A Software Architecture for Integrated Service Robot Development

R.T. Pack, D. Mitchell Wilkes and K. Kawamura
Center for Intelligent Systems
Vanderbilt University
Nashville, TN 37235, USA
{pacman,wilkes,kawamura}@vuse.vanderbilt.edu
http://shogun.vuse.vanderbilt.edu/cis/

Abstract: *This paper presents a computational architecture that addresses the grand challenge problem of system integration. Robotics research has made many advances in sensory processing, control, and planning, but few efforts focus on the problem of dynamically integrating the "best available" approaches into a single architecture to support development of integrated systems. The role of architecture is discussed, and a novel approach to intelligent robotic system architecture is presented. This new approach has resulted from the maturation of object-based software technology.*

I. Grand Challenges

The 1996 International Conference on Robotics and Automation (ICRA) contained a panel discussion to define a set of Grand Challenges for the IEEE Robotics and Automation Society. From the comments of both the panelists and the audience a minimum set of grand challenge technologies [1] was identified including Human-Robot Interfaces, Modularity, and System Issues. In this paper we address system issues. During the panel discussion, several panelists argued that much of our research never makes it to the public/commercial sector despite all of the innovations made in robotics, computer science and other areas. Thus, the Grand Challenges were formulated to help focus research on real problems that prevent application development. From our experience there is no fundamental breakthrough that will suddenly yield intelligent robots in all applications. What is needed is a dedicated effort to address the problem of developing integrated solutions [0].

Service robots interact closely with humans in a wide range of situations, applying both skills and knowledge in cooperative tasks. Thus, the robot control system should provide the resources for intelligent action. These resources are present in the modern robotics literature as separate ideas, solutions, and mechanisms. What is needed is an architecture for coherently combining them into an integrated system.

Intelligent systems have been generally attributed the following properties (modified from Booch [3]):

- They pursue goals which change over time.
- They incorporate, use, and maintain knowledge.
- They exploit diverse, (ad hoc) subsystems embodying a variety of selected methods.
- They interact intelligently with humans and other systems.
- They allocate their own resources and attention.

The characteristic of primary importance is the ability to exploit diverse subsystems that embody a variety of methods. The ability to incorporate, use and maintain knowledge is but one of several possible methods for selection of robot behavior as pointed out by Agre [4] and several other methods have been used successfully [5, 6, 7]. Another important characteristic, the management of resources, has typically been solved by explicit planning; however new action selection mechanisms provide the opportunity to combine resource management, focus of attention and intelligent action under the umbrella of action selection or arbitration mechanisms.

II. Action Selection and Arbitration

A robot operates by selecting actions that will achieve tasks and goals. Maes describes this as the action selection problem [6]. All architectures must provide a solution for this problem. Identifying the action selection problem places the focus on building software that provides resources and mechanisms for action selection, thus while approaches may vary, action selection or more generally action arbitration is the bottom line for intelligent activity. Arbitration is not limited to the actuators of the system. It can be applied at various levels within a system to build up resources for action selection (e.g., a sensor fusion process), allocate resources (e.g., task/goal arbitration or planning), and control actuators (motor action arbitration). Figure 1, adapted from Bagchi [8], shows a generalized architecture that uses action selection to build up a robot abstraction for the tasks and goals of the system as well as to handle task and goal selection.

Arbitration mechanisms form an important element of intelligent system behavior at many levels. In sensory processing, action selection results in sensor fusion. Sensor arbitration combines sensor data into logical sensors, much like those developed by Luo [9], to be used by other modules. In the motion control domain, arbitration combines influences on motion (e.g., goal points, obstacles) to yield an overall motion that simultaneously meets several goals for the robot, similar to motor schema [7]. For planning and sequencing, the arbitration mechanism predicts or activates a sequence of operations as shown by Bagchi [8]. Thus action selection or arbitration is pervasive in the design of intelligent robots that include complex motion control, sensor fusion, behavior sequencing and task planning. The type of action selection mechanism used depends on the flexibility and structure of the system architecture. Rigid architectures typically allow only a single mechanism, while more flexible architectures combine several arbitration mechanisms at different points in the system.

