. After the process of planning is performed at the next lower level (lower in abstraction and higher in resolution) and the BG-units of the higher resolution level delineated their plans, their results should be aggregated to verify whether the solution is the best possible solution. This process of aggregation should be performed before the execution starts at the level of resolution under consideration. Aggregation is a particular case of "generalization" (the process which is opposite to "refinement") model.

Task Decomposition- is the process of consecutive refinement of the goal (with its consecutive division into subgoals) which produces new tasks (assignments) for all levels of the hierarchy top-down. Each subtask is a result of considering the initial task at higher resolution. Thus, recognizing its spatial and/or temporal structure. Instead of the term, "decomposition," one can use the term, "refinement". Finding the subtasks is done as a result of refining (decomposing) the initial task.

Appendix II

Assumptions related to Behavior Generation.

The following assumptions are accepted in this book:

Assumption 1. The Decision Making Imperative: The process of Decision Making is a selection of one ("the best") alternative out of the set of available alternatives {A}. This set {A} should be constructed by mapping the subset of problem description into the set of components of solution description.

This mapping results in synthesizing the Plan. These two sets of problem description and of solution components description are intrinsic part of the World Model after it is exposed to the Goal. In fact, as soon as the Goal emerges, the World Model produces these two sets, together with the mapping among them. This mapping is utilized to construct the alternatives. Different techniques of constructing the alternatives will be presented in a separate report.

Assumption 2. The Hierarchical Imperative: Complex Systems are represented and constructed as Hierarchies because this is the available way of providing *efficient functioning under constraint of limited resources.*

It would be more correct to use the term, "multiresolutional system" instead of "hierarchy". Classical hierarchies are trees; the hierarchies we are dealing with in NIST-RCS systems are not trees. They have horizontal connections at each resolution level; they have cross-connections. The leaves of a node at a particular level can be also connected to another nodes of the same level. The representation of different levels have different values

of resolution.

We use the terms, "multiresolutional system," "multigranular system," "multiscale system," and "hierarchy" intermittently. The situation begs for a special term—we do not have it.

Applying the Decision Making Imperative entails using the Hierarchical Imperative—other concepts lead to less efficient results.

Assumption 3. NIST-RCS architecture is to be developed for the system which already exists as well as for those systems development of which is just contemplated.

We will analyze the structure and processes of Behavior Generation in a system which has been manufactured and exists. In other words, the NIST-RCS system under consideration has been already fully designed (prior to discussing the Behavior Generation). We already know how many levels of resolution we deal with.

In the future, we will be interested in considering the system at all levels including its design, not only the design of the NIST-RCS for it. This will make the discourse more dynamic and difficult for analysis. This is exactly why we decided to avoid it at this stage. However, let us first agree upon some framework for Behavior Generation, confirm this framework by application examples, then we will be ready to deal with a more fluid and complex paradigm.

More specifically this means that:

Assumption 3A. The Vocabularies of the levels of the existing System to be equipped with NIST-RCS are known.

This means that we will not address the issue of learning. (This is where get the vocabularies of the System are supposed to be taken from). The Models provided by the Knowledge base are sufficient for planning, and the Vocabularies from which the choices

are combined are known. We encounter the uncertainties of the World and should plan so that they will not fail. We are not going to discuss the process of learning from these uncertainties. This will be a subject of another document.

Assumption 4. The behavior generated at the i-th level can be analyzed meaningfully only as a part of the behavior generated at the (i+1)-th level (together with other agents of the i-th level) and only if the behavior of all agents of the (i-1)-th level is known.

This means that no activity of a separate BG-module can be considered alone without its neighbors from above and from below.(The level numbers are counted bottom-up). For the two highest resolution levels of each hierarchy, the Assumption 4 means that the behavior generated at the level of the level 1 execution controller can be analyzed meaningfully only if the behavior of all actuators (i.e. servo-motors) is known.

Appendix III

Frequent Misconceptions about Behavior Generation.

The following misconceptions are common among the designers of the architectures of computer control systems for intelligent machines and integrated manufacturing.

Misconception 1.

The NIST-RCS architecture describes the computer related activities while human activities are beyond the architecture.

