A Theory of Intelligent Systems

James S. Albus

Robot Systems Division
National Institute for Standards and Technology

Abstract

A theoretical model is proposed consisting of seven basic elements: actuators, sensors, sensory processing, world modeling, task decomposition, value judgment, and global memory/communications. These elements are integrated into a hierarchical system architecture wherein: a) control bandwidth decreases about an order of magnitude at each higher level, b) perceptual resolution of spatial and temporal patterns contracts 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 expand in space and time by about an order-of-magnitude at each higher level. At each level, tightly coupled functional modules perform task decomposition, world modeling, sensory processing, and value judgment. Feedback control loops are closed at every level.

The Elements of Intelligent Systems

The elements of intelligent systems and their relationship to each other are illustrated in Figure 1.

a. ACTUATORS -- Output from an intelligent system derives from 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 tens, hundreds, or even thousands of actuators, all of which must be coordinated in order to perform tasks and accomplish goals. Natural actuators are muscles and glands. Machine actuators are motors, pistons, valves, solenoids, and transducers.

b. SENSORS -- Input to an intelligent system derives from 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.

c. SENSORY PROCESSING -- Perception takes place in a sensory processing element that compares 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 may include recognition of speech and interpretation of language and music.

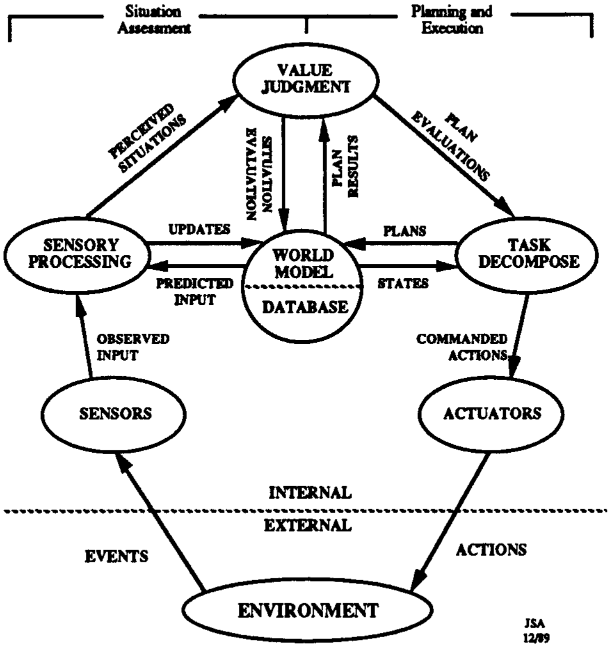


Figure 1. The elements of intelligence and the functional relationships between them.

d. 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 thus can provide 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 task decomposition element, so that it can make intelligent plans and behavioral choices, and to the sensory processing element, in order for it to perform correlation, model matching, and model based recognition of states, objects, and events. The world model is kept up-to-date by the sensory processing element.

Proceedings of the Fifth IEEE International Symposium on Intelligent Control, Philadelphia, PA, September 5-7, 1990

e. VALUES -- Value judgments determine what is good and bad, rewarding and punishing, important and trivial, certain and improbable. The value judgment element 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 also computes the probability of correctness and assigns believability and uncertainty parameters to state variables. The value judgment element thus provides the basis for choosing one action as opposed to another, or for pursuing one object and fleeing from another. Without value judgments, any biological creature would soon be eaten by others, or destroyed by its own inappropriate actions.

f. TASK DECOMPOSITION -- Behavior is generated in a task decomposition element that plans and executes tasks by decomposing them into subtasks, and by sequencing these subtasks so as to achieve goals. Goals are selected and plans generated by a looping interaction between task decomposition, world modeling, and value judgment elements. The task decomposition element hypothesizes plans, the world model predicts the results of those plans, and the value judgment element evaluates those results. The task decomposition element then selects the plans with the best evaluations for execution. The task decomposition element also monitors the execution of plans, and modifies existing plans whenever the situation requires.

In systems with higher intelligence, the task decomposition element has the ability 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. Task planning and execution often must be done in the presence of uncertain, incomplete, and sometimes even incorrect or misleading information in the world model.

