Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2007)

A Review of Intelligent Systems Software for Autonomous Vehicles

Lyle N. Long[*], Scott D. Hanford, Oranuj Janrathitikarn, Greg L. Sinsley, and Jodi A. Miller
The Pennsylvania State University

[*] Email: LNL@psu.edu, Telephone: (814) 865-1172

Abstract—**The need for intelligent unmanned vehicles has been steadily increasing. These vehicles could be air-, ground-, space-, or sea-based. This paper will review some of the most common software systems and methods that could be used for controlling such vehicles. Early attempts at mobile robots were confined to simple laboratory environments. For vehicles to operate in real-world noisy and uncertain environments, they need to include numerous sensors and they need to include both reactive and deliberative features. The most effective software systems have been hierarchical or multi-layered. Many of these systems mimic biological systems. This paper reviews several software approaches for autonomous vehicles. While there are similarities, there are differences as well. Most of these software systems are very difficult to use, and few of them have the ability to learn. Autonomous vehicles promise remarkable capabilities for both civilian and military applications, but much work remains to develop intelligent systems software which can be used for a wide range of applications. In particular there is a need for reliable open-source software that can be used on inexpensive autonomous vehicles.**

Index Terms—**Mobile robots, autonomous vehicles, intelligent agents, software, and artificial intelligence.**

INTRODUCTION

Mobile robots, or autonomous vehicles, are becoming widely used in the military and civilian sectors. These include unmanned air vehicles (UAV), unmanned ground vehicles (UGV), unmanned spacecraft, and unmanned underwater vehicles (UUV). Most of the existing systems are only semi-autonomous, and rely on regular human intervention. To go beyond this capability will require sophisticated, yet flexible, software systems. While there are many existing software packages that could be used for mobile robots, they all have their advantages and disadvantages. This paper attempts to describe some of the more common software packages. It is not possible to describe all such packages, so we have chosen some of the most common systems for review here.

Intelligent systems for mobile robots are still in their infancy. There is no perfect system yet. The ultimate system may well be a hybrid system, which uses combinations of the systems described below (or those not mentioned here). Future systems will most likely rely on combinations of computational intelligence methods, such as traditional control, fuzzy logic, neural networks, genetic algorithms, rule-based methods, or symbolic artificial intelligence.

In building intelligent systems software for mobile robots or unmanned vehicles, it will be valuable to consider biological systems, especially humans. An intelligent system will need to incorporate capabilities such as sensing, reasoning, action, learning, and collaboration.

It is useful to consider the human brain as the ultimate intelligent controller. It is an astounding device and is still not well understood. It has an estimated 10^{11} neurons (10^{14} bytes of memory) and can process at roughly 10^{16} operations/second [1]. Large parallel supercomputers are approaching the speed and power of the human brain [1]. But just as fascinating are the human sensor systems (touch, hearing, sight, smell, and taste). Replicating this vast array of sensors will be just as daunting as replicating the human brain. An intelligent system for mobile robots will need to efficiently handle the wide variety of possible sensor systems also, and perform data fusion. It will also need to emulate (to some extent) the motor control functions of humans (actions), which means the software will need a mechanism for the output of information to motors and servos.

There are several good references on control approaches for robotics [2-5]. The most promising approaches use layered or hierarchical control strategies, for example: subsumption [2], behavior-based [3], reference model [4], or three-layer [5]. Traditional artificial intelligence (AI) approaches [6] have been of limited value in developing autonomous robots, however fuzzy logic, neural networks, genetic algorithms, and symbolic processing may be useful as components within the autonomous systems software. In addition, cognitive architectures may also be very useful [7].

The systems described below all vary in their ability to incorporate sensing, reasoning, action, learning, and collaboration. Machine learning [8] is probably the most difficult part of the problem. Few of the systems below (except for the cognitive architectures and the neural networks) have the ability to learn.

Collaboration is also a crucial feature of these systems. They will be most interesting and useful when there are many of them networked together. Getting one system to intelligently operate in the real world is extremely challenging, getting several of them to work together is even more difficult.

Designing and building efficient, reliable mobile robots (or autonomous vehicles) will be extremely difficult and will require many more years of research and development. But even if we can accomplish this, there will be much more to do. Questions about when the systems might be "self aware" or conscious [9, 10] will remain for a very long time.

Another issue that will need to be addressed before autonomous vehicles are commonplace is software reliability and safety. Most of the software described herein is research software, and is not suitable for mission- or safety-critical systems. To build reliable software systems, one must follow good software engineering practices [11]. Unfortunately, too few scientists and engineers are being trained in software engineering [12].

I. UNMANNED VEHICLES

Unmanned vehicles are being used on the ground, in the air, in space, and in water (surface or underwater). Each of these offers special challenges and difficulties.