III. System Decomposition and Architecture

Integrated system design presents a monumental software engineering project for any group attempting to develop a system of appreciable size. This problem is compounded by the fact that most task and problem descriptions are in a proscriptive format, while most software tools provide a prescriptive interface to the system, what Booch [3] calls "impedance mismatch" between specification and design in software systems. To manage this problem, a good system architecture is essential. An architecture is a set of organizing principles and core components that are used to build the basis for the system [10]. By specifying the types of components and their interactions, a robot abstraction is decomposed into more

0-7803-4053-1/97/$10.00 © 1997 IEEE

primitive elements that serve as a model for the physical resources, skills, behaviors and tasks that are developed in the control system. This forms a set of building blocks and interactions for constructing intelligent control software.

The architecture should be sufficiently flexible to support a wide variety of robots, components, subsystems, etc., with regard to physical integration. Additionally, it should support multiple, parallel paths from sensing to action that may use different algorithms or mechanisms from the robotics literature. Also, it should be expandable, extensible and scaleable.

Knowledge-based system decompositions lead to architectures that excel at applying knowledge. They are typically based on the physical symbol system hypothesis [11]. Given an appropriate world model these systems can solve intricate problems. The success of these methods in problem solving and cognitive modeling has given them the reputation as the best path toward general intelligence for robots. The key architectural feature of these methods is that the components in the system work by exchanging, and manipulating knowledge, which is typically represented by some linguistic or graph based formalism as in ACT*[12] or SOAR [13]. This knowledge is used for arbitration and communication among agents. Arbitration is handled by a uniform inference engine or production system that operates on this encoded knowledge.

Newer behavior-based decompositions create architectures that depart from the concept of combining "knowledge," which is fraught with representation problems. Instead, several behavior sources combine to produce the resulting behavior of the robot. The available arbitration mechanisms for combining the behaviors include strict priority [5], voting schemes [14], and a spreading activation mechanism [6], but in each approach the principle is the same.

The success of highly reactive behavior-based systems prompted several groups to adopt a hybrid approach that combines knowledge-based and behavior-based architectural elements with system elements of each type, such as the work on integrating knowledge with behaviors in AuRA [15,7]. Other hybrid deliberative/real-time approaches[16, 17] build up layered systems in which slower deliberative (i.e., knowledge-based) components periodically modify the parameters of a more reactive (behavior-based) component beneath them.

While other approaches separate action arbitration and knowledge, the object-based approach of this paper brings knowledge and action together into active agents. An object-based decomposition incorporates both knowledge-based features and behavior-based features and is a type of hybrid approach. The integrated system design problem is then to design a set of agents and relationships that support many levels of arbitration and exhibit intelligent behavior on a robot.

IV. A Service Robotics Benchmark

We are currently developing demonstrations for our service robot system using a new architecture. What follows is a description of the scenario we have chosen to help us develop and integrate service robot technology, including human interfaces. Our system uses ISAC, a dual-arm humanoid robot, and Helpmate, a mobile robot with one arm, as a service robot team. This benchmark was selected because there is direct human interaction, a relatively unstructured environment, and simplifying assumptions in the scenario can be gradually relaxed to increase problem complexity as our development proceeds. First, we briefly describe the structure and hardware of the robots and then we describe a service robot scenario that we are using as our test-bed for development.

ISAC is a stationary humanoid with two 6 DOF SoftArms and a pan-tilt-verge active camera head. The lowest level software drivers for the hardware are accessed by resource objects. All of the processing for robot control is handled by standard PC hardware, including the control for the manipulators and image processing operations. At a higher level we have implemented several skill objects such as visual servoing, skin detection, human face localization and tracking, hand/face following, object localization, grasping, and force controlled motions. Helpmate is a Yaskawa Helpmate mobile service robot. We have fitted Helpmate with an arm to allow it to be used to "fetch" objects that are not in ISAC's workspace. In the interests of brevity, the rest of the paper is devoted to a scenario based on ISAC.

