AN ADAPTIVE PLANNING FRAMEWORK FOR SITUATION ASSESSMENT AND DECISION-MAKING ON AN AUTONOMOUS GROUND VEHICLE

By

ROBERT ALLEN TOUCHTON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2006

To Cheryle
My biggest fan since High School
My wife for 35 years (and counting)
My soul mate for eternity

ACKNOWLEDGMENTS

First and foremost, I thank my family not only for their support and tolerance, but also for their enthusiasm and interest in this endeavor. I also appreciate the prayers and tender loving care that my church family provided, both in Jacksonville and Gainesville. They helped me maintain and balance my priorities.

Many, many years ago, when I first conceived the ideas that evolved into the Adaptive Planning Framework, I was working with an amazing team of engineers and software developers. I especially want to thank Dale Gunter, Steve Rausch, and Ken Wilson for their help in getting these ideas out of my head and into a sound, working system.

I would also like to acknowledge the sponsorship and support provided by CIMAR, the Air Force Research Lab, Team CIMAR, and Team Gator Nation that made this research possible. I must add specific recognition of the significant help and support I received from my fellow Graduate Research Assistants Tom Galluzzo, Greg Garcia, Danny Kent, Sanjay Solanki, Eric Thorn, and Steve Velat. I also appreciate the excellent editorial review provided by Michelle LaMontagne.

Finally, I want to extend a special word of thanks to key reviewers of my work, David Armstrong, Dr. Douglas Dankel, Shannon Ridgeway, and Dr. Eric Schwartz, and, of course, to my committee, Dr. Carl Crane, Dr. Tony Arroyo, Dr. Warren Dixon, Dr. Gloria Wiens, and Dr. Charles Winton. Their critiques and ideas were helpful and made a real difference in my work. I add an additional expression of appreciation to Dr. Crane, who welcomed me into the CIMAR autonomous vehicle program and then mentored me through it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

OBJECT

Object page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AN ADAPTIVE PLANNING FRAMEWORK FOR SITUATION ASSESSMENT AND DECISION-MAKING ON AN AUTONOMOUS GROUND VEHICLE

By

Robert Allen Touchton

December 2006

Chair: Carl D. Crane, III
Major: Mechanical Engineering

The primary contribution of this research is the design, implementation, and field testing of an Adaptive Planning Framework (APF) that can address the problem of autonomous operation in a complex, unstructured environment. It encapsulates a new and unique approach to dynamic situation assessment, behavior management, and decision-making. This research also included a literature review and development of a Reference Implementation. The thesis behind this research is that a well-organized, three-stage process of 1) understanding the current situation, 2) understanding the suitability and viability of the available behaviors in light of that situation, and 3) providing the capability to autonomously make and execute behavior-related decisions, all in real-time, provides new levels of intelligence to autonomous ground vehicles (AGV).

This research was performed using the resources of the UF Center for Intelligent Machines and Robotics. This environment provided the ability to collaboratively explore engineering alternatives, create experimental software, and test it in a real-world setting, ultimately leading to the creation of the Reference Implementation. All this was aimed at validating the thesis of the research and producing a more robust APF, operationally proven in a representative physical environment.

The Adaptive Planning Framework has been shown to be both a viable method for representing and managing complex, situation-dependent behavior on an AGV and a valuable contribution to researchers tasked with developing and fielding such a vehicle. The viability of the architecture and design was demonstrated by the development and testing of the Reference Implementation. The value of the APF can be measured by the major role it is playing in the architecture and design of the AGV being fielded by Team Gator Nation for competing in the 2007 DARPA Urban Challenge.

The Adaptive Planning Framework makes a significant contribution to advancing the state of the practice of intelligent systems in general and AGVs in particular. Its adoption by Team Gator Nation means that it will be improved and extended by future researchers. Presuming that occurs, this work will have been the catalyst of a new way of achieving more intelligent and more autonomous ground vehicles.

CHAPTER 1
INTRODUCTION

This dissertation documents the author's efforts in pursuit of the Doctor of Philosophy, including the literature search, research results, and field testing of those results via a Reference Implementation. The primary contribution of this work is the creation of an Adaptive Planning Framework that encapsulates a new and unique approach to dynamic situation assessment, planning, and decision-making. The thesis behind this research is that a well-organized, three-stage process of 1) understanding the current situation, 2) understanding the suitability and viability of the available behaviors in light of that situation, and 3) providing the capability to autonomously make and execute behavior-related decisions, all in real-time, provides new levels of intelligence and autonomy to the autonomous ground vehicle community.

Motivation and Statement of Problem

One of the most daunting issues facing autonomous vehicle researchers is how to best exploit sensor and other information discovered during the execution of a plan. If the system takes too long to deliberate on the possible meanings and implications of this newfound data and knowledge, the vehicle may well have progressed beyond the point where it can benefit from it. Indeed, it may now be sitting atop the unforeseen obstacle that spawned the influx of new information that was being processed.

The execution of specific autonomous behaviors is becoming reasonably well understood, such as "waypoint-following with obstacle detection," though improvements and breakthroughs in these areas continue. However, the autonomous selection of which behavior(s) should be invoked, and in what sequence and by what method, is in need of movement in the state of the practice. A new way of thinking about, organizing, and applying situational knowledge to

macro-level planning and decision-making is needed by the autonomous robotics community in order to achieve the full potential of the field.

The research discussed herein addresses the competing needs of intelligent response to newly acquired information and knowledge versus the rapid performance sufficient for that response to have a positive impact.

Research Solution

An Adaptive Planning Framework for incorporating and managing a collection of virtual Situation Assessment Specialists, Behavior Specialists, and a Decision Broker to support an autonomous ground vehicle (AGV) was devised to address these competing needs. The goal for these Specialists is to continually render/update their findings regarding a predetermined set of Conditions, States, Events, and Recommendations that are of importance to the vehicle's other Specialists, and to manage the execution and modification of the vehicle's high-level behavior.

The abstract approach for this research was actually conceived by the author over 20 years ago in support of an Expert System that was being developed for the commercial nuclear power industry (Touchton 1988; Touchton, Gunter, Wilson et al. 1988). The setting was emergency management during off-normal and accident conditions wherein the Expert System modeled the Technical Support Group (TSG) that is invoked during such times to gain understanding of the situation and provide advice to the plant manager. The TSG is made up of multiple experts, each specializing in some aspect of nuclear power plant operation or emergency response. During an actual emergency, one can observe these experts delving in to their respective areas trying to understand what is happening, how the emergency might be progressing, and how to best mitigate it. These findings are presented (usually rather frantically) to the other experts who update their own findings accordingly and then they are presented to the plant manager who must weigh and assess the findings and recommendations of each expert and make the call on

what should be done at that moment in time. The Expert System was designed to mimic this collaborative approach to real-time situation assessment and decision-making.

The focus of the current research was to evolve this concept and apply it to the situation assessment, planning, and decision-making abilities of an autonomous robot–specifically an AGV. The methodology underpinning this framework is that of problem decomposition wherein large, seemingly intractable problems are systematically broken into sub-problems small enough to be efficiently and properly solved. Of course, this must be done in such a way that the solutions to the sub-problems lead to a solution to the overall problem and in a timely enough fashion to actually benefit from the solution rendered. This Adaptive Planning Framework is summarized here and is discussed in detail in Chapter 3.

To facilitate implementation of this framework on a fielded AGV, a Knowledge Representation Scheme has been devised that models a team of cooperating "Specialists" divided into three sub-domains:

- Situation Assessment Specialists–each devoted to rendering their findings regarding a set of Conditions, States, and Events that are likely to be of importance to other Specialists
- Behavior Specialists–each devoted to rendering their Recommendations on the suitability of their associated Behavior Module for controlling the autonomous vehicle, as well as reporting on what behaviors, plans, and sub-goals and other capabilities their Behavior Module might possess
- Decision Specialist–a collection of one or more Decision Brokers charged with considering the recommendations and findings from the other Specialists and making the final determination of how to proceed

The framework also establishes a reasoning mechanism and control strategy for propagating facts into findings into recommendations into executed actions. This strategy must address conflict resolution, truth maintenance and response to missing information and must support asynchronous operation of the entities. The framework may use either a centralized repository (e.g., a blackboard or knowledge store) as the source and sink of all information

produced/consumed by the Specialists, or a decentralized messaging scheme (e.g., a publish/subscribe model) where each Specialist maintains its own copy of what is relevant. Further, the framework places no constraints on the method to be used by a given Specialist, thus supporting a hybrid architecture of various AI and conventional techniques (i.e., a given Specialist could be an Expert System, a Neural Network, a Bayesian Network, a linear program, or a purely algorithmic program). A conceptual representation of the Adaptive Planning Framework can be found in Chapter 3 (Figure 3-1).

This "cooperating specialists" framework has some similarities to the multi-agent architectures (i.e., "cooperating agents") discussed heavily in the literature (e.g., Panzarasa, Jennings and Norman (2002) which itself cites 90 references most of which are on the subject of multi-agent systems). However, the most significant differences here are that these Specialists operate under a deliberative (vs. emergent) control strategy; each is designed and tuned to do their assigned job (vs. each being a replica of the others); and ultimate authority is given over to the Decision Broker (vs. a decentralized, negotiated decision process). So, while these "specialists" discussed here would fit well under many of the working definitions of software agents (Franklin and Graesser (1996) provide a useful taxonomy that supports this), the term "agent" will be avoided for the sake of clarity.

Research Setting

The research discussed in this dissertation was conducted under the auspices of the Center for Intelligent Machines and Robotics (CIMAR) at the University of Florida. CIMAR conducts both sponsored and independent research in the area of unmanned systems, including autonomous ground vehicles. To support this research, CIMAR provided a collaborative research environment that offered support in the areas of computer hardware and software, mobility platforms, perception, control and testing. CIMAR's active involvement in the Joint

Architecture for Unmanned Systems Working Group (JAUS WG) contributed immensely to the efficiency and consistency of taking ideas and concepts into the field for experimentation and testing. Much of the current research was conducted under the sponsorship of the Air Force Research Lab (Panama City, Florida) to support its work in providing autonomous perimeter surveillance and force protection. The balance of the research discussed herein was conducted as part of CIMAR's involvement in DARPA Grand Challenge, including active use of the NAVIGATOR vehicle (see Figure 1-1). CIMAR also provided access to the IFAS facility in Citra FL, the Energy Research and Education Park on campus, and the Gainesville Raceway for field-testing (see Figure 1-2).

CIMAR's NAVIGATOR AGV has emerged as the primary field-testing platform for the current work. It is a highly mobile all-terrain vehicle capable of autonomously traversing severe off-road terrain and achieving speeds approaching 30 mph on smooth terrain. It provides multiple LADAR sensors, a highly precise localization capability (twin GPS systems, a Smiths Aerospace Inertial Measurement Unit, and a drive shaft encoder), and actuation of its throttle, brake, steering, and gear shift. The NAVIGATOR's software runs on a bank of eight networked Linux-based computers (including one dedicated to the Adaptive Planning Framework) and follows a JAUS-compliant component architecture that includes components for sensor arbitration, path planning, and motion planning, all of which are necessary to support this research (see Figure 1-3 for a glimpse at the state of the design in the early stages of the research).

The Joint Architecture for Unmanned Systems (JAUS) is a component-based architecture and inter-component messaging system sponsored by the Department of Defense. Its goal for this evolving standard is to achieve interoperability, not only among the components, payloads,

and Operator Control Units that comprise an unmanned system, but also among multiple unmanned systems and components produced by different organizations. The availability of the JAUS-compliant components and messaging system on the NAVIGATOR is key to the rapid progress that was made during the field-testing of the Adaptive Planning Framework.

Finally, the ability to collaboratively explore engineering alternatives, efficiently create experimental software, and test it in a real-world setting enabled the creation of thoughtful experiments, and ultimately the Reference Implementation. All this was aimed at validating the thesis of the research and producing a more robust Adaptive Planning Framework, operationally proven in a representative physical environment, and of value to the autonomous robotics research community.

Figure 1-1. The NAVIGATOR AGV in the Mohave Desert.

A B

Figure 1-2. Views of Testing Sites. A) Road Course at Gainesville Raceway. B) Citra test site.