Ground vehicles are probably the least challenging, since they are usually in equilibrium. If there is a failure, the vehicle is not usually damaged. Of course, driving through complex terrain or at high speeds is still very difficult, as the DARPA Grand Challenge has demonstrated [13]. Ground vehicles also include interplanetary rovers, such as those developed at NASA's Jet Propulsion Laboratory [14]. These offer additional difficulties due to the long delays in communicating with them, and the inability to correct many hardware or software failures because they cannot be recalled. They also have to survive in the extreme conditions of space.

Water-based vehicles could be surface or underwater (UUV) types. Modern torpedoes are essentially intelligent autonomous vehicles. UUVs have communication challenges, and often must surface to communicate with other systems.

Autonomous aircraft are extremely challenging, because they usually need autopilots onboard just to maintain straight and level flight. The intelligent systems software must then interact with the autopilot. These vehicles can be fixed wing or rotary wing (e.g. helicopters, ducted fans …) [15]. Fixed wing aircraft will typically have higher forward speeds, larger range, and greater payload capabilities. Rotary wing vehicles will offer the capability to hover, and take off and land in smaller areas.

II. SOFTWARE SYSTEMS

In this section we will give brief descriptions of some existing software systems for autonomous vehicles. There are many other software systems, including proprietary systems which have been developed in industry and government laboratories.

A. *Java Expert System Shell (JESS)*

The Java Expert System Shell (Jess) is a rule-based engine written in Java language by Ernest Friedman-Hill at Sandia National Laboratories (Livermore, CA). Jess is available for download (http://www.jessrules.com). It is free for academic use but it requires a license for commercial use. Jess uses a forward-chaining Rete algorithm to process rules. In the most recent version of Jess, it also supports backward chaining. The Jess rules can be specified in two formats: the Jess rule language or XML. The Jess rule language is similar to LISP and it is recommended over XML. During operation, Jess rules make changes to the working memory which is a collection of knowledge. Jess can be run as a standalone program by using an interactive command-line interface. In addition, it can be imported as a Jess library in Java programs. Jess can be developed in any text editor but there is also a development environment available, JessDE, which is based on the Eclipse platform [16]. An interface to hardware can be programmed within Java programs.

Jess has been implemented in a hardware system under the concept of a network-based service robot called URC (Ubiquitous Robotic Companion) supported by the Korean government. The mobile robot can sense the external sensors in the network and can also access the remote computer using the CAMUS (Context-Aware Middleware for URC System). Jess is applied as an inference engine in the task manager of CAMUS [17]. In addition to a network-based service robot, Jess has also been used in a humanoid intelligence architecture - PKD (Phillip K. Dick). The PKD android won first place in the 2005 AAAI Open Interaction competition. The android can socialize with people by detecting faces and facial expressions and recognizing speech. The inputs are sent to a control system composed of Jess and Natural Language Generation (NLG) Functions. The output is sent to servo motors and the sound/speech units [18].

B. *Fuzzy Logic in Integrated Reasoning*

The Integrated Reasoning Group of the Institute for Information Technology of the National Research Council (NRC) of Canada has developed two fuzzy logic software packages: "FuzzyCLIPS" and "FuzzyJ." Both of them are freely available (http://www.iit.nrc.ca/IR_public/fuzzy) for educational use and a commercial license can be obtained [19].

FuzzyCLIPS is an extension of the CLIPS (C Language Integrated Production System), which is an expert system shell developed by NASA. It integrates a fuzzy reasoning engine to CLIPS facts [19]. FuzzyCLIPS is implemented in Isaac, a rule-based visual language for geometric reasoning designed for mobile robots, which is under development at New Mexico State University [20, 21]. The Isaac language has been tested by simulating a prototype inference engine to perform standard robotic tasks. However, the integration of Isaac with a robot model has not been accomplished yet [21].

The FuzzyJ Toolkit is a Java API (Application Program Interface) for handling fuzzy logic systems. It can be used as a standalone system or it can be integrated with Jess (FuzzyJess). FuzzyJess provides similar capabilities but it is more flexible than FuzzyCLIPS [19]. The input and output can be handled by using Java APIs. FuzzyJ Toolkit has been implemented in a mobile robot that used a Java-based control program for navigating corridors. The robot is a test-bed for development of an intelligent wheelchair. It is equipped with a camera and infrared sensors. The sensor inputs are processed by using fuzzy logic to avoid collisions [22].