The assumed environment is that the robots serve to aid and extend the abilities of an elderly or physically disabled person. ISAC, pictured in Figure 2, is activated and the benchmark demonstration commences. ISAC begins scanning its environment for utensils, food, phones, and other items relevant to its task. Once a certain level of certainty about this environment is achieved, ISAC begins scanning in front of itself for a user. Strange or badly classified items are noted for later use. If too much time passes, ISAC re-scans the environment and then comes back to looking for a user. When a user is detected (using skin-tone color, face detection, voice command and combinations of these) ISAC begins with a greeting using voice and gestures. ISAC may then ask the user for help (through voice, gestures and visual feedback) in identifying things that were not automatically classified in the initial pass and introduce the user to some options.

Once the user is in place and the robot has begun interaction, the user might ask ISAC to feed him soup using a voice command. ISAC should confirm this with the user using voice output and gesturing to the soup. If soup is not on the table, ISAC should set the table from a side cart. If there is no soup, ISAC should say so and list the food items it knows it actually has as alternatives for the user. Assuming that soup is requested and is available, then the demonstration should proceed.

ISAC should set the soup close to the user and pick up the spoon. The active vision system will then alternate between tracking the user and fixating the bowl to guide both parts of the feeding motion. ISAC will enter a cycle of dipping up soup (dip confirmation might use color or force information, for example) and bringing it to the user's mouth. A force transient will signal that the soup is taken and the cycle will continue. Perhaps the phone starts ringing when the robot has soup in the spoon. It should start taking the spoon back to the bowl and begin locating the phone. As soon as the phone is located, the robot should pick up the phone with its other hand while shaking off soup from the spoon with the dipping hand. When the conversation is over, signaled by force on the phone, ISAC should hang up and ask if it should resume feeding.

V. The Intelligent Machine Architecture

We will now discuss how our new architecture helps us develop complex demonstrations, such as the one above. The Intelligent Machine Architecture (IMA) is a new approach for the design of the control software for intelligent machines that are principally limited by difficulty in integrating existing algorithms, models, and subsystems. The IMA differs from traditional software systems in two aspects. First, the IMA uses a system level model that is agent-based as shown in Figure 3. Thus, each resource, task or domain element is modeled in software as an agent. Second, the IMA uses an agent level model that is component-object based as shown in Figure 5. This two level approach to the problem of complex software design for intelligent systems addresses both software

engineering issues such as reuse, extensibility, and management of complexity as well as system engineering issues like parallelism, scalability, reactivity, and robustness. The result of the approach is a system of concurrently executing software agents, formed from reusable component objects, that comprise the control software for an intelligent machine. The IMA is inspired by many modern robot architectures including the Subsumption Architecture [5], AuRA [15], CIRCA [16], and GLAIR [17] and represents the synthesis of many ideas from these systems.

An agent within IMA is assumed to have certain properties that separate it from the currently popular term "Agent" [18] and focus on what is essential to being an agent in the context of software system development. Wooldridge [19] stipulates that agents must be able to communicate in formal languages, but that is not essential to the agent concept in the software module context as pointed out by Baeijs and Marcerac[20, 21]. The properties of an agent assumed in IMA are:

•**Autonomy** - Agents own their components. They are separate and distinct concepts or objects in the domain for which the software is written. Thus, they have a strong conceptual encapsulation boundary at the system level.
•**Proactivity** - Agents act locally, based on their own internal state, resources and observations of other agents through specific relationships. The core of the agent's operation is a decision or action selection process.
•**Reactivity** - Agents are reactive because they respond to changes in their external environment.
•**Connectivity** - Agents are cooperative because they give and receive information from other agents to achieve their tasks.
•**Resource Bounded** - Agents are resource driven and may be competitive because they represent only a single conceptual element and depend on other agents as their resources. In a properly structured system, the competition between agents for a resource should represent a natural property of the system, not a bottleneck in the software architecture. In these cases, arbitration between various actions becomes important.