The System Architecture of Intelligence

Each of the elements of intelligent systems 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 components to interact and communicate with each other in intimate and sophisticated ways.

A system architecture is what partitions the elements of intelligence into computational modules, and interconnects the modules in networks and hierarchies. It is what enables the task decomposition element 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. It is what enables the world model to answer queries from task decomposition modules, and make predictions and receive updates from sensory processing modules. It is what communicates the value state-variables that describe the success of behavior and the desirability of states of the world from the value judgment element to the goal selection subsystem.

A number of system architectures for intelligent machine systems have been proposed, and a few have been implemented. [1-9] The model of intelligent machines that will be discussed 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). RCS was first implemented by Barbera for laboratory robotics in the mid 1970's [6] and adapted by Albus, Barbera, and others for manufacturing control in the NIST Automated Manufacturing Research Facility (AMRF) during the early 1980's [10,11]. Since 1986, RCS has been implemented for a number of additional applications, including the NBS/DARPA Multiple Autonomous Undersea Vehicle (MAUV) project [12] and the Army TMAP and TEAM semi-autonomous land vehicle projects. RCS also forms the basis of the NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM) being used on the space station Flight Telerobotic Servicer [13].

The system architecture described in this paper organizes the elements of intelligence shown in Figure 1 into a layered hierarchy of processing nodes as shown in Figure 2. Each node contains four types of computing modules: task decomposition (TD), world modeling (WM), sensory processing (SP), and value judgment (VJ) modules. These are richly interconnected to each other by a fifth module, global memory (GM), that provides interprocess communications and database functions. All communications flow through the GM modules, such that the GM modules appear like a global memory to the TD, SP, WM, and VJ modules. At the lowest level, outputs from TD modules drive actuators, and input to SP modules arrives from sensors.

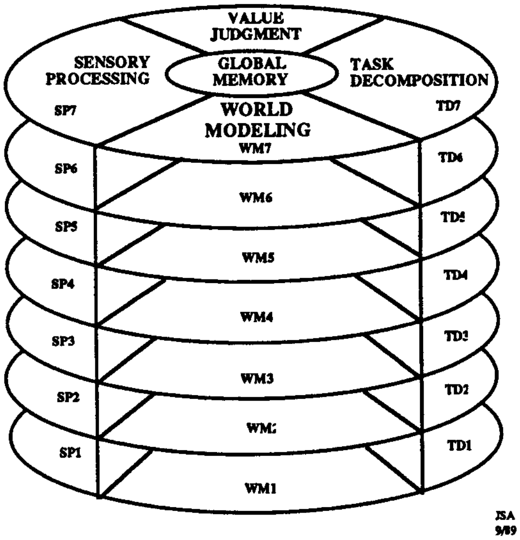


Figure 2. A hierarchy of processing modules composed of four types of computing modules: task decomposition (TD), world modeling (WM), sensory processing (SP), and value judgment (VJ). These are interconnected to each other through a fifth global memory (GM) module.

In a biological brain, the function of the GM modules are provided by axon pathways as shown in Figure 3. In artificial systems, the physical implementation of global memory may be a common memory, a message passing system, or some combination thereof. In either biological or artificial systems, a GM module may function as a communications processor, file server, database management system, indirect addressing or list processing engine. The input/output relationships of the GM module produce the effect of a virtual global memory, or blackboard system [14].

Computing modules and the communications pathways between them define a layered graph, or lattice, of nodes and directed arcs, as shown in Figure 4. Each layer contains one or more nodes which correspond to computational subdivisions of the sensory-motor system. Each node contains a TD, SP, WM, VJ, and GM submodule.

Arcs connecting the nodes correspond to communication paths. Horizontal arcs carry information such as questions, answers, and shared data. Downward flowing vertical arcs carry commands, priorities, and database configuration information. Upward flowing vertical arcs carry status and processed sensory information. Output from the bottom level TD modules drive actuators. Input to the bottom level SP modules convey data from sensors.