C. *Subsumption Architecture*

Brooks developed the idea of subsumption architectures while attempting to build intelligent robots that could be tested in real environments at each stage of development [23, 24]. A revolutionary idea of this architecture was the linking of perception and action without an internal representation of the environment in which the robot was situated. A subsumption architecture consists of control layers, each of which represents a behavior. The bottom layer of the architecture represents a simple behavior and each layer added on top of the bottom layer gives a robot the capability to perform another, more complicated behavior. An important concept is that while adding a control layer increases the number of robot behaviors, the robot can still function in a real-world environment if the new control layer is removed from the system. An early robot using the subsumption architecture had a bottom layer that allowed the robot to avoid obstacles in a dynamic environment [23]. As stated above, the robot could function with only this control layer. The next layer added to the system permitted the robot to perform a wandering behavior. With these two layers, the robot wanders until it senses it is close to an obstacle, at which point the bottom control layer inhibits the wandering behavior until there is no longer an obstacle present.

Each control layer consists of modules that are implemented as augmented finite state machines [25]. The modules have input and output lines to permit low bandwidth communication with other modules. Modules that receive sensor information and send commands to motors have special mechanisms to accomplish this communication. All of the augmented finite state machines, which can be implemented in Lisp, run in parallel to achieve the behaviors specified by the control layers [23]. A robot controlled by the subsumption architecture has demonstrated the ability to build a map of its environment without using traditional data structures [26]. There have been many fascinating applications for robots using the subsumption architecture, including control of six-legged walking robots [24], control of small rovers on Mars [27], and the development of sociable robots [28].

D. *Autonomous Robotic Architecture (AuRA)*

The Autonomous Robotic Architecture (AuRA) [29-31] is a hybrid architecture that combines deliberative and reactive components to use the advantages of both symbolic reasoning and reactive control. The hierarchical deliberative components include a mission planner, spatial reasoner, and a plan sequencer. The reactive component is a schema controller that is responsible for controlling the robot behaviors at run-time using motor and perceptual schemas. The motor schemas define primitive robot behaviors such as avoiding static obstacles and moving ahead while the perceptual schemas have used computer vision to detect colored blobs and ultrasound to sense obstacles to avoid (and to recognize objects). The MissionLab simulator, which can be used to generate schema-based control systems, can be found at http://www.cc.gatech.edu/ai/robot-lab/ .

In AuRA, deliberation only occurs during execution if a problem with the plan created by AuRA's deliberative system occurs. If a problem does occur, the hierarchical planner is used to fix the problem, starting with the bottom level plan sequencer and moving up to the spatial reasoner and then to the mission planner only if the problem has not been resolved using a lower level component. The architecture is modular, as the mission planner, spatial reasoner, plan sequencer, and schemas can all be changed without affecting the other components. Several adaptation and learning methods have been implemented in AuRA including on-line adaptation of motor behaviors, case-based learning, and genetic algorithms.

Parts of the AuRA architecture have been used in several different areas, including manufacturing, multi-robot teams, and indoor and outdoor tasks. The architecture was used to control three trash collecting robots that won the 1994 Robot Competition sponsored by the AAAI. The higher levels of AuRA's deliberative reasoning capabilities were not used for this competition. Instead, a finite state acceptor was programmed as a plan for trash collecting and was interpreted by the plan sequencer at run-time. The schema controller used information from the plan sequencer to activate the perceptual and motor schemas required to achieve the desired robot behaviors. For example, in order to pick up a piece of trash, a robot's active schemas could include a perceptual schema to detect a red blob (e.g., a soda can) and a motor schema to move to a goal position (e.g, the position of the soda can). These trash collecting robots were able to work together without communicating by using schemas that resulted in the robots not coming too close to each other, allowing the team to efficiently search for trash.

E. *ARL/PSU Intelligent Controller*

The Applied Research Laboratory at the Pennsylvania State University (ARL/PSU) has developed an Intelligent Controller (IC) architecture. This is a behavior-based architecture for the control of autonomous devices [32]. The Intelligent Controller architecture was initially based on the subsumption approach, but actual system needs presented additional requirements. This resulted in a modified approach to an intelligent control architecture [33, 34]. The IC architecture has also been extended for collaborative capabilities, resulting in a unique approach to coordinated control.

The ARL/PSU IC architecture has two main components: Perception and Response. The role of the Perception Module is to create an internal representation of the external world relevant to the IC and recognize features of interest from incoming data, such as sensor inputs. Data fusion algorithms are responsible for fusing new sensor data with existing sensor data so that a current situational awareness is maintained. A fuzzy-logic based classifier is used to determine to what degree an object displays certain properties of interest (i.e.

friend, foe, obstacle, etc) [35]. The role of the Response Module is to create a plan of action to perform a specific mission using the situational awareness created by the Perception Module. It accomplishes this by a three level hierarchy (Mission Manger, Behaviors, and Actions). The Mission Manager arbitrates between Behaviors requesting control. Behaviors are autonomous agents capable of generating, executing, monitoring and adapting plans to achieve a certain task. Actions are responsible for generating output commands to low level vehicle and sensor controllers, and other ICs (or humans) in the mission team.