The design goals of the IMA arose from many years of experience developing integrated robot systems and are also influenced by the following issues. First, the agent-based decomposition of the robot system facilitates a more conceptually sound model for the software. Each agent is based around a single element of the domain-level system description and tightly encapsulates all aspects of that element. For example, Figure 4 shows that IMA agents are built to represent the physical resources of the robot (e.g., Arms, Pan-Tilt Head, Image Capture, Sound Capture, Speech I/O, Sonar, Lidar, etc.) as well as behaviors (e.g., Avoid Collisions, Coordinate Movements, Emergency Reflexes), skills (e.g., Visual Tracking, Grasping, Visual Servoing), and tasks (e.g., Feed Soup, Find Object, Assemble Parts). However, the model of the environment is also developed as a set of agents that share the robot's physical resources with skills and behaviors (e.g. Bowls, Blocks, Parts, Walls, Places, Forks, Collections) and engage in a process of anchoring [22] to keep their state coherent with the world as experienced by the robot's sensor resources. There is also a software agent that represents the user of the system. This user agent uses the resources of the robot to estimate the state of the user and represents what the system knows about the user.

An additional feature of this system design is that the agent-based decomposition facilitates parallel processing and can take advantage of both distributed and SMP computer systems. Finally, each agent acts locally based on internal state and provides a set of services to other agents through various relationships. The semantics of a relationship between agents is that one agent provides another agent with resources for action. This model of action and communication requires a different view of system design and is more loosely coupled than traditional software approaches. One justification for this tradeoff is the time value of information in robotic systems. Unless new information entering the software system can quickly become a resource for action, it rapidly becomes useless for the robot to retain, process or evaluate this information.

The agent model also relies on reusable sets of component-objects to facilitate the growth of the system over time. By building software using this model, agents in the system focus on modeling concepts and providing services instead of becoming the "algorithm in a box" that is typical of many modular software systems. The IMA uses algorithms, representations, and links between agents as component-objects within each agent. Instances of these component objects can be reused in many agents, in varied configurations or for completely different purposes at the same time. For example, suppose a new object classification algorithm is developed as a component object. Then, by editing a few decision rules in each agent, this new method for recognition can be used by many different elements of the system. No new "module" must be built and interfaced to the system to provide this algorithmic service. The integration of new algorithms happens at a much finer grained level that is conceptually appropriate in the context of the evolving robot control system. In Figure 5, one can see that agents are composed from a standard set of component objects including:

•**Agent Manager** - Provides the scaffolding on which other pieces of the agent are built, as well as some features like persistence. This element also provides a meta-level interface through which another agent can inspect its composition and perhaps alter it if necessary. This feature is currently intended to be used to support on-line interactive tools for agent construction and tuning, but could be used for adaptation and learning.
•**Agent Decision Engine** - Provides the action selection for the agent components, based on a set of productions or rules. The engine is a kind of production system, but is based on the latest advances in multi-valued logic for action selection from Saffiotti [22]. This engine is a replaceable component and can be upgraded or altered for specific agents within a system as it grows. The current engine can support a general production system, first-order logic, fuzzy-logic, and the implementation of a hierarchical state machine for agent action selection.
•**Agent Productions** - These components are a kind of generalized production rule for the agent that includes the evaluation of agent component state, as well as an invocation mechanism for performing computations, assignments, and state updates on other agent components. Actions can be simple or composite component objects. The rule-like structure of productions can be used to model the agent action selection process.
•**Agent Resources** - These elements of the agents are data repositories (called representations), algorithms, toolboxes, and links to the state of other agents (called relationships). These elements are evaluated, invoked and updated by the selection of actions within the agent. Components can be simple or composite component objects. Some components might provide a kind of scratchpad or internal workspace for the decision engine and other components, while other components are algorithmic resources that allow the agent to invoke highly efficient and optimized computations in a particular situation.
•**Agent Component Managers** - These elements are wrappers for the agent components that handle object persistence and provide visual interfaces to the components that are used by developers to build up the agent systems more easily.