This leads to multiple mistakes in design and control. In fact, the architecture should be the organizing framework which allows to properly blend both the part related to computer and the activities of human operators. As a result, a nicely organized computer structure often process flows inefficient information which was delivered by a human. Even worse, the human operators submit what they consider to be important but it contradicts the very purpose of the NIST-RCS architecture.

Misconception 2.

The NIST-RCS architecture is about communication among the subsystem, not about what and how the subsystems process the information.

As a result, the subsystems of Planning, Execution, and Job Assignment are considered as a set of black boxes which properly communicate with each other no matter what is going on inside boxes. Most of the NIST-RCS architecture serves to the subsystem of Behavior Generation. NIST-RCS modular design should be oriented toward satisfying BG needs. We should determine these needs.The relations among the modules are an important part of the architecture. However, what these modules are doing is as important,

if not "more important." We can see that nested modules generate an important architectural demands. The system of communication should be designed since the adherence to formal interfaces specified for NIST-RCS can be used as a criterion of operability and dependability of the system. But, the information to be communicated depends on the further refinement of the communicating modules.

Misconception 3.

The designers determine the vocabularies of subsystems of the NIST-RCS intelligent controller either from experience, and/or upon their volition.

Indeed, the domain experience provides many concrete cases of behavior description in an unstructured way (for the system to be controlled.) It is a common practice to use these cases to organize the information and extract the vocabularies of the controller. This is a good practice, but not the only way. In the meantime, there exists a number of scientific techniques of vocabularies decomposition so that the desired cost-function or cost-functional could be minimized. This way can lead to solutions which have not been represented in the existing case studies.

Misconception 4.

The subsystems of STRATEGIC PLANNING and DESIGN in the structure of integrated manufacturing are usually omitted from consideration.

In the meantime, this stage is a legitimate part of the architecture belonging to a number of levels of resolution. One should not forget that each PLANNING process is a design of the future process. DESIGN is just a PLANNER at a sufficiently high level of resolution.

Misconception 5.

Levels of the architecture (levels of resolution, levels of granularity, levels of abstraction) are considered separate entities.

At the first glance, we want to consider levels separately. However, this leads to serious mistakes because each lower level of the system (higher resolution) is an inseparable part of its neighbor from above and is *nested* within this level. This is achieved via generalization of the behavior of the higher resolution level represented within the lower resolution level. Therefore, one cannot analyze a level without simultaneously considering the level below—as its component (and a base of generalization), and the level above—to which it belongs as a component. If the higher resolution level is not reliably represented to the lower resolution level, the decisions made at the level above can turn out to be meaningless for the level below, and thus, for the System in general.

In other words, the functioning of all top-down modules is actually a concurrent process in which each higher resolution level ("core-module") nested inside its lower resolution level ("shell-module") delivers to the "shell-module" the results of its higher resolution computations in generalized form. For example, the module of behavior generation at a particular level uses computations performed by the module of higher resolution after the results are generalized and translated into a proper vocabulary. Or the shell-module of World Model incorporates knowledge of the core-module transformed (generalized) to its resolution.

Misconception 6.

The process of NIST-RCS functioning is considered to be a top-down process with gradual refinement (decomposition).

This view is incomplete. The process of NIST-RCS functioning is a twofold process. It has a wave of activities spreading top-down (from the shell-module to the core module Then, from this core module it is considered now as a shell-module to its core-

module.) But, it also has a wave of the bottom-up activities (from the core module to its shell module). These two waves should lead to a converging result for each pair of adjacent levels of resolution. After this, the operation can be considered completed. If one provides a consistent (satisfying all constraints, minimizing the cost-functional, and converging) top/down and bottom/up processes for each pair of adjacent levels, the convergence will be provided for the whole hierarchy.

Misconception 7.

It is a common opinion that the PLANNER does planning while the JOB ASSIGNER does the distribution of jobs. These operations are performed sequentially rather than concurrently and cooperatively.

This is wrong. The PLANNER cannot do its planning without simultaneously considering how these activities will possibly be distributed among the participants (subsystems.) What JOB ASSIGNER does, preparation of alternatives of job distribution for further exploration, distribution of the COMMANDS among the computational models of the "participants" (the top-down part) and aggregation of the results of computation for the verification of the job distribution (the bottom-up part).