Currently, both object-oriented C++ and Ada implementations of the architecture exist. The IC software is capable of interfacing with any sensor or actuator. Sensor data is typically processed before it is sent to the IC (e.g. image processing, bandpass filtering, discrete Fourier transforms, etc.). An actuator must have a low level controller that receives setpoints and controls the actuator appropriately.

Key features of the ARL/PSU IC architecture include:

- A robust ability to react to unforeseen situations (no scripted cases)
- Autonomous, on-the-fly dynamic planning and re-planning
- Situational awareness, with the ability to make inferences and recognize the existence of properties from potentially incomplete and erroneous input data through the use of Continuous Inference Networks (CINets) [35]
- A common architecture for multi-vehicle collaboration
- Expandable to incorporate new capabilities as they are identified
- Human interaction at any desired level, but not required

The ARL/PSU IC architecture has been applied to the control of multiple autonomous unmanned aerial vehicles (UAVs) [36 - 38] and several unmanned underwater vehicles (UUV). The UUVs have been demonstrated and are at a fairly high technology readiness level. Much of this work has not been published, but there is some discussion of it in references [39 - 42]. Using the IC architecture for UAVs is a more recent application. These aircraft [36-38] are equipped with commercial autopilots (from CloudCap Technologies) that handle sensor signal processing and low level control. The UAVs running the Intelligent Controller software are capable of autonomously performing several missions both independently and collaboratively.

F. Jet Propulsion Labs ASE

The Applied Sciencecraft Experiment (ASE) is a software package designed to facilitate autonomous science. It is capable of dynamically planning and executing missions related to the capture and downlink of scientific data. The software is organized as a three level architecture [43 - 46]. The top level is known as the Continuous Activity Scheduling Planning Execution and Response (CASPER) system. This system controls long term planning. It is responsible for scheduling activities that will facilitate the collection and downlink of the greatest amount of scientific data, while obeying resource constraints (battery power, instrument constraints, processing power, available downlink bandwidth, etc.). It utilizes dynamic planning, which means that the mission plan is continuously updated as new data or goals (from users on the ground) are received. The middle layer in the architecture is known as the Spacecraft Command Language (SCL) layer. The role of this middle layer is to translate high level activities sent to it by the CASPER system to a detailed sequence of commands that must be executed in order to complete the activity. Commands from the ground in the form of SCL instructions can also be uploaded at this layer. At the lowest level is the flight software. This flight software provides low level control of the vehicle hardware.

The most interesting application of this software has been onboard the Earth Observing One (EO-1) spacecraft [43 - 45]. The role of this spacecraft is to detect and record items of scientific significance on Earth. It has been used to detect events such as volcanic eruptions, ice breakup, and flooding. The ASE software allows the spacecraft to detect significant items, and plan how to gather more data about them the next time the spacecraft passes over the point of interest. The software also makes use of limited bandwidth by only sending scientifically significant data to Earth, and discarding other data. Flying onboard a spacecraft poses some unique challenges for the software. The ASE software had to be designed to be extremely efficient because the embedded processors onboard a spacecraft are typically much less powerful than a typical PC. Also because of the cost of a spacecraft, reliability is of critical importance. The ASE software therefore provides for constraint checking and fault detection at each of its three levels. A number of tests have been performed onboard EO-1 to validate the ASE software. The software has completed these tests with a great deal of success.

G. Cognitive Architectures

Cognitive architectures have been available for years [7]. These define the fixed processes that their designers believe are important for intelligent behavior, and have been primarily used to understand and simulate human cognitive processes. But they also have interesting potential for intelligent systems software, including mobile robots. Soar and ACT-R are two of the most widely known architectures. Both architectures originally focused on areas of cognition such as problem solving and learning and did not include mechanisms for perception and action, but have since been used to interact with external environments. In particular, SOAR is an extremely interesting software package that includes all the features important in a software system (e.g. sensing, reasoning, action, and learning).

1) ACT-R

ACT-R is implemented in Lisp and can be downloaded at the ACT-R website (http://act-r.psy.cmu.edu/). ACT-R's current architecture consists of goal, declarative memory, perception, and motor modules that are hypothesized to represent processes in specific brain regions [47]. A central production system uses production rules and a subset of information from each module to select a best production to fire for each reasoning cycle. Sub-symbolic processes, such as activation of declarative memory elements, base-level learning, and calculating the utility of each production rule, also have an important role in the ACT-R architecture. In addition to this sub-symbolic learning, ACT-R is also capable of learning new knowledge "chunks" as well as links between chunks and creating new productions from combinations of existing productions. Input and output in ACT-R is performed using the visual, motor, speech, and audition modules in ACT-R/PM (http://chil.rice.edu/projects/RPM/index.html). The perception and motor modules in ACT-R are based on the sensory and motor mechanisms in EPIC [48].