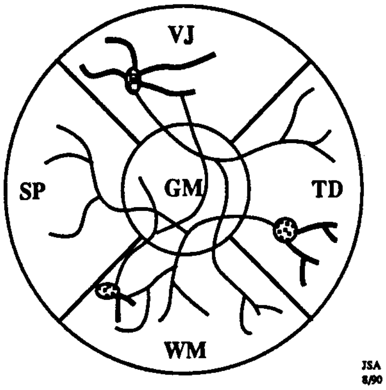


Figure 3. Neural pathways through the global memory GM module that interconnect the TD, WM, SP, and VJ modules. The thin lines denote axons. The thick lines denote dendrites. The filled ovals denote cell bodies. To neurons in the TD, WM, SP, and VJ modules, outputs go to the GM and inputs arrive from the GM. The GM module therefore looks to them like a global memory. In fact, the GM acts as a communications system, or database server.

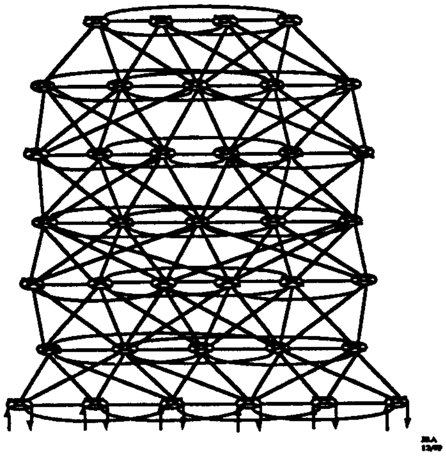


Figure 4. Each layer of the system architecture contains a number of nodes, each of which contains TD, WM, SP, VJ, and GM modules. The nodes are interconnected as a layered graph, or lattice, through the GM modules. Outputs from the bottom layer TD modules drive actuators. Inputs to the bottom layer SP modules convey data from sensors.

At at any instant in time, the computational modules of Figure 4 organize themselves such that the TD modules form a command tree, as shown in Figure 5. In this configuration, each TD module receives task command names from one and only one supervisor, and outputs subcommand names and parameters to one or more subordinate TD modules at the next lower level in the tree. A TD module may receive command parameters (but not command names) from more than one higher level module.

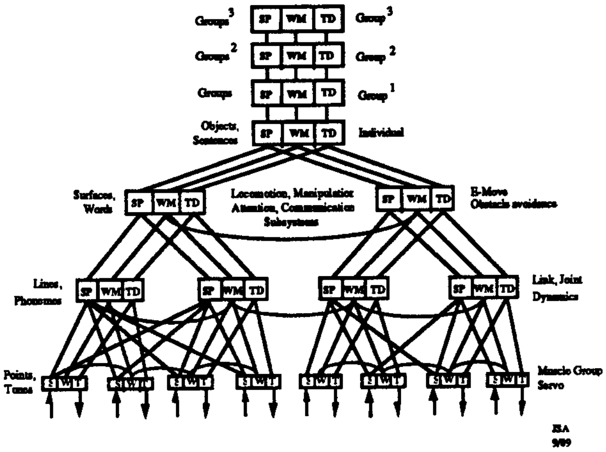


Figure 5. An organization of processing nodes such that the TD modules form a command tree. The VJ and GM modules are hidden behind the WM modules. On the right are the functional characteristics of the TD modules at each level. On the left are the type of visual and acoustical entities recognized by the SP modules at each level. In the center of level 3 are the type of subsystems represented by processing nodes at level 3.

Levels in the command 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.

The SP modules are also organized hierarchically (but not as a tree) in that sensory information is processed into increasingly higher levels of abstraction. 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.

World model information is also organized hierarchically so as to service the needs of both SP and TD modules at the various levels.

Hierarchical vs. Heterarchical

Figures 4 and 5 illustrate both the hierarchical and heterarchical (horizontal) relationships involved in the proposed architecture. The architecture is hierarchical in that commands and status feedback flow hierarchically up and down a task decomposition 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 proposed architecture is heterarchical (or 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 task decomposition, world modeling, sensory processing, and value judgment modules. For example, the TD modules communicate voluminously with WM modules at the same level, asking questions and responding with answers. SP modules also communicate intensively with WM modules at the same level, sending sensory updates and accepting predictions of sensory input.