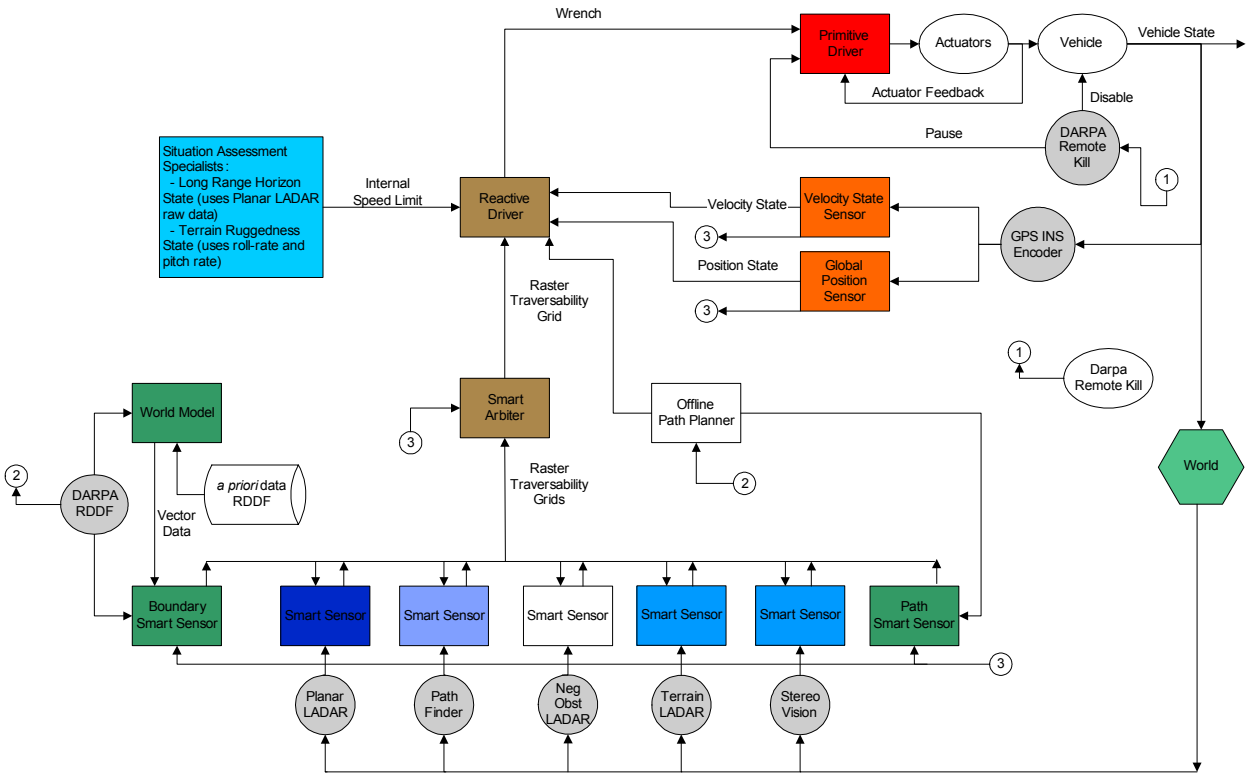


Figure 1-3. As-built NAVIGATOR Component Block Diagram used in 2005 DARPA Grand Challenge.

CHAPTER 2
BACKGROUND AND LITERATURE REVIEW

To be of use across multiple organizations and domains, the work produced here must preserve interoperability at an architectural level. Much of this has been addressed by ongoing standardization efforts, which deal with many of the general interoperability issues that affect the research. The Architectural Compatibility with Emerging Standards section addresses this topic. The Situation Assessment, Planning and Decision-making, and Knowledge Representation sections seek to catalog recent and current work that related to or influenced the research discussed herein.

Architectural Compatibility with Emerging Standards

Each of three predominant standards are discussed in the subsections that follow, along with an assessment of how the current work maintains compatibility, areas that need to be evolved, and any open issues. In addition, one lesser known architecture is presented due to its similarity and relevance to the current research.

JAUS RA

The Joint Architecture for Unmanned Systems (JAUS) Reference Architecture, Version 3.2 (JAUS 2005) defines a set of reusable components and their interfaces. In order to ensure that the architecture will be applicable to the entire domain of unmanned mobile systems, the following four characteristics have been considered by the JAUS Working Group in the creation of the Reference Architecture:

1. Vehicle platform independence. In order for JAUS components to be interoperable, no assumptions about the underlying vehicle or its means of propulsion are made.
2. Mission isolation. The JAUS components can typically be assembled such that a variety of missions can be supported.

3. Computer hardware independence. No assumption of or requirement of particular computer hardware is made. This allows for future adaptability and enhancement as new computer hardware becomes available.

4. Technology independence. This is similar to the computer hardware independence, but focuses more on the technical approach rather than the computer hardware. For example, many approaches could be used to determine vehicle position and orientation. No one approach, such as GPS, inertial dead reckoning, or landmark-based navigation for example, is specified.

As currently defined, the JAUS RA establishes a pre-defined set of standard, yet flexible, components that provide a menu of capabilities that can be drawn from to design an unmanned system. Components are divided into five categories:

- Command and control components
- Communications components
- Platform components
- Manipulator components
- Environmental sensor components

The RA also defines a standardized messaging construct (header and content) that enables JAUS components to exchange information in an efficient and robust fashion. The messaging approach first defines the content and usage of a standardized JAUS Header. It then prescribes the legal JAUS data types that can be incorporated into a message. Then it defines each JAUS message.

The Adaptive Planning Framework Reference Implementation (see Chapter 4) was developed within such components and using such messages as defined by JAUS. Specifically, the Reference Implementation in support of this research is cast in an operational JAUS-compliant vehicle. That is to say that the NAVIGATOR is fully compliant with the JAUS RA 3.2, as extended by permitted "User-defined Components" and "Experimental Messages." However, the concepts and ideas that make up the Adaptive Planning Framework are not tied to nor specifically depend on JAUS. This enables other organizations to implement the framework

in an alternative architecture, such as those discussed in the next three subsections, or in future evolutions of the JAUS Reference Architecture itself. This latter point is especially important in light of the current JAUS initiative to transition the RA into an SAE standard under its newly formed AS-4, Unmanned Systems subcommittee.

NIST 4D/RCS

The National Institute of Standards and Technology (NIST) has been working for over two decades on establishing a standardized approach to the intelligent control of unmanned vehicle systems. The most comprehensive summary of their approach is given in NISTIR 6910, *4D/RCS: A Reference Model Architecture* (NIST 2002). The 4D/RCS architecture is itself derived from NIST RCS, a domain-independent architecture developed by NIST a decade plus earlier (see NIST (1992) for a good overview of the generic RCS methodology). 4D/RCS goes on to specialize RCS to the domain of intelligent vehicle systems for military use.

4D/RCS focuses on ways to ensure that military missions involving unmanned vehicles can be analyzed, decomposed, distributed, planned, and executed in an intelligent, effective, efficient, and coordinated fashion. 4D/RCS describes in detail the functions and associated interfaces necessary to provide sensory processing, world modeling, knowledge management, cost/benefit analysis, and behavior generation. Of particular interest is its hierarchical treatment of time, providing a temporally layered set of eight planning/execution regimes (see Figure 2-1). For example, it suggests that a vehicle Subsystem Planner (Level 3) ought to execute at ~1-5 Hz with a 5 second, 50 meter planning horizon at a 40 cm grid map resolution, while a Section Planner (Level 5) might need to re-plan every 50 seconds, with a 10 minute, 5 km planning horizon at a 40 m grid map resolution.

4D/RCS defines notions of Value Judgment, Mission Planning, and Behavior Generation that are somewhat analogous to the Situation Assessment, Planning and Decision-making notions

presented in the current work, making them readily transferable to 4D/RCS. Another area of accord is that both frameworks support the use of hybrid technologies for implementing any given functionality.

A challenge posed by 4D/RCS is that their hierarchy of nodes calls for each node to possess a complete set of functional capabilities (i.e., World Model, Value Judgment, Behavior Generation, etc.), scaled and scoped to its level of operation in the hierarchy. The partitioning, decomposition, and distribution of the Adaptive Planning Framework Specialists and Decision Brokers across a 4D/RCS hierarchy will be a completely new research area. Of greater concern is that 4D/RCS puts a great deal of power and functionality into their World Model, including prediction and simulation. It will be a non-trivial task to evolve the Adaptive Planning Framework to exploit predicted outcomes and to take advantage of an on-board simulation capability. However, it should be noted that both of these areas of concern stem from a lack of maturity of the framework and not from a weakness in its design.

Service Oriented Architecture/Component Oriented Architecture

Several facets of the Information Technology (IT) sector have been working to establish standards that support software interoperability across diverse organizations under the moniker of Service Oriented Architecture or SOA. SOA enables loose coupling among diverse software entities across a common network. This is accomplished by maintaining a strictly enforced standardized interface among the entities and a standardized messaging construct that enables one entity to request a service from another entity and for that service provider to send its response. This rapidly emerging standard is of interest here because the JAUS WG has begun a transition to a SOA-style architecture and thus it will be important for ensuring long-term compatibility of the Adaptive Planning Framework.

The most mature of these efforts is sponsored by the World Wide Web Consortium (W3C), which relies heavily upon SOA as the foundation of its Web Services initiative and, therefore, is leading the way in its maturation and adoption as a standard. Web Services extend the SOA concept to address anonymous entities that can discover one another and engage one another's services autonomously over the World Wide Web. They have published a treatise on the Web Services Architecture that includes an excellent overview of SOA in Section 3.1 of W3C (2004). They go on in that section to outline some of the pitfalls of a SOA, such as network reliability and latency, lack of shared memory between service provider and consumer (i.e., everything that must be conveyed from one entity to another must be done explicitly via message content, and side effects of receipt of a message must be well understood and agreed upon), concurrency mismatches, and so on.

Industry has also taken a strong role in promoting SOA as a *de facto* standard. IBM (http://www-128.ibm.com/developerworks/webservices/standards/), Sun Microsystems (http://java.sun.com/developer/technicalArticles/WebServices/soa2/SOATerms.html#soaterms), and Microsoft (http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp), to name three, have all embraced the notion.

Academia has also been active in this arena. IEEE Computer Society has formed a Technical Committee on Services Computing (http://tab.computer.org/tcsc/link.htm), and ACM has been actively including SOA topics in many of their conferences and symposiums.

A closely related predecessor to SOA is component-based architecture (COA), which differs primarily in its stronger predisposition of what services a software entity ("component") will provide and less standardization of how components communicate with each other. In other words, COA does not worry so much about a component performing a single task (as in SOA) as

long as the multiple services provided by a given component, and the interface for executing those services, are well documented. The emphasis is on providing a good platform for problem decomposition and loose coupling among components, with less emphasis on component interoperability. Aksit (2002) provides an excellent compilation of articles on the topic of COA, especially Chapter 3, "Component-Based Architecting for Distributed Real-Time Systems," which, in turn, includes a detailed example of using a COA to devise a Car Navigation System (page 85). All in all, SOA can be considered a maturation, and perhaps specialization, of COA.

The main conclusion that can be drawn from the review of these modern software architectures is that the results of the current research are naturally congruous with them. As JAUS conforms more to a SOA over time, there should be little negative impact on the Adaptive Planning Framework described herein. In fact, the more that the government and industry adopt such architectures, the more important it will be to have such a framework available.

DAMN

The Distributed Architecture for Mobile Navigation (DAMN) was originally published as a Ph.D. Dissertation (Rosenblatt 1997) and, while not as widely adopted as the architectures discussed above, it has provided many useful insights for the current work. Even though the scope of DAMN is limited to navigation and obstacle avoidance, its distributed approach, its support of hybrid planning and implementation styles, its blend of centralized and decentralized processing, and its thoughtful treatment of salient challenges to real-time decision-making all make it worthy of elaboration here.