ACT-R is probably best known for its use in the development in many cognitive models of human performance and for an algebra tutoring system [47]. Recently, ACT-R models have also been used to control robots. Bugajska et al. [49] developed a hybrid controller with reactive aspects for low-level calculations and cognitive aspects using ACT-R for high level processes such as reasoning. This group has also developed a robot that interacts with a child and uses an ACT-R model to study how children learn to play hide and seek [50]. Another group intends to use an ACT-R model controlling an anthropomorphic robot to learn how to collaborate with a human tutor to put together a jigsaw puzzle [51].

2) *SOAR*

The Soar architecture [7, 52], which was originally written in Lisp and has since been rewritten using C, can be downloaded at the Soar webpage (http://sitemaker.umich.edu/soar/home). In a Soar model, domain knowledge is encoded as production rules and information about the current state is stored in working memory in attribute-value form. If a Soar agent does not have enough knowledge to select the best operator for each reasoning cycle, an impasse occurs in which Soar automatically creates a subgoal to determine the best operator. The learning mechanism in Soar (chunking) stores the results of the problem solving that is used to achieve the subgoal as a new production. Although the Soar architecture originally focused on internal problem solving and execution, it's planning, execution, and learning processes have also been used to interact with dynamic real world environments. Laird and Rosenbloom [53] discussed the characteristics of Soar that make it suitable for use in these environments and described two robotic systems (a robotic arm and a mobile robot that used sonar) that Soar was used to control. Two of these characteristics that make Soar well suited to work in such environments are the creation of impasses and automatic subgoals that support the hierarchical execution of complex goals (abstract operators) and a mix between deliberate selection of an operator (when sufficient knowledge is available to choose one) and on demand planning that can occur when a lack of knowledge results in an impasse. In addition, the Soar Markup Language (SML) has recently been developed to allow simple interfaces between the input and output links in Soar's working memory and sensors and motors.

Past research projects have used Soar agents to control unmanned vehicles in simulation environments. TacAir-Soar [54] demonstrated the ability of Soar agents to operate in a complex real-time simulation environment. TacAir-Soar agents used 5200 production rules, 450 total operators, and 130 abstract operators to fly U.S. military fixed-wing aircraft in a dynamic nondeterministic simulation environment that was accessible only through simulated sensors. Included in the flights performed by these agents were collaborative missions that required interaction between agents through simulated radio systems.

H. *NIST RCS*

Reference model architectures, such as the NIST RCS (Real-time Control System), have been successfully used in intelligent systems [55 - 57]. RCS is available for download at ftp://ftp.isd.mel.nist.gov/pub/rcslib. The RCS library (http://www.isd.mel.nist.gov/projects/rcslib/index.html) contains an archive of Java, C++, and Ada code and includes development tools, such as a Java based graphical design tool for automatic code generation. RCS has been tested on many platforms, including the Windows and Linux operating systems using Borland C++, Microsoft C++, and GNU g++ compilers [58].

RCS focuses on intelligently controlling real machines in real world environments. This architecture is organized into a hierarchy of nodes determined by the decomposition of a system's mission into increasingly simple tasks. Each node has a combination of cognitive and reactive mechanisms, has a knowledge base, and is capable of sensory processing, world modeling, value judgment, and behavior generation. The nodes at the lowest level of the hierarchy interact with sensors using their sensory processing modules and with actuators using their behavior generation modules. RCS has been integrated with a 4-D machine vision approach [59] and the architecture's knowledge base now includes visual images and maps in addition to symbolic data structures. RCS has been used in many areas, including manufacturing and unmanned vehicle applications. Recent work using RCS has focused on autonomous driving ground vehicles in real world environments and autonomous tactical behaviors for teams of unmanned military vehicles. Additionally, implementation of learning algorithms into 4D/RCS modules for the LAGR (Learning applied to ground robots) program has been a recent area of interest [60].

I. *Other*

While we have tried to discuss some of the most well-known approaches to software and algorithms useful for autonomous vehicles, there are many other approaches that we did not have space to discuss. In particular, there is the evolutionary robotics approach [61] which uses a combination of neural networks and genetic algorithms. Draper Labs also has an agent architecture called ADEPT [62], which represents years of research in autonomous systems. There is also ICARUS [63] and COGNET [64]. Another interesting technology is the PROLOG computer language [65].