Thus, although the proposed architecture incorporates vertically structured task decomposition and sensory processing hierarchies, there exists a massive horizontal flow of information between TD, WM, SP, and VJ modules at the same level, especially within the same command subtree. The horizontal flow of information is most voluminous within a single node, or between related nodes in the same command subtree. The communications bandwidth is much less between computing modules in separate command subtrees. Communications bandwidth is indicated in Figure 5 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.

Time Dependencies

Relationships between modules within and between layers of the hierarchy may be reconfigured from time to time to in order to accomplish different goals, priorities, and task requirements. Thus, although the TD modules are organized into a command tree at each instant of time, the command tree is not necessarily stationary with time. This means that any particular TD module 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).

Command tree reconfiguration is implemented through the interconnections (shown in Figure 4) that exist between TD modules at different hierarchical levels. These enable each TD module to receive input messages from several different supervisors at the same time. It is the responsibility of the job assignment and planning submodules in each TD module to assess priorities, negotiate for resources, and coordinate task activities such that a TD module never attempts to execute commands from more than one supervisor at a time.

Theorem: For task decomposition to succeed in a dynamic and unpredictable world, it must be accomplished in a hierarchy of temporal and spatial levels of resolution.

Planning complexity grows exponentially with the number of steps in the plan (i.e. the number of layers in the search tree). If real-time planning is to succeed, any given planner must operate in a limited search space. If there are too much resolution in the time line, or in the space of possible actions, the size of the search tree 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 so that at each layer no planner needs to search more than (for example) ten steps deep in a game tree, 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 characteristic temporal planning horizons and characteristic degrees of detail for each level.

Theorem: Intelligence increases with each additional level in a hierarchically layered architecture wherein:
a) control bandwidth decreases about an order of magnitude at each higher level,
b) perceptual resolution of spatial and temporal patterns contracts 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 experimental data indicate perceptual and behavioral "chunking" of both temporal and spatial entities in groups of about five to ten [15]. This data supports conditions b), c), and d) above.

A Timing Diagram

Based on the above theorems, we can construct a timing diagram for the proposed hierarchical architecture, as shown in Figure 6. The range of the time scale, and hence the planning horizon and event summary interval increases exponentially by about an order of magnitude at each higher level. The loop bandwidth and frequency of subgoal events decreases exponentially at each higher level. The seven hierarchical levels in Figure 6 span a range of time intervals from three milliseconds to one day. Three milliseconds was chosen as 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. Different tasks may require additional levels to be inserted in (or deleted from) various command subtrees.

The origin of the time axis in Figure 6 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.

Figure 6 suggests a duality between the task decomposition 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.

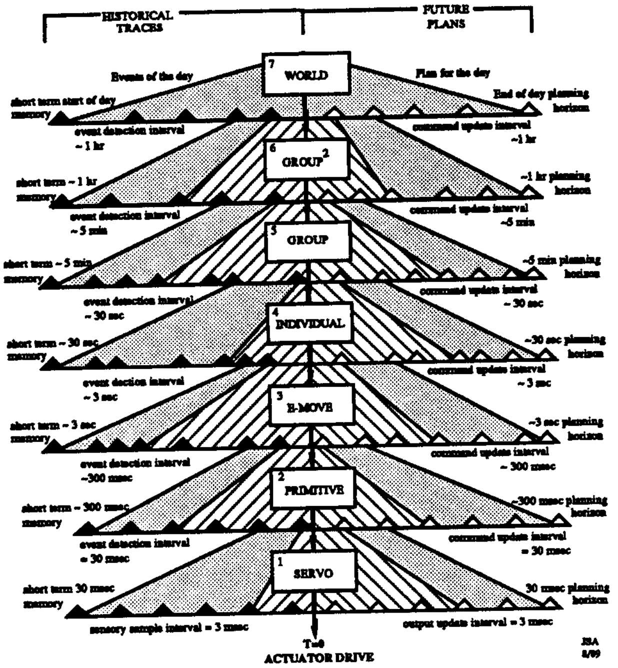


Figure 6. A timing diagram illustrating the temporal flow of activity in the task decomposition and sensory processing systems. At the world level, high level sensory events and circadian rhythms react with habits and daily routines to generate a plan for the day. Each element of that plan is decomposed through the remaining six levels of task decomposition into action.