The basic premise behind DAMN is that centralized arbitration of distributed decision-making processes provides a reasonable and useful balance between the demands for real-time responsiveness and the challenges brought about by the asynchronous, latency-filled, heterogeneous, uncertain environment encountered by an autonomous ground vehicle. As in the

other architectures discussed, DAMN provides a modular, extensible, and interoperable framework for supporting the generation and arbitration of sensor data, behaviors, and commands to the mobility platform, controllers, and actuators. This notion is shown schematically in Figure 2-2, where sensor data and high-level commands have been bundled with the assortment of behaviors depicted.

The treatise goes on to present and analyze alternative structuring of the placement of arbitration (e.g., sensor vs. command vs. effect) and to explore various action selection schemes. A detailed presentation of the DAMN implementation on a CMU Navlab AGV and the experimental results achieved provides further insights into the merits and shortcomings of the architecture. Another major contribution of that research was the application of utility theory to the behavior arbitration process, as further discussed later in this chapter.

Situation Assessment

The situation assessment domain of interest to this research is that of an unmanned system understanding its surroundings and status at a higher, more abstract level than that provided directly by its perception systems. In reviewing the literature, one must filter the use of the term when used in the context of the design of manned combat systems; such references often address such topics as own and enemy radars, missile tracking, and weapon lethality. Most such references are in the context of providing situational assessment for a human (Howard and Stumptner 2005; Yanco and Drury 2004), such as pilot support on board a combat aircraft. Of interest here, however, is the applicability to unmanned systems, wherein the raw sensor and signal data is processed into more general situational conclusions, usually as a result of some form of inference or deduction. For clarity, the term "situation assessment" when used in this document will refer to this latter connotation. This domain is sometimes mentioned in the

literature as "situational awareness" and could be referring to either of the connotations discussed above.

The objective of reviewing literature on situation assessment was to determine whether a framework or technology already exists that could be adopted or adapted for use in the current research. Although the review uncovered none, there was certainly parallel research that provided insights and has influenced the research. Work at USC (Zhang and Hill 2000) described the use of templates and patterns to provide situation assessment in virtual humans. They demonstrate a way to use situation assessment to improve decision-making by allowing the software system to better focus its attention (i.e., computing resources) with the goal of improved utilization of on-board resources.

Of particular interest is the work underway at NIST. They are working in several areas that address situation assessment. One has to do with incorporating situation assessment feedback to human operators of robotic devices (Scholtz, Antonishek and Young 2004). While their emphasis is on the human-machine interface, there are insights to be gained from the situation identification and classification schemes that they developed. An even more relevant front is their work on using 4D/RCS to control on-road robotic vehicles. There are both formal papers (Schlenoff, Madhavan and Barbera 2004) and materials and presentations available on the NIST web site (see Figure 2-3) that demonstrate ways to incorporate situation assessment notions into the 4D/RCS architecture.

Much of the literature revealed material that would provide potential additions, alternatives, or improvements to the Situation Assessment Specialists envisioned for the current research. For example, Weiss, Philipps, To et al. (2005) present a capability that could be adapted into a Traffic Specialist. It provides situation classification and prediction for an

assortment of expressway-related conditions (such as same/different lane assignment for other vehicles that are detected), and states (such as approach rate {approach | approach with distance warning | approach with collision warning}). Similarly, Hillenbrand, Kroschel and Schmid (2005) introduce material that could form a Collision Avoidance Specialist that could manage interactions with moving obstacles using such notions as "time to collision," "time to brake," and "time to disappear." Another area of interest is vehicle self-awareness and work such as Reichard and Crow (2005) sheds light on how a Vehicle Health Specialist might be devised.

Finally, it should be noted that much of the discussion of situation assessment in the literature was secondary to a broader discussion and is, thus, of most use in providing insights into possible nomenclature and classification. References such as these are discussed in the Knowledge Representation section rather than here.

Planning and Decision-making

Since the scope of this topic is so broad, its treatment here will be, first of all, limited to the domain of real-time planning and decision-making on an AGV and then further organized as an assortment of "views." As it relates to the Adaptive Planning Framework, the notion of planning refers to the orchestration of executable behaviors to achieve a goal (e.g., find a series of waypoints that will take the vehicle to a desired goal, then drive the vehicle to those waypoints while avoiding obstacles, obeying driving rules and maintaining stability), as well as the low-level planning conducted by a given behavior (e.g., finding an obstacle-free path towards the next waypoint within the perception horizon of the vehicle).

The goal of this phase of the literature review was to better understand the planning and decision-making domain, especially regarding those approaches and techniques that might be suitable candidates for inclusion in the Adaptive Planning Framework. An outgrowth of this goal was assembly of a menu of Behavior Specialists based on the assortment of types and styles

of on-board behavior mechanisms found in use across the AGV research community. The following list of behavior primitives are but a sampling of those gleaned from the literature:

- Seek Goal
- Avoid Obstacles
- Follow Road
- Respond to Blocked Path
- Explore
- Wander
- Maintain Stability
- Seek Target/Intruder
- Intercept Target
- Mark Location

Viewed as a Sense-Plan-Act Problem

This is perhaps the most fundamental view of autonomous control of a mobile robot and one into which many autonomous robotic implementations can be cast. The notion is to neutralize uncertainties in the robot's perception of its world, its understanding of its own state, and the effects of its own actions by indirectly "closing the loop" through the continuous gathering of feedback from its environment while executing its plan (Nilsson 1998). Since it is anticipated that the plan itself will be divided into a sequence of steps, the idea is that the results of executing the initial steps can be observed and compared with expected results. If expectations are not being adequately met (in essence, forming an "error" signal), then the subsequent steps can be adjusted accordingly, or an entirely new plan can be published. It is presumed that the robot will have an ability to store its perception and state knowledge in some form of a world model, which can, in turn, be used by the planner.

This design style best describes the autonomous control used on the NAVIGATOR (Crane, Armstrong, Touchton et al. 2006). The four environmental sensors publish their findings, in the form of a traversability grid, to a sensor arbiter. Two additional pseudo-sensors each publish a traversability grid to the sensor arbiter denoting the *a priori* route boundary and *a priori* path

plan. The sensor arbiter then fuses these inputs and publishes to the receding horizon planner a comprehensive traversability grid, which represents a localized view of a world model. The receding horizon planner uses an A* search algorithm and a simple vehicle model to iteratively produce viable plans that achieve a desired goal state and to choose the one that minimizes traversability cost. The goal itself is based on the *a priori* path plan and is replaced with a new goal once the vehicle nears it. The planning search that occurs has as its only objective the publishing of an instantaneous wrench command (steering, throttle, brake) to the vehicle's primitive driver, whose job is to execute that wrench as actuator positions. Thus, every cycle of the planner produces a new wrench command. Since every component in the chain executes at a nominal rate of 20 Hz, a new "plan" (as manifested in the instantaneous wrench command) is always being issued, thus providing a responsive behavior, with some deliberation on how that behavior is generated. Figure 2-4 shows a snapshot of an arbitrated traversability grid and the instantaneous plan.

One difference in the NAVIGATOR's implementation of the Sense-Plan-Act paradigm is that, by encapsulating the *a priori* plan into a pseudo-sensor whose findings compete with those of the other sensors, the conventional aspects of planning provide only "suggestions" for a preferred action, rather than forcing the vehicle onto a defined course. Although implemented quite differently, this notion is in concert with the findings of Payton, Rosenblatt and Keirsey (1990), who go on to note that "In general, internalized plans should be conceived as representations that allow the raw results of search in an abstract state space to be made available as advice to continuous real-time decision-making processes.(p. 16)"

There are many good examples of robotic systems that have implemented some fashion of the Sense-Plan-Act paradigm. Most have to do with navigation and obstacle avoidance, such as

Batavia and Nourbakhsh (2000). Examples of this paradigm applied to other aspects of robotic planning and decision-making are much harder to find.

Viewed as a Subsumption Problem

The notion of empowering a mobile robot to operate without any centralized control was first introduced by Rodney Brooks as he devised a self-managing, layered control scheme dubbed the "subsumption architecture"(Brooks 1986). By decomposing a robot's control system into layers of task-achieving behaviors, control is governed by the dominant layer in play at an instance in time, which, in turn, "subsumes" the behaviors of the lower level layers. For example, let "Wander" be considered a level 1 behavior and "Explore" a level 2 behavior. Since Explore is the higher level, it will self-determine whether exploring is an appropriate behavior under current circumstances. If so, then it will alter the Wander behavior to be not random, but to fulfill its wishes to visit new areas. If not, then it will allow the Wander behavior to proceed without any alteration. This notion is extrapolated across all possible behaviors. This style of planning and decision-making is often referred to as "reactive." The resultant behavior of the robot is referred to as "emergent" since it is likely that the observed behavior is some extemporaneous blend of the possible behaviors that the robot could execute.

This approach to planning and decision-making has a dedicated following and is especially appealing for multi-agent and swarm applications. For example, the subsumption architecture and reactive behavior play a major (though not exclusive) role in the design of robots at the Idaho National Lab (see www.inl.gov/adaptiverobotics). The primary point of departure for the Adaptive Planning Framework is the existence of the Decision Broker whose role is to oversee the overall operational behavior of the robot.

The differences between these first two views can be captured by the relative importance placed on each of the three components of the Sense-Plan-Act view. For example, a purely

reactive system does almost no local planning since every stimulus anticipated during the sensing stage has a prescribed behavioral action, thus relegating the planning stage to simply resolving action conflicts when more than one stimulus is perceived or queuing and dispensing actions when one stimulus invokes multiple actions (i.e., managing the subsumption process). Conversely, a deliberative system will have a large emphasis on the planning stage, attempting to formulate a new plan that incorporates newly sensed information along with any changes in state of the vehicle or its mission while simulating the effect of alternative actions on the quality and viability of the plan. The juxtaposition of the Sense-Plan-Act view's emphasis on deep planning through possibly time-consuming deliberation and the Subsumption view's potentially unpredictable, but fast, reaction to stimuli, explains why researchers are still seeking other, hybrid or blended, planning and decision-making styles.

Viewed as a Decision Theory Problem

Another rich area of exploration is how classical decision theory might be applied to the AGV domain. For example, Karacapilidis and Papadias (2001) describe how argumentation can be automated and used to support collaborative decision-making. Perhaps their ideas for automating argumentation constructs, such as "Scintilla of Evidence," "Beyond Reasonable Doubt," and "Preponderance of Evidence," can play a role as the design of the Decision Broker evolves.

Rauenbusch and Grosz (2003) and others speak of devising explicit "Plan Trees" whose nodes encapsulate the desired action/behavior, associated constraints, and contextual applicability and whose structure models the desired decision-making outcomes. The search through the tree is conducted using some measure of cost or value such that the correct path through the tree delivers the correct series of actions/behaviors. The use of trees to represent and manage the planning and decision-making process was examined as part of the current research

and their use could become an integral part of the Adaptive Planning Framework in the future (see the Future Work section of Chapter 5).

Hoffman and Yates (2005) present a synopsis of what has become known as the "three-step decision-making process." In this paper, they report that most, if not all decisions can be modeled as a cascading set of three-step activities. One of the models specifically referenced for use in process control is 1) Situation Assessment, 2) Planning, and 3) Commitment to a course of action. Each of these steps may be expanded into another three-step decision-making process, such as deciding which situational conditions are present or relevant, or whether to keep or abandon a committed action. The validation given for the three-phase Adaptive Planning Framework is comforting while their eleven Cardinal Issues for devising intelligent decision aids (to supplement human decision-making) provide food for thought on how these issues might impact autonomous decision-making on an AGV.

A final realm under decision theory that was reviewed is known as Hierarchical Task Network (HTN) planning. Erol, Hendler and Nau (1994) provide an overview of this concept and cites the seminal works that have contributed to it on the way to introducing a formalism of HTN planning semantics. The basic premise of HTN planning is to iteratively decompose tasks until primitive tasks are reached (defined as tasks that cannot be further decomposed and that are actionable). These primitive tasks are assembled into a network of increasingly abstract tasks allowing a planning algorithm to select a high-level task, recursively expand its children until its primitive tasks are reached. Some expansions may be constrained based on the current situation, thus pruning the search when compared with an unconstrained expansion of the network. Each reachable path from the high-level tasks to the primitive tasks becomes a candidate plan. While this exploration of the HTN is taking place, the candidates are being evaluated by so-called

"critics" so that any arising conflicts can be identified and the winning candidate declared. Because of its deep reasoning, HTN-based planning is not typically used for real-time applications. However, the notions embodied in it have influenced the Adaptive Planning Framework Decision Protocols discussed in the next Chapter. A Decision Protocol is similar to an HTN "method," which contains the instructions for how to expand a non-primitive task, but benefits from *a priori* linkage and entry/exit condition constraints. This in essence eliminates the abstract searching and testing aspects of the HTN approach.