Misconception 8.

It is a common opinion that there are two different activities: planning and scheduling.

It is not so. Scheduling is a part of planning. Planning has two components: spatial planning (spatial organization of activity) and temporal planning (temporal organization of activity). These are strongly interrelated. One of them cannot be done without another. Thus, spatial planning and scheduling are two interlinked components of the unified process of planning.

Misconception 9.

There exists an opinion that PREDICTION is a component of PLANNING and does not belong to other subsystems.

Several issues are confused in this opinion. PLANNING is a design of the future processes based upon some temporal model of the World. So, planning does include PREDICTION. The latter has already been incorporated in the temporal model of the World[76] . Thus, PREDICTION is performed only when the temporal model of the World is being created. However, EXECUTOR must also be able to predict. To compute the feedback compensation command (which is one of the EXECUTOR functions), it must predict how the process develops so that the compensation could be introduced properly. Evaluation of the derivative in the PID controllers play the role of a short-term predictor and should be performed as a part of the EXECUTOR's activities.

Maintenance of the World Model should include storage and organization, as well as recognition of the hidden tendencies and evaluation of the expected changes, i.e. prediction. Indeed, if some tendencies of change are detected in the World Model, the subset of it submitted to Planner to use, should reflect these tendencies. Planner is supposed to synthesize the motion trajectory and not be involved in the analysis of tendencies of the changes in the World Model. The World Model predicts the tendencies of changes as well as the results of hypothetical events.

Misconception 10.

There exists an opinion that hierarchies and hierarchical systems are arbitrary or somewhat outmoded concepts.

It became very fashionable to consider a swarm of autonomous agents to be the desirable architectural solution. It is expected that this undifferentiated swarm will produce

[76] See J. Albus, "A Reference Model Architecture for Intelligent Systems Design" [107]

the output behavior that is more reliable, robust, and possibly, even more efficient. This expectation is a mistake.

Hierarchies and hierarchical systems emerge as a result of conscious and/or natural design of systems (e.g. design via evolution and selection) because this is the only way to achieve the highest possible efficiency in the system with limited resources. The uncoordinated activities of autonomous agents can work only for simple cases. Even those enthusiastically working in the area of "autonomous agents" have come to the conclusion that as soon as the problem becomes goal oriented and complex, the hierarchy emerges with necessity.

Some objections against the hierarchical architectures are based upon the fact of existence of inefficient hierarchies. Yes, inefficient hierarchies exist. However, the maximum of efficiency can be achieved only in a hierarchical system. The last misconception, emerges sometimes when only one facet of the hierarchical system is considered: the "authority" allocated at a level of resolution while all other laws, properties, and responsibilities are neglected.

Misconception 11.

The phenomena of Task Decomposition and Multigranular (Multiscale, Multiresolutional) Knowledge Representation are unrelated.

The tree of Task Decomposition is tightly linked to the phenomenon of multiresolutional world representation. To understand this connection, one should consider any object-oriented hierarchy as a result of the evolutionary design in which only efficient solutions to survive. Because the maximum of efficiency can be achieved only by introducing a multiresolutional structure which delivers such maximum (proper multiresolutional structure), only those solutions survive which fit within the proper structure. These solutions remain as the words in the vocabulary, as engineering and/or biological artifacts and become the tools for other systems construction.

For the newly designed systems, the words-artifacts, as well as the tasks-artifacts, do not necessarily fit within the proper hierarchy for them. At this point, the trade-off should be

achieved. The designer should decide what is more beneficial: to use the existing artifacts (words, objects, facts) which make the efficiency lower than the ideally achievable, or to recreate the vocabulary which will allow for building the proper hierarchy, achieve the maximum of efficiency but will be substantially more expensive. In the reality, the designers combine both the existing and the novel components.

Misconception 12.

Behavior of the system is considered to be an internal rather than external phenomenon.

Indeed, there are authors who call "behaviors" elements of the architecture (autonomous agents sometimes are called "behaviors." One can even find an expression "to build an architecture out of behaviors"). This is a misleading interpretation of the scientific term which has a well-established meaning totally consistent with its application both in robotics and in the everyday life.