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.

At each level, plans consist of at least one, and on average 10, subtasks. The planners have a planning horizon that extends about one average input command interval into the future. Figure 7 illustrates this principal.

In a real-time system, plans must be regenerated periodically to cope with changing and unforeseen conditions in the world. Replanning may begin at cyclic intervals, or whenever necessary. 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 is possible only if the depth of search is limited through hierarchical planning.

Plan executors at each level have the task of reacting to feedback every control cycle interval. Control cycle intervals are thus inversely proportional to the control loop bandwidth.

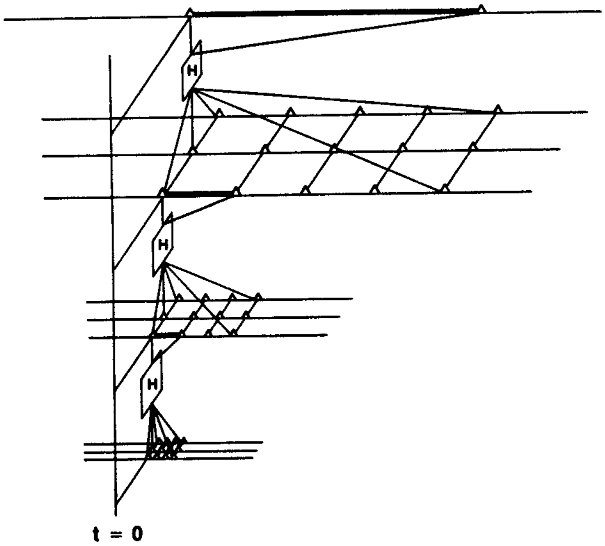


Figure 7. Three levels of real-time planning illustrating the shrinking planning horizon and greater detail at successively lower levels of the hierarchy. At the top level a single task is decomposed into a set of four planned subtasks for each of three subsystems. At each of the next two levels, the first task in the plan of the first subsystem is further decomposed into four subtasks for three subsystems at the next lower level.

Typically the control cycle interval is at least ten times 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.

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; i.e. 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.

Task Decomposition

Task decomposition is inherently a hierarchical process. At each level of the task decomposition hierarchy, tasks are decomposed into subtasks that become task commands to the next lower level. At each level of a task decomposition 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.

Task decomposition 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 TD module, and each branch of the tree corresponds to a communication link in the chain of command.

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 8.

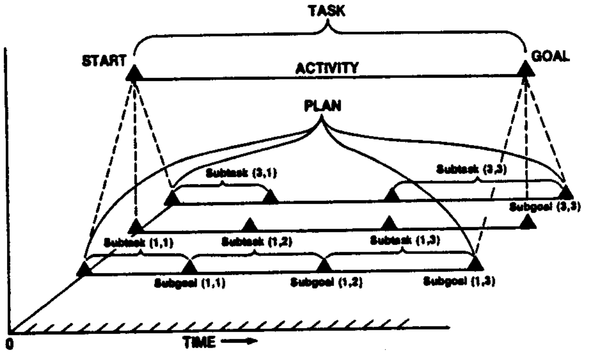


Figure 8. A task consists of an activity which typically begins with a start event and is terminated by a goal event. A task may be decomposed into several concurrent strings of subtasks which collectively achieve the goal event.

In a plan involving concurrent job activity by different subsystems, there may be 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 and cooperative action by several subsystems. This requires that both planning and execution of subsystem plans be coordinated.

There, of course, may be several alternative ways that a task can be accomplished. Alternative task or job decompositions can be represented by AND/OR graphs in the procedure section of the task frame.

Axiom: For any intelligent system, there exists a set of tasks that the system knows how to do.

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 a creature capable of learning, the task vocabulary is not fixed in size. It can be expanded through learning, training, or programming.

Definition: A task is a piece of work to be done, or an activity to be performed.