Component managers also provide a place for knowledge about component properties to be stored. Thus, the component managers can be used to implement design rules to prevent a component from being misused within an agent. This is a kind of meta-data about the component.

VI. Architecture For Our Scenario

Figure 4 shows a proposed agent network that represents the physical and basic behavioral resources for our dual-armed humanoid robot, ISAC. This agent network shows the basic agents for arm control and active visual sensing. Built on the basic resource agents are agents that represent the behaviors and skills of the robot, including collision avoidance, stereo tracking, color tracking, grasping, and visual servoing. Another set of agents represents the elements of ISAC's environment including agents to represent utensils, bowls, food, telephones, users, and unknown objects. These agents use the basic resources and skills of the robot to engage in a process of anchoring [22] to update their values based on sensory inputs. Finally, there are task agents that use the resources of the robot to determine when to activate. Once a task agent is activated, it acts as a coordinator and has primary control over the resources and other agents in the system; however, it does not have total control. If the states of the resource agents change enough from the state region that activated the task, then another task agent may take over and gain control. The entire operation of the system is distributed, and even the planning of tasks and sub-task agents is coordinated through a distributed spreading activation relationship between task agents. The human user is also modeled as a software agent that encapsulates what the robot knows about the state of the user in one coherent scope. This software agent interacts with the robot resources, skills, and task agents to get things done for the user. What follows is a more detailed description of the kinds of agents that we are implementing for our service robotics benchmark demonstration.

The lowest level agents in our system represent the links and sensors on our 6-axis manipulators. Other agents can participate in this relationship and exert "pseudo-forces" on the arm agents to realize any type of motion control, including joint control, position control, velocity control, even task space impedance control similar to that achieved by Cameron [23]. This motion control network can handle physical mechanisms that have a tree structure such as multi-armed robots, mobile manipulators and robots with dexterous hands. This motion control approach is also applied to control the motions of ISAC's 4-axis pan-tilt-verge camera head.

Other low level agents represent the senses of the robot or interfaces to external software components that are not agents. These include both stereo monochrome and color monocular image agents. There is also a sound capture, and voice input/output agent, and the motion control agents which provide sensory information in the form of positions, velocities and forces.

Based on these resources, tracking agents are sensory-motor skills that use the image capture agents and exert "pseudo forces" on the motor control agents to track features with the pan-tilt-verge system. Subtraction-based tracking with a small low resolution image of the target is used. This approach provides us with sufficiently high accuracy for our tasks and is relatively insensitive to changing background imagery. One skill agent in development is the visual servoing agent based on the development of a novel visual servoing method by Bishay [25]. This agent uses tracking agents as resources, observes the pan-tilt-verge agent and contributes "pseudo-forces" to the manipulator agents to move the robot manipulators toward a tracked, fixated target point in visual space.

Environmental agents which are used as the internal model of things that interact with ISAC engage in a process called anchoring. Agents that engage in anchoring use the basic resources and skills that are part of the robot to compute or estimate their own internal states. They may also incorporate a library of methods for recognition and manipulation and a small knowledge base that describes ISAC's knowledge about the external component. Examples of environmental agents include the user, spoons, forks, bowls, cups, telephones, napkins and unknowns. The unknown agent is assigned to elements of the environment that ISAC has not currently identified successfully or has only partial information about. The user is integrated into the system by building a software agent that models the human user and interacts with the rest of the system using the same mechanisms. This captures the user as an agent within the system and makes explicit the connections between the user and other parts of the software.

The model for interaction between task agents in IMA is based on the notion of common currency exchange as described by McFarland [24]. This model for interaction makes it possible to accommodate a wide variety of interactions in the network. If the currency is like a one-dimensional energy function, then the interactions can model spreading activation which has been shown to support both behavior sequencing by Maes [6] and, with a different interpretation of currency, Bagchi used the same mechanism for planning in uncertain environments [8].