III. Summary

Short descriptions of several intelligent software systems have been presented here. Figure 1 summarizes the key features and approaches included in these systems. The IEEE typically assumes that "computational intelligence" encompasses neural networks, genetic algorithms and fuzzy logic; but this definition is too restrictive. While many of these systems use fuzzy logic, none of them use neural networks or genetic algorithms. And rule-based or symbolic AI is often used. Many of the systems are reactive/deliberative; which seems essential for complex robots operating in uncertain environments. While early systems used Lisp most modern packages use either C/C++ or Java. Unfortunately, most of the packages have little or no ability to learn.

IV. Conclusion

This has been a brief review of several software systems that might be used to control autonomous mobile robots. It is important to compare these systems and their algorithms, in order to develop better autonomous vehicles. At the present time it is extremely difficult to compare these systems since they have not been applied to the same problem or vehicle, and most of them are very difficult to use as well. It would be very valuable to have some well defined test cases which could use these methods and compare results. There is also a need for more open-source software that is reliable and easier to use.

References

[1] Long, L. N. and Gupta, A., "Scalable Massively Parallel Artificial Neural Networks," AIAA Paper No. 2005-7168, AIAA InfoTech@Aerospace Conference, Wash., D.C., Sept., 2005.

[2] Brooks, R. A., *Cambrian Intelligence: The Early History of the New AI*, MIT Press, Cambridge, 1999.

[3] Arkin, R.C., *Behavior-Based Robotics*, MIT Press, Cambridge, 1998.

[4] Meystel, A.M. and Albus, J., *Intelligent Systems: Architecture, Design*, Control, Wiley, New York 2001.

[5] Bekey, G.A., *Autonomous Robots: From Biological Inspiration to Implementation and Control*, MIT Press, Cambridge 2005.

[6] Russell, S.J. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey, 1995.

[7] Newell, A., *Unified Theories of Cognition*. Harvard Univ. Press, Cambridge, 1990.

[8] Mitchell, T.M., *Machine Learning*, McGraw-Hill, 1997.

	Sub-Sumption	AuRA	SOAR	ACT-R	Fuzzy CLIPS	JESS	NIST RCS	PSU/ARL IC	FuzzyJ Toolkit	NASA ASE
Neural Network		✓?								
Genetic Algorithm		✓?								
Fuzzy Logic	✓				✓			✓	✓	
Symbolic AI		✓	✓	✓		✓				
Learning			✓	✓		✓				✓
Language	Lisp & Other	Lisp & Other	C & C++	Lisp	C	Java	C++	C++ & Ada	Java	C & Other
Reactive (R) or Deliberative (D)	R&D	R&D	D	D	D	D	R&D	R&D	D	R&D
Easy Sensor Input	✓	✓	✓		✓	✓	✓	✓	✓	
Freely Available			✓	✓	✓	✓	✓		✓	
Collab-oration			✓			✓	✓	✓		
Approx. Year	1986	1987	1987	1987	1994	1995	1997	1997	2001	2002

Figure 1. Summary of features and methods used in software systems.