Typically, a task is performed by a one or more actors on one or more objects. The performance of a task is usually described as an activity which begins with a start-event and is directed toward a goal-event. This is illustrated in Figure 8.

Definition: A goal is an event which successfully terminates a task. A goal is the objective toward which task activity is directed.

Definition: A task command is an instruction to perform a named task. A task command may have the form:

DO <Taskname> AFTER <Start Event> UNTIL <Goal Event>

Definition: Task knowledge is knowledge of how to perform a task, including information as to what tools, materials, time, resources, information, and conditions are required, plus information as to what costs, benefits and risks are expected.

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

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 [16] 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	-- 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 -- conditions that must be satisfied to begin the task -- information that may be required
procedures	-- a state graph 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. 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 others. 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.

In most cases, the ability to successfully accomplish complex tasks is more dependent 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.

TD modules

In the control architecture defined in Figure 5, each level of the hierarchy contains one or more TD modules. At each level, there is a TD module for each subsystem being controlled. The function of the TD modules are to decompose task commands into subtask commands.

Input to TD modules consists of commands and priorities from TD 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 TD modules may consist of subtask commands to TD 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.

In general, each TD module at each level consists of three sublevels as shown in Figure 9.

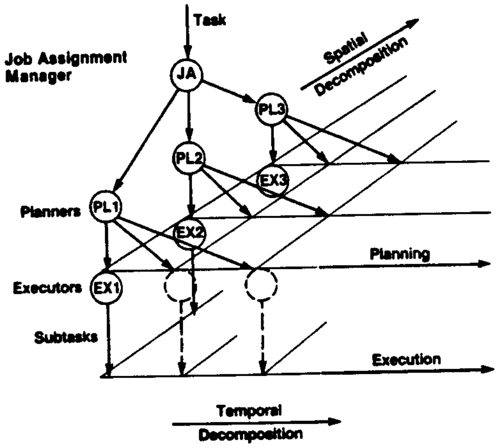


Figure 9. The job assignment JA module performs a spatial decomposition of the task into jobs for j subsystems. For each subsystem, a planner PL(j) performs a temporal decomposition of its assigned job into subtasks. For each subsystem, an executor EX(j) closes a real-time control loop that servos the subtask to the plan.

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 TD module.

The JA submodule is also responsible for assigning tools and allocating 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 <point camera> task command.

The job assignment submodule may also select the coordinate system in which the task decomposition at that level is to be performed.

2) the planner sublevel -- PL(j) submodules j = 1, 2, . . , N

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

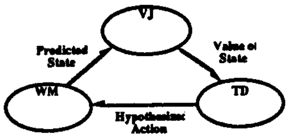
Task planning may be accomplished by case-based planners that simply select partially or completely prefabricated plans from a library of plans, scripts, or schema [17,18]. This may be done by evoking a rule of the form, IF(case_x)/THEN(use_plan_y). The planners may complete partial plans by providing 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.

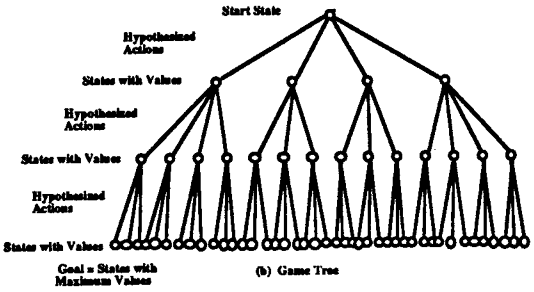
Task planning may also be accomplished by search-based planners that search the space of possible actions. This requires the evaluation of alternative hypothetical sequences of subtasks, as illustrated in Figure 10. Each planner PL(j) hypothesizes some action or series of actions, the WM module predicts 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 10(a). This results in a game (or search) tree, as shown in 10(b). The path through the game tree 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 10(c).