Typical development mixes decision processes (action selection) and the functional or computational aspects of operation. However, within the agent object model of IMA there is a clean separation between functional and representational components and the local decision process that guides them. Each component within an agent may be checked by an "evaluator" that assigns values used by the action selection mechanism. The action selection mechanism assigns activation levels to "activator's" that may in turn invoke computations, data transfers, updates, etc. on the "data-path" objects in the agent (see Figure 6). By extending the set of properties evaluated on these functional components to timing values, etc. it is possible for the agent to make choices between a variety of methods of computation, based on the current situation.

The action selection mechanism currently used in IMA agents is based on multi-valued logic or fuzzy-logic. This logic provides a smooth connection between the rule-based decision process that guides an agent's operation and the continuous dynamics of the system. The evaluators and activator objects provide the abstraction of input and output membership functions for the inference engine. The activator objects also provide object-specific defuzzification. In a sense the objects generalize the fuzzy inference process to any component that can define evaluator and activator objects to map values. For example a simple "state variable" representation component might provide the service of a scratch-pad for the agent. It's evaluator and activators would then correspond to numerical comparison and numerical assignment respectively. Thus, this action selection mechanism can be applied to give the agent a generalized fuzzy production system, capable of emulating a fuzzy expert system. The novelty in this use of such a mechanism comes from the narrowing of scope provided by the agent-based system decomposition. This reduces the problems faced in scaling the rules of the "mini-expert" because it only worries about it's own agent's perspective.

VII. Current Development Status

Our current state of development consists of converting

existing algorithms and old modules into our new agent-based system. We have implemented initial combinations of visual servoing, skin-tone tracking and arm motion control that show promise in building dynamic skills for the robot from a network of interacting software agents. Our initial agents also used a very limited form of component-based software development that greatly simplified development of modules for the system. Our new architecture is based on a more extensive and structured concept of what constitutes a software agent, and uses component-based software development at every level of agent development for improved code reuse and more rapid development. The descriptions in the paper have been deliberately kept simple to help bring out the philosophical distinctiveness of the IMA. Many features of the architecture have not been fully discussed, but are being included in our development process. For example, agents may participate in composite structures or groups of sub-agents that each handle part of the overall concept: A user agent might include a head/face sub-agent, a user body sub-agent and two hand sub-agents that attempt to anchor themselves to these features of the human user and use their knowledge about the constraints between their relationships to help make better decisions.

VIII. Conclusions

The importance of system integration has been discussed in the context of intelligent service robot development. Various architectures for building integrated intelligent systems were discussed and a novel architecture for integrated system development presented. The components of this architecture are nearing completion and will be applied to build highly integrated, human-interactive robots in our research lab. We plan to evaluate both the robot performance under this architecture as well as the usefulness of the architecture to the development process. As the scale and complexity of robot software increases, the successful construction of integrated robot systems depends less on the performance of any one particular algorithm and more on system architecture and decision processes in each element of the system. Our robot performance will be evaluated based on its interaction with a human user as well as reliability and speed of task completion. The architecture itself will be evaluated based on the efficiency of operation, ease of extension, code reuse and usefulness to developers of our robot system.

IX. References

[1] J.D. Crisman and G. Bekey, "Grand challenges for robotics and automation: The 1996 ICRA panel discussion," *IEEE Robotics and Automation Magazine*, 3(4):10-16, December 1996.

[2] K. Kawamura, D.M. Wilkes, T. Pack, M. Bishay, and J. Barile, "Humanoids: Future robots for home and factory," In *International Symposium on Humanoid Robots*, pp. 53-62. Waseda University, Tokyo, Japan, October 1996.

[3] G. Booch. *Object-Oriented Analysis and Design*. Addison-Wesley, 1994.

[4] P.E. Agre and D. Chapman, "What are plans for?," In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, vol. 6, pp. 17-24, 1990.

[5] R.A. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. Robotics and Automation*. RA-2(1), March 1986.

[6] P. Maes, "Behavior-based artificial intelligence," *Proc. 2nd Conf. on Adaptive Behavior*, MIT Press, 1993.