[9] Dennett, D.C., *Consciousness Explained*, Back Bay Books, Boston, 1991.
[10] LeDoux, J., *Synaptic Self: How our Brains Become Who We Are,* Penguin, NY, 2002.
[11] Sommerville, I., *Software Engineering*, 8th Ed., Addison-Wesley, 2006.
[12] Long, L. N., "The Critical Need for Software Engineering Education for Aerospace Systems," to appear in CrossTalk: The Journal of Defense Software Engineering, 2007.
[13] http://www.darpa.mil/grandchallenge/
[14] http://www.jpl.nasa.gov/
[15] http://www.fas.org/irp/program/collect/uav_roadmap2005.pdf
[16] http://www.jessrules.com
[17] Kim, H., Y. J. Cho, and S. R. Oh, "CAMUS: A middleware supporting context-aware services for network-based robots," IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO2005), 2005. pp. 237-242.
[18] Hanson, D., A. Olney, I. A. Pereira, and M. Zielke, "Upending the Uncanny Valley," AAAI Conference Proceeding, 2005.
[19] http://www.iit.nrc.ca/IR_public/fuzzy/
[20] Pfeiffer, J.J., Jr., "A Language for Geometric Reasoning in Mobile Robots," Proceedings of the 1999 IEEE Symposium on Visual Languages, 1999. pp. 164-169.
[21] Pfeiffer, J.J., Jr., "A Prototype Inference Engine for Rule-Based Geometric Reasoning," Diagrammatic Representation and Inference, Springer: Germany, 2004. pp. 216-226.
[22] Ono, Y., H. Uchiyama, and W. Potter, "A Mobile Robot for Corridor Navigation: A Multi-agent approach," Proceedings of the 42nd annual Southeast Regional Conference, ACM Press, 2004. pp. 379-384.
[23] Brooks, R. A. "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, March 1986, pp. 14–23.
[24] Brooks, R. A., "Intelligence Without Representation", *Artificial Intelligence Journal* (47), 1991, pp. 139–159.
[25] Brooks, R. A. "A Robot that Walks; Emergent Behavior from a Carefully Evolved Network", Neural Computation, 1:2, Summer 1989, pp. 253–262. Also in IEEE International Conference on Robotics and Automation, Scottsdale, AZ, May 1989, pp. 292–296.
[26] Mataric, M. J. and R. A. Brooks, "Learning a Distributed Map Representation Based on Navigation Behaviors", Japan–USA Symposium on Flexible Automation, Kyoto, Japan, July 1990, pp. 499–506.
[27] Matijevic, M., "Autonomous Navigation and the Sojourner Microrover," *Science*, 17 April 1998: Vol. 280. no. 5362, pp. 454 – 455.
[28] Breazeal, C., *Designing Sociable Robots*, The MIT Press, 2002.
[29] Arkin R.C. and Balch T., "AuRA: Principles and Practice in Review," *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 9, No. 2-3, Apr. 1997, pp. 175 – 189.
[30] Mackenzie, D., Arkin, R.C., and Cameron, J., "Multiagent MissionSpecification and Execution," *Autonomous Robots*, Vol. 4, No. 1, 1997, pp. 29-57.
[31] Likhachev, M., Kaess, M., Kira, Z. and Arkin, R.C., "Spatio-Temporal Case-Based Reasoning for Efficient Reactive Robot Navigation" Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, 2005.
[32] Stover, J.A. and Kumar, Ratnesh, "A Behavior-based Architecture for the Design of Intelligent Controllers for Autonomous Systems," IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, Cambridge, MA, pp. 308-313. Sept. 15-17, 1999.
[33] Stover, J.A. and Gibson, R.E., "Modeling Confusion for Autonomous Systems," SPIE, Science Artificial Neural Networks, 1710, pp. 547-555, 1992.
[34] Stover, J.A. and Gibson, R.E. "Controller for Autonomous Device", US Patent #5,642,467, June 1997.
[35] Stover, J.A., Hall, D.L., and Gibson, R.E., "A Fuzzy-Logic Architecture for Autonomous Multisensor Data Fusion," IEEE Transactions on Industrial Electronics, 43, pp. 403-410, 1996.
[36] Weiss, L., "Intelligent Collaborative Control for UAVs," AIAA Paper No. 2005-6900, AIAA InfoTech@Aerospace Conference, Washington D.C., Sept. 26-29, 2005.
[37] Miller, J., A. F. Niessner. Jr., A. M. DeLullo, P. D. Minear, and L. N. Long, "Intelligent Unmanned Air Vehicle Flight Systems," AIAA Paper No. 2005-7081, presented at the AIAA InfoTech@Aerospace Conference, Washington D.C., Sept. 26-29, 2005.
[38] Sinsley, G.L. Miller, J.A., Long, L.N. Brian R. Geiger, Albert F. Niessner, Jr., and Joseph F. Horn "An Intelligent Controller for Collaborative Unmanned Air Vehicles," to be presented at the IEEE Symposium Series in Computational Intelligence, Honolulu, Hawaii, April 1-5, 2007.
[39] R. Kumar and J.A. Stover, "A Behavior-Based Intelligent Control Architecture with Application to Coordination of Multiple Underwater Vehicles," IEEE Transactions on Systems, Man, and Cybernetics, 30, pp. 767-784, 2000.
[40] "An Assessment of Undersea Weapons Science and Technology," National Academy Press, 2000.
[41] "Autonomous Vehicles in Support of Naval Operations," National Academy Press, 2005.
[42] Brown, N. M., "The World of Machine Intelligence," Research/Penn State, Vol. 17, no. 2, June, 1996. (http://www.rps.psu.edu/jun96/machine.html)
[43] Chien, S., Sherwood, R., Rabideau, G., Castano, R., Davies, A., Burl, M., Knight, R., Stough, T., Roden, J., Zetocha, P., Wainwright, R., Klupar, P., Van Gaasbech, J., Cappelaere, P., and Oswald, D., "The Techsat-21 Autonomous Space Science Agent," Proceedings of the Autonomous Agents and Multi-Agent Systems Conference, July 2002.
[44] Chien, S., Sherwood, R., Tran, D., Castano, R., Cichy, B., Davies, A., Rabideau, G., Tang, N., Burl, M., Mandl, D., Frye, S., Hengemihle, J., Agostino, J., Bote, R., Trout, B., Shulman, S., Ungar, S., Van Gaasbeck, J., Boyer, D.,

Griffin, M., Burke, H., Greeley, R., Doggett, T., Williams, K., Baker, V., and Dohm, J., "Autonomous Science on the EO-1 Mission," International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2003), May 2003.