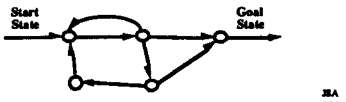
A job command to a subsystem planner may contain constraints on time, or specify job-start and job-goal events. A job assigned to one subsystem may also require synchronization with other jobs assigned to different subsystems. These constraints and requirements may be specified by, or derived from, the task frame. Each planner PL(j) submodule is responsible for checking its plan against plans generated by each of the other N-1 planners at the same level to determine if there are mutually conflicting constraints. If a conflict is found, constraint relaxation algorithms [19] may be applied until a solution is found. If no solution can be found, the planner reports failure to its job assignment submodule, and a new job assignment may be tried.



(a) Planning Loop



(b) Game Tree



(c) Plan Graph

Figure 10. The planning loop (a) produces a game tree (b). A trace in the game tree from the start state to a goal state is a plan that can be represented as a plan graph (c). Nodes in the game tree correspond to edges in the plan graph, and edges in the game tree correspond to nodes in the plan graph. Multiple edges exiting nodes in the plan graph correspond to conditional branches.

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

> There is an executor EX(j) for each planner PL(j). The executor submodules are responsible for successfully executing the plan state-graphs generated by their respective planners. When an executor is informed by the world model that a subtask in its 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. Its planner meanwhile 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 TD modules at level i-1.

Planners PL(j) operate on the future. Each planner is continuously preoccupied with assuring that there exists a plan for its subsystem to the end of its planning horizon. Executors EX(j) operate on the present. Each executor is constantly monitoring the current (t=0) state of the world as estimated by the world model from sensory data. 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 Task Decomposition Hierarchy

Task goals and task decomposition functions often have characteristic spatial and temporal requirements. For any task, there exists a hierarchy of task vocabularies that can be overlaid on the spatial/temporal hierarchy of Figure 6 [20].

For 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 servos the position, velocity, and force of individual actuators.

Level 2 is where commands for maneuvers of body components are decomposed into smooth coordinated dynamically efficient trajectories. Feedback servos coordinated trajectory motions.

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.

Level 4 is where commands for an individual intelligent machine to perform simple tasks on single objects are decomposed into coordinated activity of body locomotion, manipulation, attention, and communication subsystems. Feedback initiates and sequences task activity.

Level 5 is where commands for behavior of an intelligent individual relative to others in a small group is decomposed into interactions between the self and nearby objects or agents. Feedback steers small-group interactions.

Level 6 is where commands for behavior of the individual relative to multiple groups over longer time frames. Feedback steers large-group interactions.

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

The mapping of TD functionality onto levels one to four in this proposed hierarchy 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 class of intelligent machine system. 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 machine relative to others in groups, multiple groups, and the world as a whole.

World Modeling, Global Memory, and Value Judgments

The world model is an intelligent system's internal representation of the external world. It is the system's best estimate of objective reality. It provides an interface between sensory processing and task decomposition. The world model is hierarchically organized so as to provide multiple levels of resolution in space and time.

Knowledge in the world model database includes both a-priori information which is available to the intelligent system before action begins, and a-posterior knowledge which is gained from sensing the environment as action proceeds. World model knowledge includes information about space, time, entities, events, and states of the world, including states of the system itself. The correctness and consistency of world model knowledge is verified by sensory processing mechanisms that measure differences between world model predictions and sensory observations.

Global memory modules provide memory, communication, and switching services that make the world model behave like a global common memory in response to queries and updates from the TD, WM, SP, and VJ modules. The GM module in each node provides a communication window (the equivalent of a network terminal, or mailbox interface) into the global memory database for each of the TD, WM, SP, and VJ modules in that node.

Value judgments provide an evaluation of hypothesized plans, and perceived objects, events, and situations. These evaluations produce value state-variables that indicate cost, benefit, risk, priority, desirability, attractiveness, and uncertainty.

A detailed description of the proposed world model, global memory, and value judgment systems is contained in [21].

Sensory Processing

Sensory processing is the intelligent system's mechanism of perception. Perception is the establishment and maintenance of correspondence between the internal world model and the external real world. The function of sensory processing is to extract information about surfaces, entities, events, states, and relationships in the external world, so as keep the world model an accurate and up to date representation of the real world.

The sensory processing system is hierarchically organized. At each level, predictions from the world model are compared with observations from the sensory data stream. Differences are used directly to update the world model predictions. Similarities are integrated to produce both spatial and temporal correlations. When correlations rise above threshold, entities (or events) are recognized (or detected), and entered into the world model global memory database. A detailed description of the proposed sensory processing system is contained in [22].