Viewed as a Behavior Arbitration Problem

The concept of Behavior Arbitration was introduced as part of the Distributed Architecture for Mobile Navigation (DAMN) (Rosenblatt 1997) as a key ingredient for achieving its goal of balancing centralized and decentralized design styles. All (decentralized) behavior generators submit their control output (referred to as a "vote") and the (centralized) DAMN Arbiter fuses their votes into a single command set to the vehicle. This approach has similarities to the Adaptive Planning Framework's Decision Broker, with the main difference being that the Decision Broker may empower a given behavior to possess sole vehicle control (that is, without submitting its vote to the Decision Broker for each command) while silencing others until the situation warrants. This notion builds on Rosenblatt's centralized/decentralized hybrid architecture while reducing command stream latency by allowing one controlling entity to be in direct command of the vehicle.

"Utility fusion," which uses traditional utility theory to provide an alternative to command fusion, is another concept that evolved from DAMN (Rosenblatt 2000). This notion requires each behavior generator to submit a probabilistic utility estimate along with its vote, thus enabling a "utility arbiter" to compute the Maximum Expected Utility and use it to select the

optimal behavior. Some flavor of this idea may have merit in how the Decision Broker is designed and should be studied further as part of any future research.

Viewed as an Action Selection Problem

Action Selection is another way to view planning and decision-making on an AGV. In this view, the mobile robot is tasked with selecting the most appropriate action based on the current situation, in spite of inaccurate, incomplete, and possibly unforeseen information. For this to happen, there must exist some catalog or repository of possible actions from which to select and the criteria upon which to base a selection decision. Pirjanian (1997) provides an excellent overview of ten varying approaches to the action selection problem. In this treatise, he summarizes each (including DAMN), then compares and contrasts them in terms of eight criteria, including planning vs. reactivity, synchronous vs. asynchronous, hierarchy vs. no hierarchy, and knowledge representation which all have a direct bearing on the current research.

NIST has also developed an approach to action selection via its hierarchical planning and control scheme (Lacaze 2002; Murphy, Abrams, Balakirsky et al. 2000). The scheme enables the system to plan at different rates at each level, with the scope of planning fixed for each level. For example, high-level goal planning might take place at a lower resolution and update rate, but would cover a larger expanse than say planning for obstacle avoidance. The plans themselves are broken into a tree or graph of subgoals and subtasks (task decomposition itself is discussed under Knowledge Representation) and the actions are selected, executed, and monitored in accordance with the defined planning levels. The planning levels are chosen to be consistent with the time, duty cycle, and range horizon parameters established in the 4D/RCS architecture. For example, AGV mobility planning is broken into four levels: Servo, Prim(itive), Autonomous Mobility, and Vehicle System. Balakirsky and Lacaze (2000) elaborate how planning, in the

form of Value Judgment and Behavior Generation, takes place for the Vehicle System planning level.

Since the Adaptive Planning Framework itself has an action selection flavor, it was important to understand and consider the strengths and weaknesses of these other approaches and to adopt what is good about each while attempting to remedy their shortcomings. Further, since the Adaptive Planning Framework was to remain congruent with NIST's 4D/RCS hierarchical planning approach, an understanding of their approach to action selection was needed.

Viewed as an Adaptive Planning Problem

Note that in some literature, "adaptive" is used to mean that the system "learns" from its experience, thus improving its performance over time, whereas the connotation used here is that the system alters its plan based on new, situational information that has been provided by upstream knowledge and data processing. Thus, while the possibility of actually changing the *a priori* behaviors from which to choose through learning should not be ruled out for future generations of the Adaptive Planning Framework, it is certainly not the emphasis or the motivation for using the term "adaptive" in its moniker. The genesis of adaptive planning as used here was a search to improve the performance of (manual) military mission planning through the use of expert systems, such as the Adaptive Mission Planning System in Seares (1987). The quest continues as military planners seek to reduce 24-month planning cycles down to a year or less for complex deployments and even less for Crisis Action Planning (Hoffman 2004). In fact, their definition, "Adaptive Planning is the systematic, on-demand creation and revision of executable plans, with up-to-date options, as circumstances require," (p. 3) could suffice for the work conducted here as long as its transition to an autonomous, real-time setting is understood.

The need to alter a plan already in progress can have a number of causes, including insufficient time for completion, ineffective results, changes in the situation, and receipt of a new objective to name a few. The Artificial Intelligence community has driven related work in this area, but application to mobile autonomous robotics has not been at the forefront. For example, Hayes-Roth (1995) presents an excellent treatise of an adaptive planning architecture based on the premise that an "agent dynamically constructs explicit control plans to guide its choices among situation-triggered behaviors." (p. 330) To accomplish this, she identified and explored five areas where an intelligent system might require adaptive behavior, depending on the situation encountered:

1. Perception Strategy - Adapt to information requirements and resource limitations
2. Control Mode - Adapt to goal-based constraints and environmental uncertainty
3. Reasoning Tasks - Adapt to perceived and inferred conditions
4. Reasoning Methods - Adapt to available information and current performance criteria
5. Meta-Control Strategy - Adapt to dynamic configurations of demands and opportunities

As an example of more recent work that does focus on mobile robotics in a real-time setting, Hassan, Simo and Crespo (2001) offer a behavior-based architecture that will adapt to temporal constraints by allowing itself to utilize more deliberative techniques when time is available, but moving towards more reactive behaviors when time is at a premium. They also introduce the notion of adjusting the quality of service that a given element might deliver based on the situation encountered. For example, this approach might allow the system to attempt to achieve its goal with a "rough" plan if a "complete" plan could not be delivered in a timely enough manner. Musliner (2001), and his Adaptive Mission Planner, provides another view on how to empower an autonomous system to alter its plans based on temporal constraints and in light of changing environments, objectives, and system capabilities. That work built upon his earlier efforts to devise the Cooperative Intelligent Real-time Control Architecture (CIRCA)

(Musliner, Durfee and Shin 1995), which provides formalisms on how to represent tasks and decisions in a LISP setting. While CIRCA has not been applied in the mobile robotic domain (making its suitability to support an AGV unknown), there are insights to be gained from this work. Finally, NIST has incorporated an element of adaptive planning in their recent work on autonomous on-road driving as part of 4D/RCS. For example, Balakirsky and Scrapper (2004) discusse an expert system and knowledge representation scheme that support adaptive planning for autonomous lane and speed management.

Knowledge Representation

In this section, related work on Knowledge Representation relevant to the domain of AGVs is explored. Knowledge Representation refers to the schemas and constructs used to document, standardize, normalize, and utilize the entities within the domain of interest. It must capture the semantics and meanings of the relationships among the entities, as well as their names, descriptions, attributes, and the method or reasoning mechanism for determining their current state or value.

Sources of such domain knowledge include technical documents, specifications, training manuals, etc. (many of which can be accessed via the web). Example knowledge sources include a table of Autonomous Mobility Situation Coverage Requirements from Demo III requirements analysis (Robotic Systems Technology 1998), a Functional Taxonomy chart for an AGV from a TACOM (the U. S. Army's Tank-Automotive COMmand) PowerPoint presentation (Pritchett 2002), and by drawing analogies from human military operations as found in the Army Universal Task List (US Army 2003). Remaining knowledge gaps must be filled in by interviews of subject matter experts or perhaps empirically through experimentation.

By far, the most work in knowledge representation for intelligent vehicles has been done by NIST. Thus, this section will conclude with an extended example, demonstrating their

approach to representing knowledge about situational conditions, states and events, planning, and behaviors within the 4D/RCS context.

Lexicons, Taxonomies and Ontologies

One technique for knowledge representation is to progress from a lexicon (a domain-specific dictionary of terms), to a taxonomy (a logical ordering and categorization of those terms), to an ontology (an explicit specification of those terms along with the semantics and relationships among them). One on-line dictionary defines ontology as follows:

> An explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them… Definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions or other objects) with human-readable text describing what the names mean and formal axioms that constrain the interpretation and well-formed use of these terms… The hierarchical structuring of knowledge about things by subcategorizing them according to their essential (or at least relevant and/or cognitive) qualities. (Howe 2005)

Much work is in progress attempting to build general purpose, or even "common sense" ontologies that would be useful to all domains. The most famous of these is the OpenCyc project (http://www.opencyc.org/), which, with 47,000 concepts and 306,000 assertions about them to date, is well on its way to achieving its vision to become "the world's largest and most complete general knowledge base and commonsense reasoning engine." Another initiative of interest is the DARPA Agent Markup Language (DAML) project (http://www.daml.org/), which was sponsored by DARPA to create an xml extension that provides (among other things) a rich and suitable language for the creation of general-purpose ontologies (282 distinct ontologies had been created using DAML by the time the program funding was terminated in 2006 and the work absorbed by W3C).

For the AGV domain, and thus for the Adaptive Planning Framework, the scope of the knowledge that must be represented via the techniques discussed in this subsection is still quite

broad. Situational knowledge spans from urban to highway to off-road environments, potential obstacles and hazards that might be present, traffic rules and driving best practices, and so on. Planning and behavior knowledge encompasses a wide variety of missions and tasks, whether they are high-level (conduct search and rescue operation), tactical (pass the vehicle), elementary (change lane), behaviors (avoid obstacles, maintain stability), planning rules and processes, and so on. Even knowledge about "self" or "ego" must be represented, such as capabilities, limitations and constraints, or current status. There is a major initiative under way at NIST, sponsored by TACOM, to develop an Intelligent Systems Ontology that is useful and relevant to the current research. Although still a work in progress, this intelligent-vehicle-specific ontology is expected to provide a standard set of domain concepts, their attributes and their interrelationships, delivered in a fashion that facilitates knowledge capture and reuse (Schlenoff, Washington, Barbera et al. 2005). This ontology is beginning to gain traction as it makes its way into the AGV navigation planning community outside of NIST (Schlenoff, Balakirsky, Uschold et al. 2003).

The strategy embraced for the current research was to a) stop short of building a formal ontology, and 2) accumulate the lexicon/taxonomy/ontology content on an as-needed basis, driven by the needs of the Adaptive Planning Framework Specialists as they are created. The Knowledge Representation Tools section of the next Chapter and their use in the Reference Implementation demonstrate how this was accomplished.

World Model Knowledge Store (WMKS)

Another dimension of knowledge representation is how data, information, and knowledge are stored. Whether it is provided *a priori,* or it is perceived, inferred, or received by the AGV, there must be a place and a format for storing, accessing, and analyzing it. Such data, information, and knowledge are often referred to as the "world model" and the place where they

are stored as the "knowledge store." The breadth and sophistication of the world model knowledge store for a given AGV design will vary widely, depending on its degree of autonomy, the scope of its behavior, the complexity of its design, etc.

Situation Assessment findings, which must also be managed, fit into what some communities refer to as "meta-knowledge," i.e., knowledge about the knowledge. For example, while pumping out its perception data, a sensor could independently assess and report on its own confidence in its findings and its own health, and perhaps even declare that its own results should not be used right now (say, due to a camera white-out).

Although not always so, the knowledge store is usually persistent, using either a relational database or an object-oriented knowledge based system. Since much of the information stored is of a geo-spatial nature, the knowledge store often includes geo-spatial extensions for explicitly representing GIS and topographical data, polygonal objects, etc. Another consideration is whether the WMKS contents are stored in a central location, accessible by all AGV modules (sometimes referred to as a "blackboard architecture") or each module maintains a subset of the WMKS containing just the content it needs, with data, information, and knowledge marshaled among the AGV modules on an as-needed/as-requested basis (sometimes referred to as a "publish/subscribe architecture").