Random House Dictionary of the English Language (1987) gives the following definitions: behavior- "manner of behaving or acting; observable activity in a human or animal; the aggregate of responses to internal or external stimuli; a stereotyped species-specific activity". Webster (New Universal Unabridged, 1983) says: "It (behavior) expresses external appearance of action. In this sense it is used also to inanimate objects; as, the behavior of a ship; the behavior of a magnetic needle". The Penguin Dictionary of Psychology[77] has the following definition: "Behavior- A generic term covering acts, activities, responses, reactions, movements, processes, operations, etc., in short, any measurable response of an organism".

A. Newell and H. Simon have a similar vision of this concept. They indicate that for behavior planning, it is very important to consider alternatives of behavior and to map them from the problem space into the space of behavior: "We shall find it necessary to describe not only his (the subject's. A. M.) actual behaviors but the set of possible behaviors from

[77] See Subsection 5, Planner.

which these are drawn; and not only his overt behaviors, but also the behaviors he considers in his thinking that do not correspond to possible overt behaviors."[78] Formally, it was represented by A. Sloman[79] in the form

$$P, B(x)\text{-->}M(x)$$

where P- is the behavior instantiating program,

B(x)- is the behavior of the agent x,

M(x)- are the mental states related to the agent x.

One can see that *mental states* (as an internal phenomenon) are clearly separated from the *behavior* which is an external phenomenon and should be addresses as such.

---

[78] A. S. Reber, The Penguin Dictionary of Psychology, Penguin Books, London, 1985

[79] A. Newell, H. Simon, Human Problem Solving, Prentice Hall, 1972, p. 59

Appendix IV

Search for the Desirable State Space Trajectory

Any search for the optimum state space (including "input" and "output") trajectory can be considered a particular case of the State Space Search. The latter is performed by the following algorithm.

An Algorithm of State Space Search (S^3-algorithm)

Stage I. Establish the state space in which the State Space Search procedure will be executed.

Step 1 Name and list the inputs and their operating intervals

Step 2 Name and list the outputs and their operating intervals

Step 3 Formulate all mappings required for the system to be analyzed:

Comment:the request is sent to the World Model which submits the bulk of available information including the differential and algebraic equations; inequalities and logical statements known from experience; set of preferable "traces".

Step 4 Select the cost function and the form of cost-functional in which all costs will be computed.

Step 5 Declare the data structures in which the states will be analyzed.

Comment: in the test example of applying this algorithm, the following structures were created:

(a) a node adjacency storage for the graph representation of the state space.

(b) a cost record which holds data indicating the node number, its cumulative cost from the source node, and the index which allows to find the cost offset array with this node and the index that shows whether the node is permanent or temporary.

(c) a structure which contains only temporary nodes, its successor and predecessor (list OPEN).

(d) results of least-cost path algorithm which contains a node number, a corresponding cost, its successor and predecessor, for all permanent nodes (list CLOSED).

(e) list of the paths at different resolution levels (PDR)

(f) list of the vicinities for each node stored in the list CLOSED.

Stage II. Execute the State Space Search Procedure

Step 1 Generate random nodes at the required density within the envelope under consideration (and store them).

Comments:

(a) Density is estimated by the number of nodes per unit of the area[80]. The value of density is known only for the final iteration of the algorithm: it is determined by the required accuracy of computation. The initial density is selected in such a way as to have the average distance between two points in the random graph less than or equal to 10% of the interval of computation.

(b) The initial area of computation is determined by the full intervals of all inputs and all outputs.

(c) The second, third, (and so on until the final) zones in which the random nodes are generated are equal to the "vicinity" of the node which is introduced in Step 2.

---

80 A. Sloman, "Did Searle Attack Strong Strong or Weak Strong AI", eds. A. Cohn, J. Thomas, Artificial Intelligence and Its Applications, Wiley,1986

Step 2 Generate the vicinity for the first node.

Comments:

Vicinity is a part of the state-space adjacent to the current point in which all random nodes previously generated are considered to be connected to the current point; in the case of this program, the vicinity is bounded by an ellipse.