[7] R.C. Arkin, "Towards the unification of navigational planning and reactive control," In *AAAI Spring Symposium on Robot Navigation*, 1989.

[8] S. Bagchi, G. Biswas and K. Kawamura, "Interactive task planning under uncertainty and goal changes," *Robotics and Autonomous Systems*, 18:157-167, 1996.

[9] R.C. Luo and M.G. Kay, "Multisensor integration and fusion in intelligent systems," *IEEE Transactions on Systems, Man and Cybernetics*, 19(5), 1989.

[10] R.C. Arkin, "Just what is a robot architecture anyway? Turing equivalency versus organizing principles," In *AAAI Spring Symposium: Lessions Learned from Implemented Software Architectures for Physical Agents*, 1995.

[11] E. Rich and K. Knight. *Artificial Intelligence*. McGraw Hill, 1991.

[12] J.R. Anderson. *The Architecture of Cognition*. Harvard University Press, 1983.

[13] J. Laird and S. Newell, "An analysis of SOAR as an integrated architecture," In *SIGART Bulletin 2*, pp. 85-90, 1991.

[14] J.K. Rosenblatt, "Damn: A distributed architecture for mobile navigation," In *AAAI Spring Symposium: Lessons Learned from Implemented Software Architecture for Physical Agents*, 1995.

[15] R.C. Arkin, "Integrating behavioral, perceptual and world knowledge in reactive navigation," *Robotics and Autonomous Systems*, 6:105-122, 1990

[16] D.John Musliner, *CIRCA: Cooperative Intelligent Real-Time Control Architecture*. PhD thesis, University of Michigan, 1993.

[17] H.H. Hexmoor, J.M. Lammens, and S.C. Shapiro, "An autonomous agent architecture for integrating unconscious and conscious, reasoned behaviors," In *AAAI Spring Symposium: Lessons Learned from Implemented Architectures for Physical Agents*, 1995.

[18] M. Minsky. *The Society of Mind*. Simon and Schuster, 1986.

[19] M. Wooldridge. Time, knowledge and choice. Technical report. Department of Computing, Manchester Metropolitan University, United Kingdom, 1994.

[20] C. Baeijs, Y. Demazeau and L. Alvares, "Sigma: Application of multi-agent systems to cartographic generalization," In *Agents Breaking Away: 7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pp. 163-176, 1996.

[21] P. Marcenac, "The multiagent approach," *IEEE Potentials*, 16(1):19-22, 1997.

[22] A. Saffiotti, K. Konolige, and E.H. Ruspini, "A Multivalued Logic Approach to Integrating Planning and Control," *Artificial Intelligence*, 76: 481-526, 1995.

[23] J. Cameron, D. MacKenzie, K. Ward, R.C. Arkin, and W. Book, "Reactive control for mobile manipulation," *Proc. IEEE Conference on Robotics and Automation*, 1993.

[24] D. McFarland and T. Bosser. *Intelligent Behavior in Animals and Robots*. The MIT Press, 1993.

[25] M. Bishay, A. Peters II, D.M. Wilkes, Fixation Point Servoing, Center For Intelligent Systems, Vanderbilt University, 1997, CIS Technical Report 97-01.