The strategy embraced for the Adaptive Planning Framework was to focus on the World Model content. Since the JAUS platform used for experimentation has a robust messaging system in place, including the needed publish/subscribe mechanisms to support it, publish/subscribe messaging was the approach used for the Reference Implementation.

Knowledge Representation at NIST

NIST advocates task decomposition as a key knowledge representation technique to support the hierarchical control strategy emphasized in its 4D/RCS architecture and has

published widely on various ways to accomplish it (Barbera, Albus, Messina et al. 2004a; Barbera, Messina, Huang et al. 2004b). This technique for representing the actionable elements that could be assembled to create a plan strives to break high-level tasks (e.g., a mission objective) into distinct hierarchical levels and also to identify multiple subtasks at a given level. Figure 2-5A shows an example of how the "GoToDestination" task is decomposed into a "planning graph" that ultimately leads to a specific wrench command to the vehicle. The system must know (or be able to infer) the state of each node in the tree along with the cost of each arc in order for the associated control module to formulate the appropriate plan. Extending the example in Figure 2-5B, a Destination Manager has determined that staying on the current road is appropriate and a Route Segment Manager has decided that passing the vehicle in front of it is the most desirable way to reach the destination. A Driving Behaviors module knows that its own vehicle has already changed into the passing lane and has further determined that the best thing to do right now is to stay in that lane, while a low-level Elemental Maneuvers module has found a wrench that ought to produce the requested outcome. Each Manager or module manages its own situational understanding either from direct sensory input or from its own local subset of the World Model Knowledge Store. Naturally, there are other tree elements and control modules that address following distance, speed, and so on, in addition to non-mobility-related tasks, such as payload management, communications, etc.

Once a plan is devised and approved, its elements must be executed by invoking one or more actions or behaviors, or perhaps by unleashing an entire subsystem to take over low-level control of the vehicle. NIST advocates the use of State Tables to represent the action decision-making knowledge (Barbera et al. 2004a). A State Table is crafted for each node in the Task Decomposition Tree containing the rules that the control module is to use for mapping node

inputs (states or situations) to allowable output actions. Figure 2-6 continues the lane-changing example by showing that once the system determines the situation to be that the vehicle is now in the passing lane ("InPassingLane"), a "FollowLane" Output Action will be sent down to the Elemental Maneuvers module.

To trigger the appropriate and desired state response, the matching situation must be known. The NIST approach to this is to determine and store the cascading precursor situational knowledge as a collection of "world states," but, in conformance to the 4D/RCS architecture, only that subset relevant to a given module. The lane-changing example concludes with a glimpse of the dozens of situational findings that lead up to the finding of interest ("ConditionsGoodToPass"), as shown in Figure 2-7.

It is important to note that this figure only depicts the names of and relationships among the entities shown. The attributes of each and the objects and rule(s) for determining whether each is in effect have not been included here. For example, the LaneMarkingsAllowPass condition would also have stored with it the rule that IF LaneMarkings = BrokenYellow AND LaneMarkingLocation = OurSide, THEN LaneMarkingsAllowPass = True. Further, either the WMKS must contain mapping data that can be queried to determine the lane marking data needed by the rule, or the perception subsystem must be able to provide it by observation.

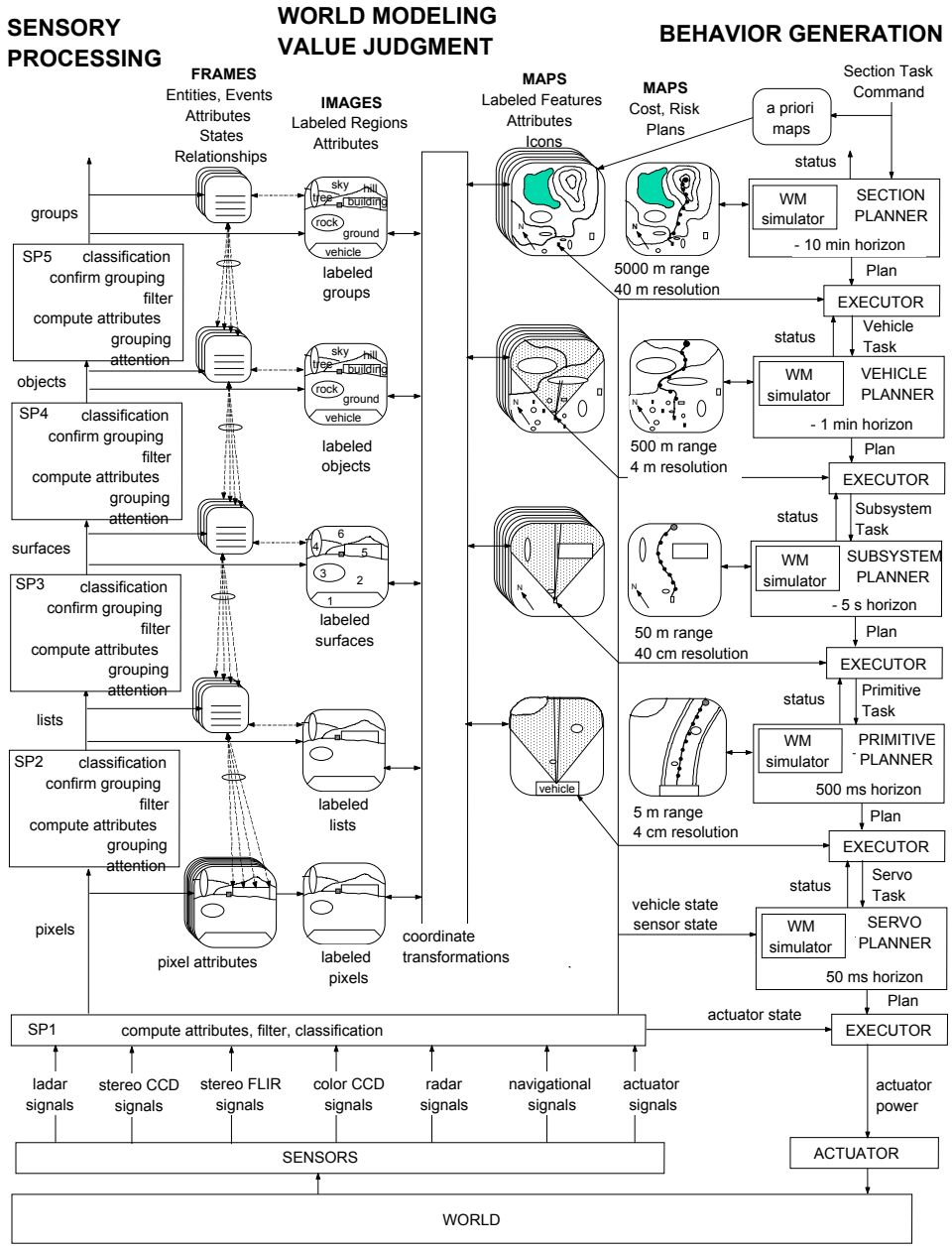


Figure 2-1. Excerpt from NIST PowerPoint Presentation (source: http://www.isd.mel.nist.gov/projects/rcs/presentationhui/sld019.htm, last accessed October 13, 2006).

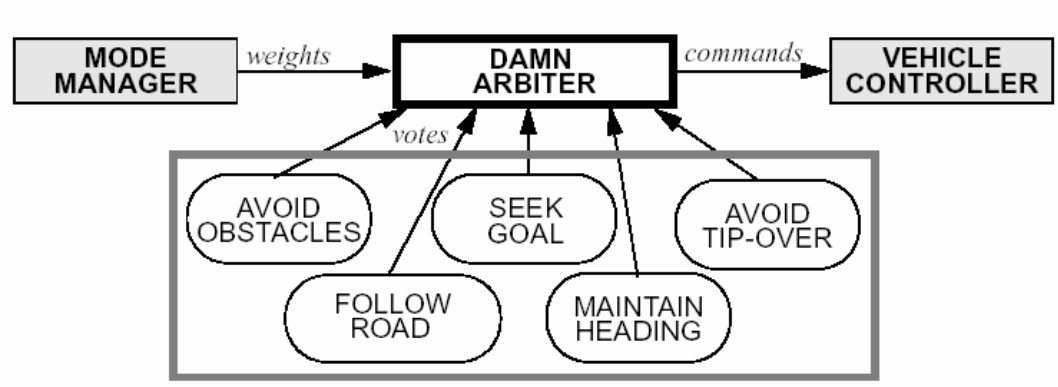


Figure 2-2. DAMN Arbiter and Behaviors (source: Rosenblatt (1997), page 9, Figure 1-2).

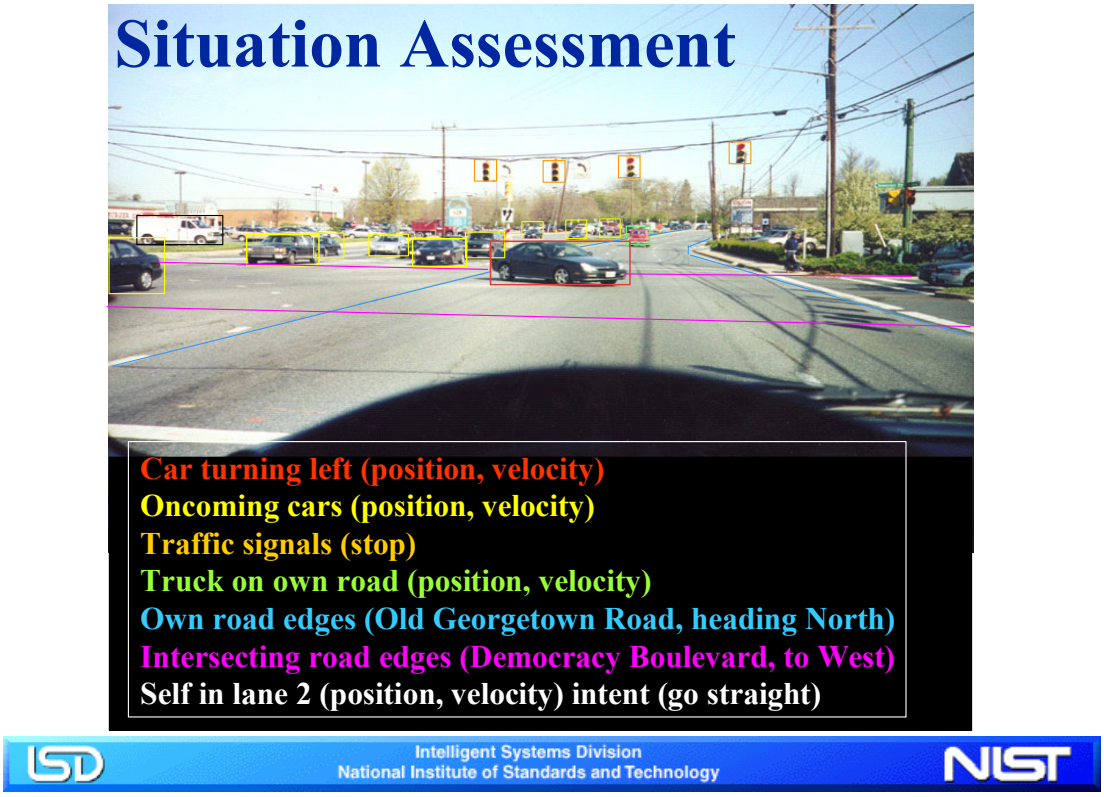


Figure 2-3. Excerpt from NIST PowerPoint Presentation (source: http://www.isd.mel.nist.gov/projects/rcs/presentationhui/sld061.htm, last accessed October 13, 2006).