Step 3 Store the condition of vicinity in the form of an equation (or a set of equations) of line(s) which serve for subsequent comparison of all nodes of the random graph whether they are within the vicinity or outside.

Step 4 Check all nodes of the random graph and extract from them a subset which is located within the vicinity.

Comment:

(a) Only the subset located within the vicinity of the current node is considered the "successor" of the current node.

Step 5 For the set of successors, do the following:

(a) compute the cost of motion from the current node to the particular successor.

(b) store the coordinates of all of the successors with their costs in the list OPEN.

(c) remove from the list OPEN the partial path (PP) node for which cost is minimum; resolve ties arbitrarily.

Step 6 IF PP is the goal node, GOTO 2.9 with the solution.

Step 7 ELSE Put PP on the list CLOSED.

Step 8 GOTO Step 2., considering the point from Step 7 the initial point

Step 9 Put the solution on the PDR list.

Comment:

(a) Optional: draw the path at a particular resolution

(b) Clearly, for the best path which has been found at a particular resolution, we have the string of their nodes and the vicinity constraints

which have been remembered from above.

Step 10 Put k random points in each vicinity of all points of the path under consideration; in the areas where vicinities overlap, merge all couples of points which have distance between them smaller than the distance at the level of resolution under consideration. Consider the union of all these points as a new random graph.

Comments:

(a) The number n of points to be put in the vicinity depends on the resolution level; ratio between average distances between two points in the graph at different resolution levels is selected from design considerations which are addressed separately.

(b) Merger of the points which are located too close to each other can be done in different ways: by extinguishing one of these two points, by finding a midpoint, etc.

Step 11 IF the density in the new vicinity is higher than is required by conditions of accuracy, THEN exit with FINAL SOLUTION

Step 12 ELSE GOTO Step 2 considering the new vicinity size.

Searching for the input command sequence when the output trajectory is preassigned

The simplest possible search technique which can be applied to a "single input-single output" (SISO) system, is dichotomy (binary search) or DS. It is an exhaustive method which provides a guaranteed admissible solution if it exists, and which is accurate to the arbitrary non-zero value of indistinguishability zone. Let the output trajectory y*[k] be defined for some finite k in the interval $0 \leq k \leq p$ and let the desired command tracking accuracy be defined as

$$\| y^*[k] - y[k] \| \leq e$$

Also, let the admissible set of inputs be defined as u such that $U_{min} \leq u \leq U_{max}$. which is a statement of the usual physical constraints on inputs to real systems.

DS involves the following algorithm:

Step 1. For each k in the interval $0 \leq k \leq p$, beginning at

Step 2. Test the midpoint of the interval [Umin , Umax] by selecting

$u_{ave} = (Umin + Umax) / 2$ and applying it to the model of the system.

Step 3. Compare the sign (sgn) and magnitude $| \bullet |$ of the error given by

$E = y^*[k] - y_{ave}[k]$

Step 4. If $| E |$ is less than or equal to e, record u[k] and continue with the next k.

Else if $sgn(E) < 0$,

set $U_{max} = U_{ave}$ and go to step 2.

Otherwise,

set $U_{min} = U_{ave}$ and go to step 2.

This Figure illustrates the method

Two last steps can be modified as follows:

Step 3. Compare the magnitude $\| \bullet \|_2$ of the error given by

$Emin = y^*[k] - y_{min}[k]$, and $E_{max} = y^*[k] - y_{max}[k]$

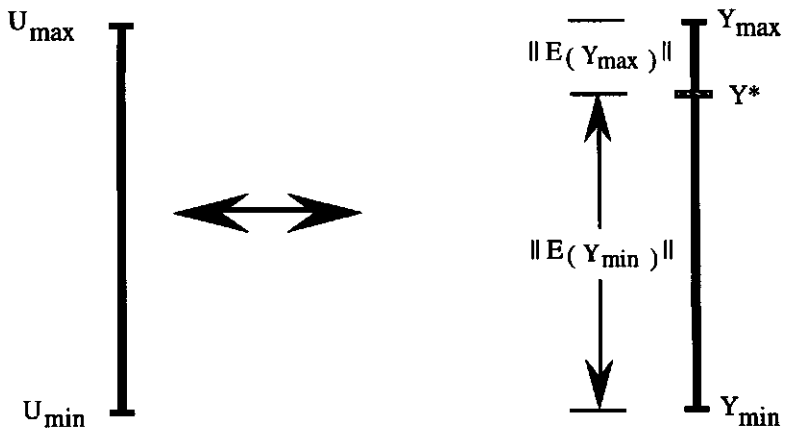
Step 4. If $\| E \|$ is less than or equal to e, record u[k] and continue with the next k.