[45] Chien, S., Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davis, Dan Mandl, Stuart Frye, Bruce Trout, Seth Shulman, and Darrell Boyer, "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," Journal of Aerospace Computing, Information, and Computing, Vol. 2, April 2005

[46] http://ase.jpl.nasa.gov/

[47] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An Integrated Theory of the Mind," Psychological Review 2004, 11(4), pp. 1026-1060.

[48] D.E. Kieras and D.E. Meyer, "An Overview of the EPIC Architecture for Cognition and Performance With Application to Human-Computer Interaction," Human-Computer Interaction, 1997, 12, pp. 391-438.

[49] M. D. Bugajska, A. C. Schultz, J. G. Trafton, M.Taylor, & F. E. Mintz (2002). A hybrid cognitive-reactive multi-agent controller. In Proceedings of 2002 IEEE/RSJ International conference on Intelligent Robots and Systems (IROS 2002). Switzerland.

[50] J. G. Trafton, A. C. Schultz, D. Perznowski, M. D. Bugajska, W. Adams, N. L. Cassimatis, D. P. Brock (2006). Children learning to play hide and seek. HRI 2006: Proceedings of the 2006 ACM Conference on Human-Robot Interaction, v 2006, HRI 2006: Proceedings of the 2006 ACM Conference on Human-Robot Interaction - Toward Human Robot Collaboration, 2006, p 242-249

[51] C. Burghart, C. Gaertner, and H. Woern. 2006. Cooperative Solving of a Children's Jigsaw Puzzle between Human and Robot: First Results. In Cognitive Robotics: Papers from the AAAI Workshop: Papers from the 2006 AAAI Workshop, ed. M. Beetz, K. Rajan, M. Thielscher, and R. B. Rusu, 33-39. Technical Report WS-06-03. American Association for Artificial Intelligence, Menlo Park, California.

[52] J.E. Laird, A. Newell, and P.S. Rosenbloom, "Soar: An Architecture for General Intelligence. Artificial Intelligence," 1987, 33(3), pp. 1-64.

[53] J.E. Laird and P.S. Rosenbloom, "Integrating Execution, Planning, and Learning in Soar for External Environments," In AAAI-90 Proceedings, pp. 1022-1029.

[54] R.M. Jones, J.E. Laird, R.E. Nielsen, K.J. Coulter, R. Kenny, and F.V. Koss, "Automated Intelligent Pilots for Combat Flight Simulation," AI Magazine, Spring 1999, pp. 27-41.

[55] J.S. Albus, "The NIST Real-time Control System (RCS): an approach to intelligent systems research," Journal of Experimental and Theoretical Artificial Intelligence 9(1997), pp. 157-174.

[56] J.S. Albus and A.J. Barbera, "RCS: A cognitive architecture for intelligent multi-agent systems," In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (2004).

[57] C. Schlenoff, J. Albus, E. Messina, A.J. Barbera, R. Madhavan, and S. Balakirsky, "Using 4D/RCS to Address AI Knowledge Integration," AI Magazine, Summer 2006, pp. 71-81.

[58] V. Gazi, M. L. Moore, K. M. Passino, W. P. Shackleford, F. M. Proctor, J. S. Albus, The RCS Handbook, John Wiley & Sons, Inc., N.Y., N.Y., 2001.

[59] E.D. Dickmanns, "An Expectation-Based Multi-Focal Saccadic (EMS) Vision System for Vehicle Guidance," In Proceedings of the 9th International Symposium on Robotics Research (ISRR '99). Stanford, CA: International Foundation of Robotics Research.

[60] R. Bostelman, T. Hong, R. Madhavan, T. Chang, H. Scott. "Performance analysis of unmanned vehicle positioning and obstacle mapping." In Proceedings of SPIE - The International Society for Optical Engineering, Unmanned Systems Technology VIII, 2006.

[61] Nolfi, S. and Floreano, D., "Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines," MIT Press, 2000.

[62] Ricard, M. and S. Kolitz, "The ADEPT Framework for Intelligent Autonomy," in VKI Lecture Series on Intelligent Systems for Aeronautics, von Karman Institute, Belgium, May 13-17, 2002.

[63] Shapiro, D. and Langley, P.. "Controlling physical agents through reactive logic programming," Proceedings of the Third International Conference on Autonomous Agents, 386-387, ACM Press, Seattle, 1999.

[64] Zachary, W., J-C. Le Mentec, and J. Ryder, "Interface Agents in Complex Systems," in C. Nutuen and E. H. Parks (Eds.), Human Interaction with Complex Systems: Conceptual Principles and Design Practice, Norwell, MA: Kluwer, pp. 35-52, 1996.

[65] Bratko, I., *Prolog Programming for Artificial Intelligence,* Addison Wesley, Boston, 2000.