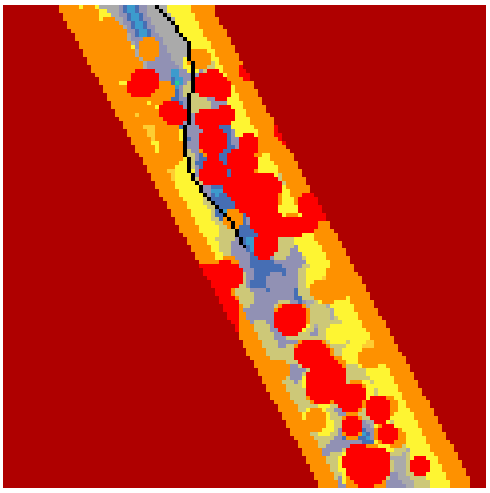
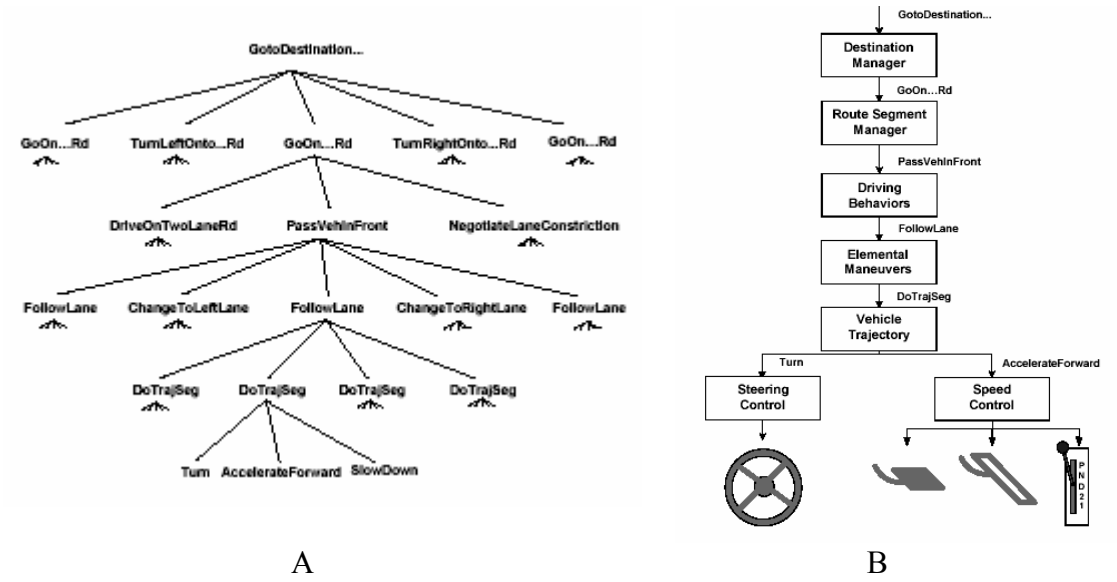


Figure 2-4. Example Traversability Grid taken from the NAVIGATOR while in a Cluttered Roadway (red, orange, and yellow indicate lessening severity of obstacles, gray and blue indicate improving degrees of smoothness of terrain, and the instantaneous plan is indicated in brown).



A B

Figure 2-5. NIST Knowledge Representation Schemes for On-road Driving. A) Task decomposition decision tree. B) Hierarchy of agent control modules (source: Barbera et al. 2004a, Figures 3 and 4).

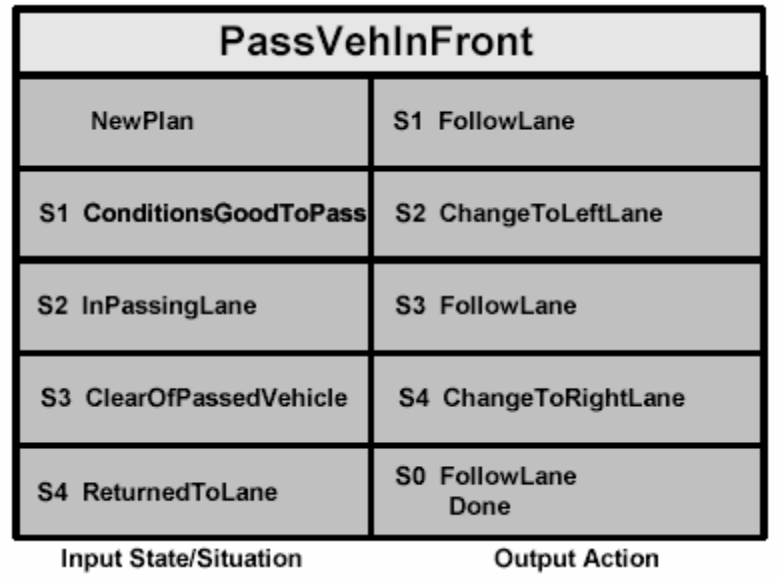


Figure 2-6. NIST Knowledge Representation Scheme for Behavior State Transition Rules (source: Barbera et al. (2004a), Figure 5).

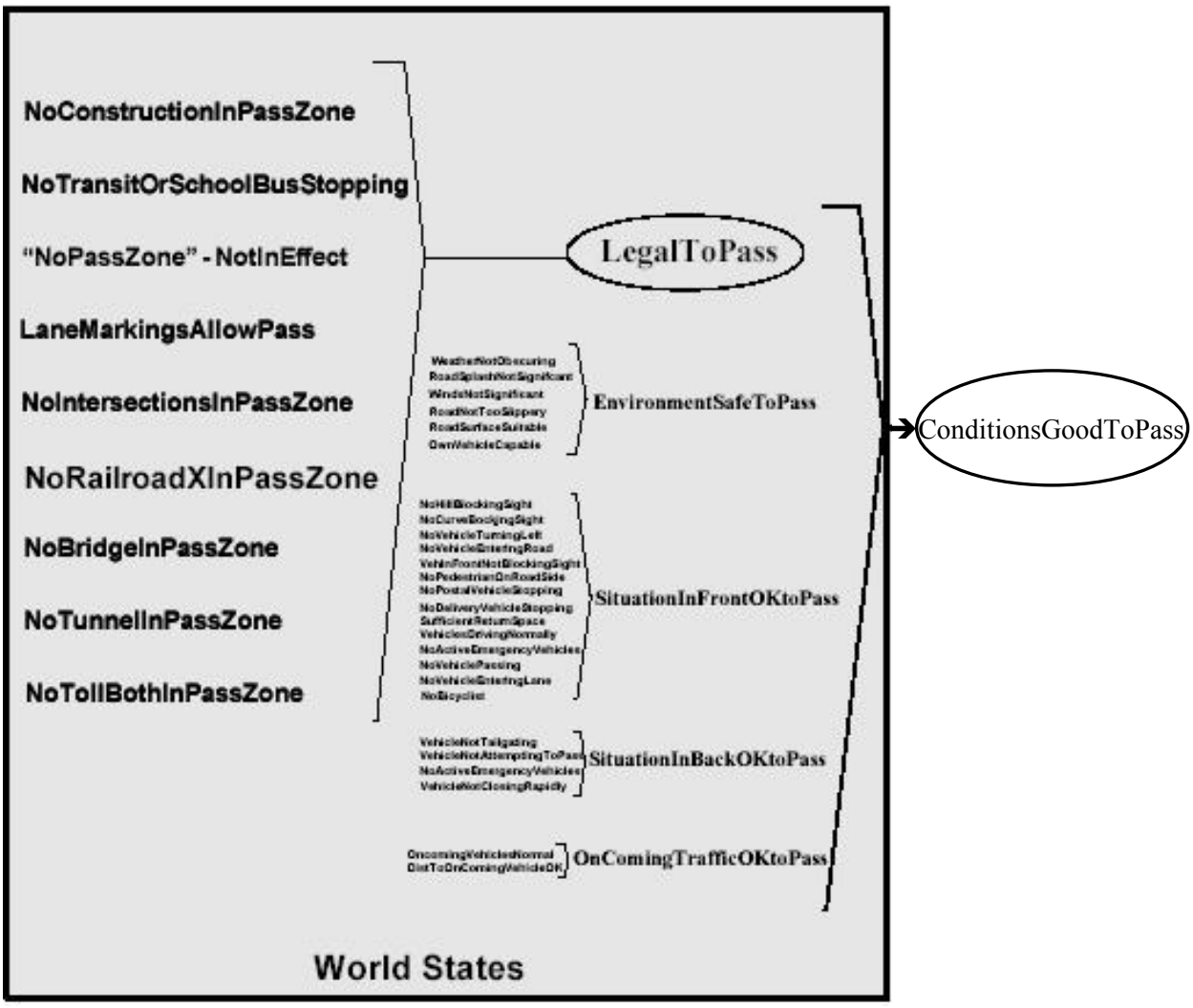


Figure 2-7. NIST partial Knowledge Representation Scheme for Situational Conditions Leading up to ConditionsGoodToPass (source: adapted from Barbera et al. (2004a), Figure 6).

CHAPTER 3
THE ADAPTIVE PLANNING FRAMEWORK

This chapter describes the Adaptive Planning Framework, which was the direct result and primary deliverable of the research performed.

Knowledge Representation Scheme

To facilitate implementation of this framework, a Knowledge Representation Scheme has been devised for use during the design phase of the implementation. It models a team of cooperating "Specialists" divided into three domains:

- Situation Assessment Specialists–each devoted to rendering their "Findings" regarding a set of Conditions, States, and Events that are considered to be of importance to other Specialists
- Behavior Specialists–each devoted to rendering their "Recommendations" on the suitability of their associated behavior for controlling the autonomous vehicle, as well as reporting on what behaviors, plans, and sub-goals and other capabilities their behavior might possess
- Decision Specialist–a collection of one or more Decision Brokers charged with considering the Recommendations and Findings from the other Specialists and making the final determination of how to proceed

The Knowledge Representation Scheme also introduces the notion that situational Findings be restricted to "Conditions," "States," or "Events." Similarly, the Findings of a Behavior Specialist regarding the suitability of their assigned behavior are constrained to "Recommendations." The motivation for these constraints is to allow for better management of the knowledge acquisition and validation process and to add consistency to the reasoning process, without unduly restricting the system developer's ability to adequately model the domain. A conceptual representation of the Adaptive Planning Framework is shown in Figure 3-1.

Below are the working definitions and (non-exhaustive) examples of Conditions, States, Events, and Recommendations developed for this framework for use by the various Specialists:

- Condition: an independent, ongoing circumstance that can (in general) coexist with other conditions and whose value is simply "Present" or "Absent" (i.e., the condition can come and go over time and the goal is to determine the *presence* of the condition). The primary rule for when to classify a situational result as a Condition is that *its absence is not of interest, so it need not be proven* (conversely, its presence will be retracted at each decision-making cycle and must, thus, be re-proven). The following are examples:
  - Close-Range-Obstacle
  - Excessive-Roll
  - Adjacent-Lane-Safe

- State: an abstract entity that can have only one of two or more enumerated values. The value of each State must be explicitly found in order for it to change. The enumeration may be prioritized (or one of its members be assigned as the default value) to resolve ambiguities. The following are examples:
  - Mission-Mode is {Ahead-of-Schedule | Nominal | Behind-Schedule}
  - Mission-Goal is {Optimize-Speed | Optimize-Risk}
  - Mobility-Mode is {Low-Speed | High-Speed}
  - Terrain is {Smooth | Rugged | Very-Rugged}

- Event: a circumstance whose mere occurrence is of interest and may not be ongoing or still in effect (the rule for when to classify a Finding as an Event is that the *occurrence* of the event is what is of most importance). Its Truth-value should be associated with the point in time when the event occurred and an expiration time, after which the occurrence of the event is no longer relevant. The following are examples:
  - Enemy-Fire-Detected
  - Air-Conditioning-Failed
  - Intersection-Became-Clear

- Recommendation: a special case of a State responsible for representing the suitability, appropriateness, or viability of a behavior. Examples include:
  - Passing-Behavior is {OK | Not Appropriate | Not Legal | Unsafe }
  - Roadway-Navigation-Behavior is {OK | Blocked | Stuck | Unsafe}

Notice how each of these example Findings works in the following sentence template: "The <finding-name> is <finding-value>." Although beyond the scope of the current research, this makes it possible to build a generalized explanation facility and natural language man-

machine interface on the foundation provided by the Adaptive Planning Framework. This idea could be extended to include the ability to query a Specialist to divulge its reasoning by putting the input values used in its determination into a similar sentence structure, such as "The <finding-name> is <finding-value> because the <input-finding-1-name> is < input-finding-1-value> and <input-finding-2-name> is < input-finding-2-value>."