Else if $\| E_{min} \| \leq \| E_{max} \|$,

set $U_{max} = (U_{min} + U_{max}) / 2$ and go to step 2.

Otherwise,

set $U_{min} = (U_{min} + U_{max}) / 2$ and go to step 2.



A further possible modification is to replace the termination condition of both these algorithms with a loop counter and to stop the refinement of the intervals of search after a fixed number of iterations instead of trying to achieve an arbitrary e.

The reason for the choice of the Euclidian norm for reducing the search space was that it appeared to afford a better opportunity to extend the algorithm to more than one dimension. Before this purely intuitive approach, which has not yet been exhaustively analyzed, is described, it is useful to consider the proof of the Euclidian algorithm.

For the system described above, the outputs due to two inputs u_1 and u_2 applied at time 'k' and beginning from the same state 'x' may be written as:

$$y_1[k+1] = \mathbf{CA}x[k] + \mathbf{CB}u_1[k]$$

and

$$y_2[k+1] = \mathbf{CA}x[k] + \mathbf{CB}u_2[k]$$

Now, we can write

$$E_1[k+1] = \mathbf{CB}[\, u^*[k] - u_1[k] \,]$$

and

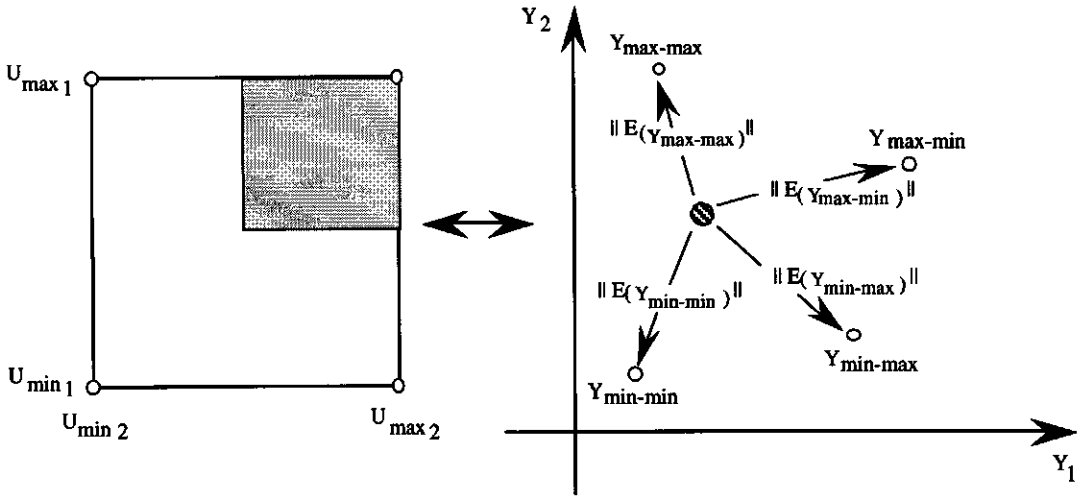
$$E_2[k+1] = \mathbf{CB}[\, u^*[k] - u_2[k] \,]$$

Clearly, if $\| E_1[k+1] \| \leq \| E_2[k+1] \|$ and u^* is in the interval $[u_1, u_2]$, then

$$| u^*[k] - u_1[k] | \leq | u^*[k] - u_2[k] |$$

and a new interval of search containing u* may be defined by $[u_1, (u_1+u_2)/2]$.

Unfortunately the same argument does not carry over to the multidimensional case because of the interactions between the outputs of a system. A hypothetical situation is illustrated in Figure below:



The selection of the upper right quadrant of the two dimensional input space (marked by the dotted line) on the basis of the smallest error from amongst the tested input/output pairs is not an approach which can be guaranteed.