Summary and Conclusion

While much remains to be discovered, much is already known, both about the mechanisms and function of intelligent machine systems. The study of intelligent machines is an extremely active field. There are major programs in Europe, Japan, and the United States on computer integrated manufacturing, robotics, and intelligent machines for a wide variety of military and commercial applications. Research in learning automata, neural nets, and brain modeling has given insight into learning and the similarities and differences between neuronal and electronic computing processes. Computer science and artificial intelligence is probing the nature of language and image understanding, and has 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 for fusing sensory input from multiple sources, and assessing the believability of noisy data.

Progress is rapid in many individual areas, and there exists an enormous and rapidly growing literature in each of the above fields. What is lacking is a general theoretical model of intelligent machine systems which ties all these separate bodies of knowledge into a unified framework.

This paper is an attempt to formulate at least the broad outlines of such a model. In its present form, the model is far from complete. None of the theorems in this paper have been proven in any formal sense. Many important issues remain uncertain and many aspects of intelligent behavior are unexplained. Nevertheless, the model does provide a framework for integrating many different concepts from many different fields. Hopefully, this framework will contribute to a better understanding of the nature of intelligent systems, and will lead to more widely accepted methods for designing and building intelligent machines.

REFERENCES

1 Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R., "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", Computer Survey, 23, pgs. 213-253, June 1980.

2. Laird, J. E., A. Newell, and P. Rosenbloom, SOAR: An Architecture for General Intelligence, Artificial Intelligence, 33, p. 1-64, 1987.

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. Greenwood, J.R., G. Stachnick, H.S. Kaye, "A Procedural Reasoning System for Army Maneuver Planning", Knowledge-Based Planning Workshop Proceedings, Austin, TX, December 1987.

6. Barbera, A.J., J.S. Albus, M.L. Fitzgerald, and L.S. Haynes, "RCS: The NBS Real-time Control System", Proceedings of Robots 8 Conference and Exposition, Detroit, MI, June 1984

7. Brooks, R., "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, Vol. RA-2, 1, March 1986

8. Saridis, G.N. "Foundations of the Theory of Intelligent Controls", Proceedings IEEE Workshop on Intelligent Control, 1985.

9. Mystel, A., "Intelligent Control in Robotics", Journal of Robotic Systems, 1988.

10. Simpson, J.A., 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.

11. Albus, J.S., 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.

12. Albus, J.S., "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles", National Institute of Standards and Technology, Technical Report 1251, Gaithersburg, MD, September 1988.

13. Albus, J.S., 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.

14. Hayes-Roth, B., "A Blackboard Architecture for Control", Artificial Intelligence, pgs. 252-321, 1985.

15. Miller, G. A., "The Magical Number Seven, Plus or Minus T: Some Limits on Our Capacity for Processing Information", The Psychological Review, 63, pp.71-97, 1956.

16. Minsky, M., "A Framework for Representing Knowledge", In *The Psychology of Computer Vision*, Winston (Ed.) (1975), 211-277

17. Schank,R.C., and Abelson, R.P., *Scripts Plans Goals and Understanding*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

18. Lyons, D.M. and Arbib, M.A., "Formal Model of Distributed Computation Sensory Based Robot Control", IEEE Journal of Robotics and Automation, 1988

19. Sathi, A. and M. Fox, "Constraint-Directed Negotiation of Resource Reallocations", CMU-RI-TR-89-12, Carnegie Mellon Robotics Institute Technical Report, March, 1989

20. Albus, J.S., *Brains, Behavior, and Robotics*, BYTE/McGraw-Hill, Peterborough, N.H.,1981

21. Albus, J., "The Role of World Modeling and Value Judgment in Perception", Proceedings IEEE Conference on Intelligent Control, Philadelphia, 1990

22. Albus, J., "Hierarchical Interaction Between Sensory Processing and World Modeling in Intelligent Systems", Proceedings IEEE Conference on Intelligent Control, Philadelphia, 1990