Another topic investigated as part of the research was the merit of including "Not Yet Determined," "Unknown," and "Not Relevant" as valid values available to all Findings. These values would be used to inform downstream consumers of the special circumstances affecting the determination of the Finding. "Not Yet Determined" could be reported for cases where the Specialist has not yet begun execution, perhaps as the default value assigned by the constructor of the data structure used to hold the Finding. "Unknown" could be used for cases where a critical input to the Specialist is not available, rendering it unable to render a result of any kind. "Not Relevant" could be used when a certain combination of input values makes the Finding of no interest regardless of the outcome of the rule or algorithm used to produce it. For example, the Terrain Specialist might report that "The Terrain State is Not Relevant" if it realizes that the amphibious vehicle on which it is running was currently afloat.

The Knowledge Representation Scheme assumes that a variety of inputs will be available to the various Specialists:

- Raw (non-visual) sensor readings (e.g., global position, speed, heading, roll, and pitch)
- Derived readings (e.g., rate-of-change of heading/roll/pitch)
- Sensor meta data (e.g., whiteout/blackout or closest object detected), but not necessarily the rasterized obstacle/traversability maps
- Planning and control elements (e.g., mission goal completion rate or remaining waypoints)
- Previous Findings of the various Specialists

Situation Assessment Specialists

The Specialists focusing on Situation Assessment (SA) are organized into categories that together provide comprehensive coverage of the AGV domain. The rationale for organizing Findings into categories is to facilitate the knowledge extraction and validation process. The framework also requires that a particular Finding be managed by only one SA Specialist, thus avoiding potential ambiguities and dual maintenance. The framework is flexible such that new categories can be added as conditions warrant, such as introducing a "Payloads" category if it is deemed that payload-related Findings do not fit under any of the existing Specialists.

The SA categories and examples of typical SA Specialists that might be assigned to them are shown in Table 3-1. As a situation of interest is discovered, it is treated as follows:

- A unique and unambiguous name is given to it
- It is assigned to the most appropriate SA category
- It is given one or more Findings for which it is to be responsible
- Each Finding is classified as a Condition, State, or Event and the data appropriate for the class is determined

Although the Conceptual Model depicts these SA Specialists as independent entities (i.e., the logical view), they may in practice be distributed across the modules and components that make up the AGV's software platform (i.e., the deployment view), more in the spirit of the 4D/RCS approach. For example, the best way to implement the Obstacle Specialist on the CIMAR NAVIGATOR was to embed it as a function call within the Planar LADAR Smart Sensor component.

Behavior Specialists

The Behavior Specialists are organized by the available behavioral modules or selectable behaviors to be deployed on the AGV. Note that the Behavior Specialist is a separate entity from

the behavior itself in that it renders its Findings about the performance and situational suitability of the behavior that it monitors. Thus, it must understand under what circumstances that behavior ought to operate and be able to render an opinion on how well that behavior is performing. However, it is not the duty of the Behavior Specialist to Enable/Disable its associated behavior, as that honor is reserved for the Decision Broker.

As an example, the Roadway Navigation Behavior Specialist would monitor the operation and suitability of the Roadway Navigation planning and control component. It would know that global position (lat/long) is required for the component to operate. Should the global positioning component (GPOS) quit working, the Roadway Navigation Behavior Specialist would downgrade its Recommendation and report "GPOS Unavailable" as the reason that the Roadway Navigation component has stopped the vehicle (aside: assuming that it is still in control, the Roadway Navigation component would presumably enter an emergency state when it stops receiving GPOS input, which would cause it to stop the vehicle, after which it would attempt to re-initialize its connection to the GPOS component; however, it would have no way of reporting why all of that happened).

A best practice (not required or enforced by the framework) is to embed each Behavior Specialist into the code base of its associated behavior. This will give it intimate access to the inner workings and internal states and data structures of the component that it is tasked to monitor. This reduces marshalling of that data across components. The loss of modularity (due to coupling) is mitigated if the Behavior Specialist is required to comply with its interface specification as if it were a stand-alone component. That is, it sends and receives messages (or reads from and writes to the knowledge store) in exactly the same way it would if it did not

cohabitate with the behavior. The only exception is for those cases where the Behavior Specialist/behavior pair is the sole producer/consumer of the data being transferred.

Decision Specialist

The Decision Specialist assumes ultimate authority over how the AGV will operate while in autonomous mode. It uses an entity referred to as a Decision Broker to manage this process. The Decision Broker does this by considering the Recommendations and Findings from the other Specialists and applying that information against its own decision-making knowledge to make the final determination of how to proceed. Note that the Decision Specialist may be manifested by a single, centralized Decision Broker or divided into a cohesive collection of Decision Brokers distributed across the system. For example, if a particular behavior itself must choose among several sub-behaviors during execution, a Decision Broker can be tasked with deciding which sub-behavior is desired. To date, there are just seven fundamental types of actions that can be taken by the Decision Broker:

- Monitor a specified behavior (test a value and take action if satisfied)
- Verify a specified behavior (test a value and do nothing if satisfied)
- Enable a specified behavior
- Disable a specified behavior
- Set (maximum) Travel Speed
- Wait for a period of time before retesting/taking action
- Execute another Protocol

Naturally, more primitives can be added as the need arises. However, it is possible to assemble these primitive actions into high-level decision-making Protocols. For example, if the Decision Broker realizes that the vehicle is blocked while under the control of a "Navigation" behavior, it might follow this Protocol:

1. Set Travel Speed = 0 mps (as a precaution, since a blocked vehicle should not be moving)
2. Verify that the Reverse Behavior Specialist believes that it is "OK" to employ the Reverse Behavior

3. Disable the Navigation Behavior (note: assume that it continues to try to find a successful plan even though it is no longer controlling the vehicle)
4. Enable the Reverse Behavior
5. Set Travel Speed = 1.5 mps
6. Monitor the Navigation Behavior for success
7. Monitor the Reverse Behavior for unsafe conditions
8. (assuming Navigation success) Set Travel Speed = 0 mps
9. Disable the Reverse Behavior
10. Enable the Navigation Behavior
11. Set Travel Speed = currently appropriate speed (per another Protocol)
12. Execute high-level monitoring Protocol

This general approach to using Protocols to generate high-level plans and provide intelligent behavior can then be implemented any number of ways, including the plan trees and state tables found in 4D/RCS and possibly the Mission Generator and Mission Spooler under development by the JAUS Working Group. The Reference Implementation discussed in Chapter 4 provides a cohesive set of Protocols; each Protocol was implemented as a distinct C-language function.

Reasoning Mechanism

To support the operational phase, the framework calls for an asynchronous, iterative, forward-chaining reasoning mechanism and control strategy for propagating facts into Findings into Recommendations into executed actions. This means that for a given Specialist, at whatever cycle rate it operates and on whatever processing module it inhabits, its inputs are updated and examined, the algorithm is executed, and its outputs are updated and published. Naturally, the control strategy supports appropriate hysteresis, or dampening, of changes in Findings to avoid thrashing in downstream consumers of those Findings. Even though the strategy is forward

chaining in nature, the implementation may be event-driven by injecting an event handler into the input examination step such that it allows the module to exit if none of its inputs have changed. Also, the notion that Conditions always reset to "Absent" and must be re-proven to remain in effect helps ensure truth maintenance (at least for Conditions).

The framework allows for use of either a centralized repository (e.g., a blackboard or knowledge store) as the source and sink of all information produced/consumed by the Specialists, or a decentralized messaging scheme (e.g., a publish/subscribe model). Further, the framework places no constraints on the method to be used by a given Specialist, and thus, supports a hybrid architecture of various AI and conventional techniques (i.e., a given Specialist could be implemented as an Expert System, a Neural Network, a Bayesian Network, a tree search routine, a linear program, or a purely algorithmic program). Likewise, a given behavior module could be purely reactive, purely deliberative, a hybrid of the two, or something completely new.

Concept of Operation

The operational goal of the Adaptive Planning Framework is to use the elements of the Knowledge Representation Scheme derived during the design phase to produce actionable, high-level decisions at run-time. These decisions, in turn, lead to vehicle behaviors that achieve a mission or a set of goals in light of the current situation. This is accomplished by allowing each Specialist to repetitively apply its rules and algorithms to produce its Findings. The concept of operation at the lowest level then is for each Specialist to gather and analyze inputs and produce results as quickly as possible (nominally targeted as 20 Hertz). These "local" Findings are immediately made available to the entities that need them, possibly for further refinement or in support of a behavioral decision. Thus, the concept of operation at the vehicle level is that data, information, and earlier Findings are transformed into new Findings, which are in turn used to

produce even newer Findings, to enable Behavior Specialists to provide Recommendations, and/or to affect decision-making. This concept is portrayed schematically in Figure 3-2.

Since these Specialists are likely to be executing on different computers, at different iteration rates, there could be instants in time where a Finding used by a Specialist or Protocol is out of date by a fraction of a second. Such latencies and logical "noise" must be dealt with in the formation of the decision-making Protocols.

Transmission of Findings

Data marshalling can operate in one of two ways, depending on the underlying messaging architecture. If it is centralized, each Specialist is tasked with updating its Finding(s) in the knowledge store or blackboard whenever a value changes. Users of Findings are responsible for setting up "on-change" triggers in the knowledge store/blackboard to be given new values whenever the value of a Finding of interest is updated. This change-driven approach reduces both network traffic and component processing demands while maintaining an "arms-length" relationship among the various entities. In other words, there is no need for a producer to know who its consumers are and *vice versa*. While this approach offers simplicity, there is a performance tradeoff. It takes at least three time periods to deliver a new Finding to its consumer, one for the Specialist to send the new Finding to the repository, one for the repository to process it, and one for the repository to send the Finding to those who have signed up for it. This approach would be appropriate for applications whose individual components operate at iteration cycles much higher than that needed to assure sound operational performance.

If a decentralized messaging architecture is used, then a subscription process must take place as each subscriber comes on line. Assuming that the Specialist that publishes the Finding is already up and running, the subscriber would ask the publisher to add them to their list of subscribers for that Finding. If the publisher is not operational, then the subscriber would have

to periodically resend the subscription request. Once the publisher and subscriber have linked up, then the publisher will send updated information to the subscriber directly. This approach shortens the latency to a single time step since a "point-to-point" link has been established. The drawback of this approach is that either *a priori* knowledge of publishers must be known when the subscribing entity is designed, or the concept of operation must include a discovery process that enables subscribers to seek out entities that publish the Findings they need.

Knowledge Representation Tools

In order to standardize the content and the process for representing knowledge in the Adaptive Planning Framework, a collection of knowledge representation tools was devised for defining behaviors, Findings, and Protocols. Each of these tools is discussed in the subsections that follow and are used to define the Reference Implementation discussed in Chapter 4. As an aid while reviewing the design and use of the templates presented below, the reader may wish to examine the fully populated templates found in Appendix C.

Behavior Use Cases

Use Cases are utilized to define each distinct behavior in order to capture and manage the behavioral alternatives available to the Decision Broker. Figure 3-3 shows an empty Behavior Use Case Template to be used for representing a deliberative behavior. The elements of the template each capture a notion vital to the full and unambiguous definition of the behavior. The common name of the behavior should be added to the title of the Use Case to create a unique title (e.g., "Roadway Navigation Behavior Use Case"). The Description field allows the designer of the behavior to convey the duties and goals of the behavior, along with any other background information that may be of importance to designers of other parts of the autonomous system, developers tasked with implementing the subject behavior, or team members asked to conduct a design review. The Assumptions field should contain any assumptions related to the vehicle, its

environment, its operation, etc. that, if not satisfied, would obviate the suitability, stability, and/or safety of the subject behavior. The Constraints field captures the limitations and boundaries of the subject behavior in terms of what it must do or must not do. The Entry Conditions field enumerates items that must be in place before the behavior can take control of the vehicle, such as feeds from other components, confirmed control of other components, vehicle state, etc., whereas the Exit Conditions field enumerates the desired state of the vehicle and the subject behavior when it is being discontinued. Inputs/Outputs enumerate the data, information, Findings, and any other meta data consumed or produced by the subject behavior, respectively.