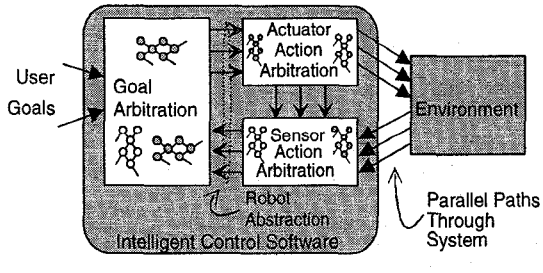

Figure 1 - Fundamental Decision Process

Figure 2 - ISAC System and User

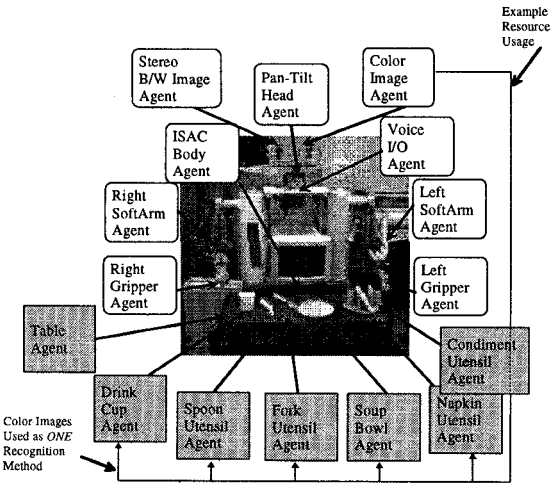

Figure 3 - Correspondence Between Agents and Physical Objects

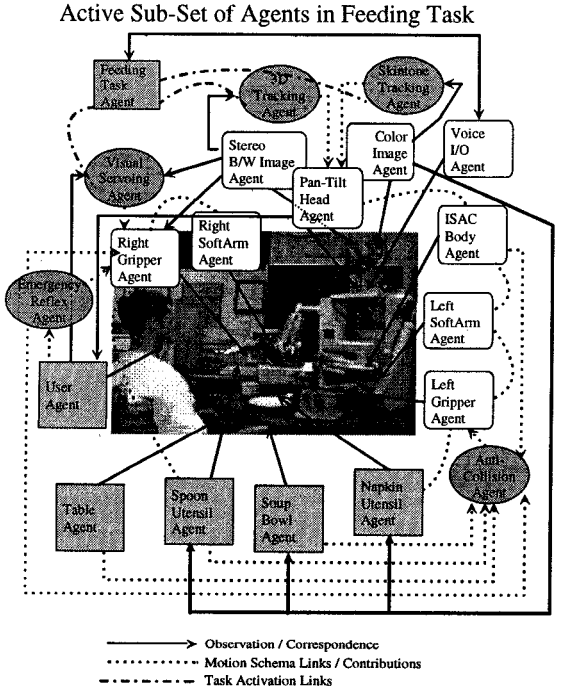

Figure 4 - Agents Active in a Task

Interactions of Objects Within an Agent

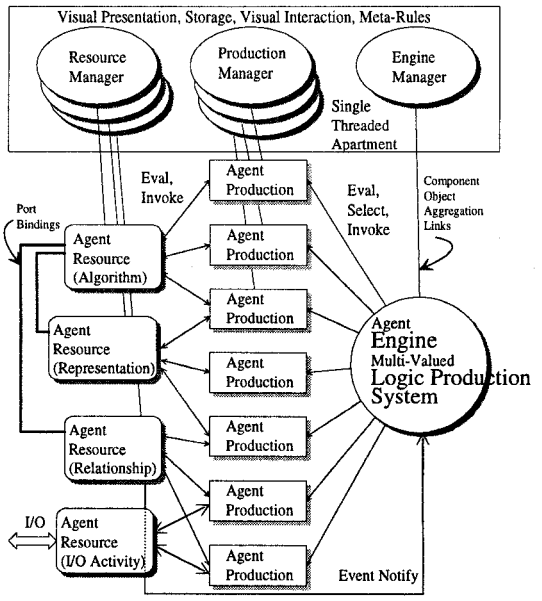

Figure 5 - IMA Agent Components

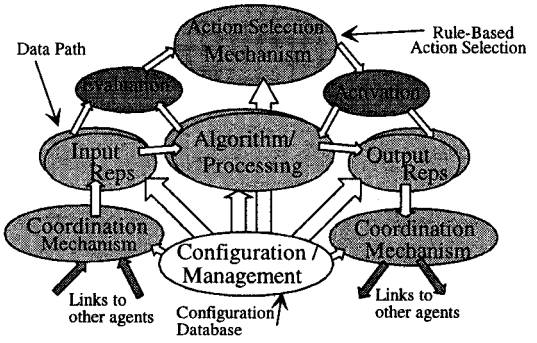

Figure 6 - A Functional Description of IMA Agent Operation