The heart of the Use Case is the section containing the Steps required to execute the behavior once it is given control of the vehicle. The Steps are presented in a three-column format, with the first column simply numbering the steps to enable direct reference by other steps to support non-serial flow, such as branching and looping. The second column dictates the Action that should be taken at that step and will typically begin with a verb. The third column dictates the Contingency Actions(s) that should be taken in the event that the Action prescribed by the step fails or otherwise cannot be taken. This format allows the nominal flow of the behavior to progress down the middle column while the third column is reserved for off-normal paths. Since some steps are completely internal to the behavior, or present a very low to nonexistent risk of failure, the Contingency Action for a given step may be left blank.

Figure 3-4 shows an empty Behavior Use Case Template to be used for representing a reactive behavior. This type of Use Case is the same as the deliberative flavor except that the notion of a Behavior Model is added. The Steps are directed at basic “housekeeping” duties, such as successfully and safely starting up the behavior, and launching the production of the

Behavior Model. The model follows a three-column approach for conveying the subsumptive nature of the reactive behavior. As such, each possible action available to the behavior is enumerated from highest priority to lowest with the Priority noted in the first column and the Action in the second. The Stimulus that will enable the Action to be executed is captured in the third column. The model is executed such that the highest priority Action whose Stimulus is satisfied is the action that is executed. Special consideration for hysteresis and damping can be indicated such that action thrashing is avoided (e.g., once an action is allowed to execute, it must be allowed to continue for some minimum period of time, assuming that it is safe to do so, even if a higher priority action becomes available). Finally, the last row of every model should contain the action that should be taken when none of the stimuli are present. Note that if the behavior is responsible for generating a control signal (as opposed to a control intent), then preserving signal continuity and addressing drivability would be handled as a separate topic and not modeled as part of the Use Case other than perhaps being noted as an Assumption.

Findings Worksheet

A Findings Worksheet was devised in order to define and manage the various Findings needed for an implementation of the Adaptive Planning Framework. Figure 3-5 shows an empty Findings Worksheet template. The elements of the worksheet each capture a notion vital to the full and unambiguous definition of a Finding. A given Specialist will have one or more Findings and a Finding will have multiple Possible Values and its Type will be a Condition, State, Event, or Recommendation. A Findings Worksheet should be completed for each unique combination of Specialist and Finding slated for an implementation.

The Rule(s)/Algorithm(s) section provides the crux of the definition of the Finding. An Element should be added for each way of determining each Possible Value of the Finding. Any case where more than one Possible Value can be reached must have an Element added that

selects the one, final result for that case. The Comments column allows for entry of remarks that aid in the understanding or implementation of the associated Element. Comments are used for such things as explanation of terms and side effects, notation of configurable parameters (since a configuration change could affect reasoning), and connection to sources of inputs.

The set of Findings Worksheets should be reviewed for cohesiveness, completeness, and ambiguity. To be considered cohesive, all of the Possible Values of all of the Findings should be used by some entity on the vehicle, such as another Finding, a Decision Protocol, or as a direct input to a software component. An exception can be made when a given Possible Value is included for completeness. For example, if "High" and "Low" were Possible Values of a State that were indeed used by other entities, it would be permissible (even desirable) to include "Nominal" even though no other entity ever used it. Conversely, if it turned out that only "High" were being used, then the set of Findings would become more cohesive by converting the State into a Condition whose high-value can be either Present or Absent.

To be considered complete, every Possible Value must have a method for finding it. Recall that the default value for a Condition is "Absent" and that it will always report "Absent" if no rule or algorithm evaluates to "Present." This truth maintenance strategy, while implicit, still qualifies as a method for determining "Absent" and thus, is sufficient when assessing the completeness of a Condition. Further, every data element or Finding value used as an input to a rule or algorithm must exist, be determinable, and be available to the Specialist executing the rule or algorithm.

To be considered free from ambiguity, the collection of rules and algorithms that produce the Possible Values of a Finding must always produce a single result. For any case wherein more than one result could be produced, additional logic must be added in order to choose the

single, final result. It is permissible for there to be multiple ways to reach the same result, as long as there is a valid, situational reason for the added complexity (this style of reasoning is usually equivalent to establishing an "OR" relationship between the multiple paths to the same result).

If the set of Findings is cohesive, complete, and free from ambiguity, there will be a continuous, distinct, mappable chain from the raw data and information used by the Findings, through the set of Findings, and out to the ultimate consumers of the Findings.

Decision Broker Protocol Worksheet

Protocol Worksheets are used to define each of the distinct actions that could be taken by the Decision Broker. There will typically be a pair of Protocols for each behavior, one for transitioning to it and one for transitioning out of it. In addition, a single, "executive" Protocol is needed for monitoring and orchestrating the behavior selection process. Figure 3-6 shows an empty Decision Broker Protocol Worksheet. The elements of the template each capture a notion vital to the full and unambiguous definition of the Protocol. A unique Name should be selected for the Protocol and its Goal field should convey the intent and purpose of the Protocol, along with any other background information that may be of importance to designers of other parts of the autonomous system, developers tasked with implementing the subject Protocol, or team members asked to conduct a design review. The Assumptions field should contain any assumptions that, if not satisfied, would obviate the subject Protocol. Any data, information, Findings, or any other meta data needed by the Protocol should be added to the Input Parameters field. The Entry Conditions field enumerates items that must be in place before the Protocol can begin execution, such as feeds from other components, confirmed control of other components, vehicle state, etc., whereas the Exit Conditions field enumerates the desired state of the vehicle and the subject Protocol when exiting it. Since Protocols often deal with waiting times and

speed parameters, the worksheet provides for a default Wait State Timeout period, which indicates how long a standard Wait action should be, and a default Speed Tolerance, which indicates how exact a speed threshold must be (especially vital when the Protocol calls for a velocity of 0 mps).

As with the Use Cases, the heart of the Protocol is the section containing the Action Steps and Contingency Steps. However, the entries for these steps should be limited to one of the defined fundamental action types that were discussed earlier in this Chapter (there are currently seven of them). The sequencing of the Action Steps provides a script for the nominal path for achieving the Goal and Exit Conditions. The Contingency Steps provide a script for dealing with off-normal conditions and should be invoked when their associated Action Step fails or cannot be taken. If an Action Step has little or no risk of failure, then its Contingency Step may be left blank.

Foundational Research

During the course of the research, several intermediate prototypes were developed. These efforts served to shape the research results presented here. The most comprehensive of these intermediate efforts was a Proof of Concept prototype of the framework that helped to clarify and validate the idea. Although quite limited in scope, it served its purpose well and became the springboard for the subsequent work. Appendix A contains a detailed presentation of this prototype.

The NAVIGATOR vehicle built for DARPA Grand Challenge 2005 carried on it an embryonic implementation of the Adaptive Planning Framework, focusing on a pair of Situation Assessment Specialists delivering a handful of Findings to enable the Decision Broker to set the maximum speed of the vehicle. Appendix B contains a detailed presentation of this early implementation.

Table 3-1. Example Assignment of Situation Assessment Specialists.

SA Category	Typical SA Specialist	Typical Situational Finding
Mission	Mission Goal Specialist	Mission Goal State {Behind | Nominal | Ahead}
	Mission Progress Specialist	Mission Status {Waiting | In-Progress | Failed | Complete} Mission Type {Seek-goal | Wander | Cover-Area}
Plan Segment	Boundary Specialist	Vehicle State {In Bounds | In Fringe | Out of Bounds}
	Plan Element Specialist	Plan Segment Status {Waiting | In-progress | Complete} Plan Segment Type {Navigate | Park | Retrieve-item}
Mobility	Mobility Specialist	Mobility State {Operational | Stuck | Blocked} Mobility Type {Cruising | Creeping | Waiting}
Roadway	Terrain Specialist	Terrain State {Smooth | Rugged | Very Rugged}
	Roadway Law Specialist	Legal to Pass Condition {Present | Absent}
	Roadway Convention Specialist	Appropriate to Pass Condition {Present | Absent}
Intersection	Intersection Specialist	Intersection-Clear Event {True @ timestamp | False} Intersection Type {Right-of-way | 2-way | 3-way | 4-way}
Obstacles	Close Range Safety Specialist	Close Range Left-Side-Safe Condition {Present | Absent} Forward-Left-Safe Condition {Present | Absent}
	Long Range Safety Specialist	Long Range Obstacle Condition {Present | Absent}

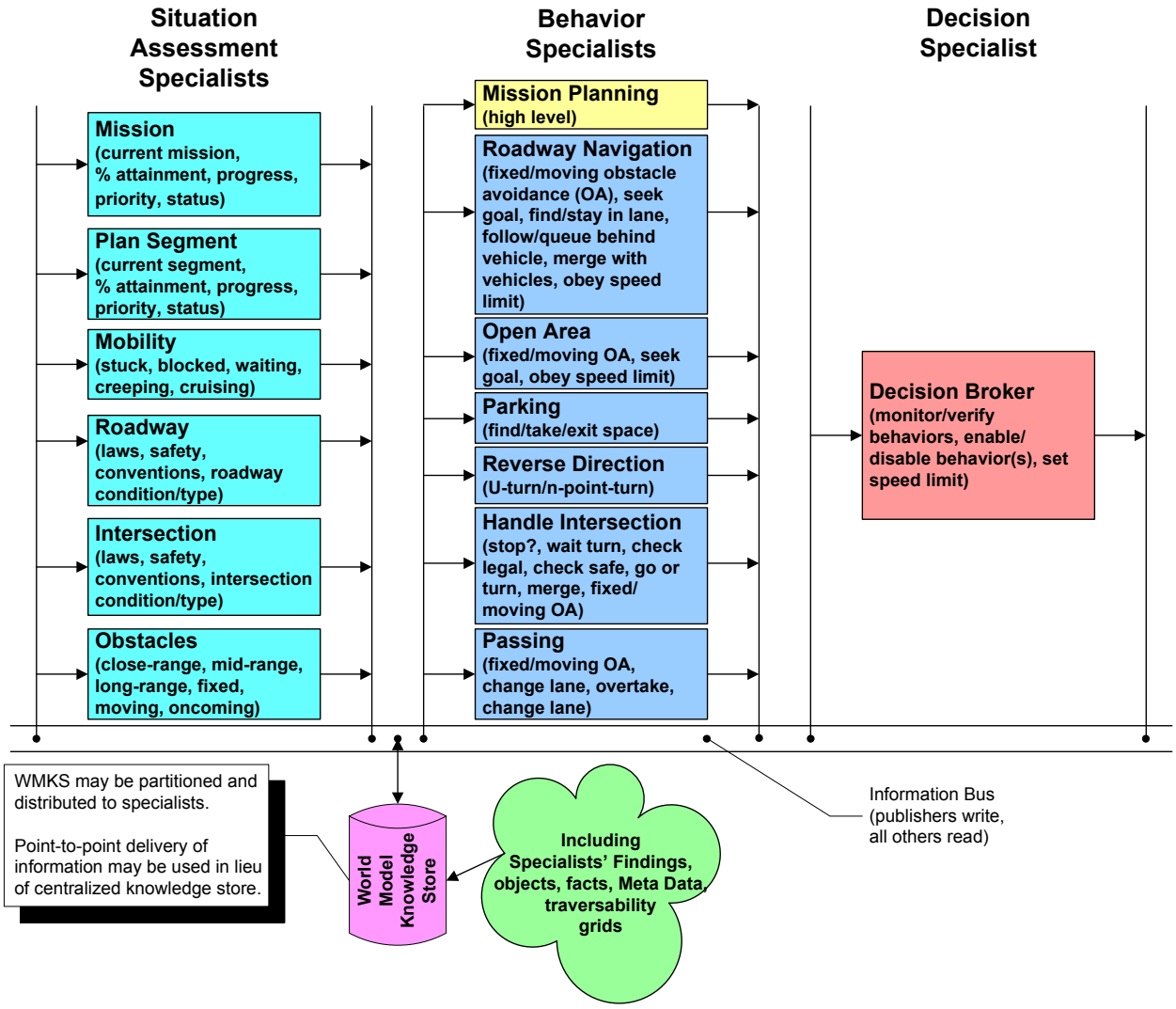


Figure 3-1. Adaptive Planning Framework Conceptual Model showing Representative Examples.

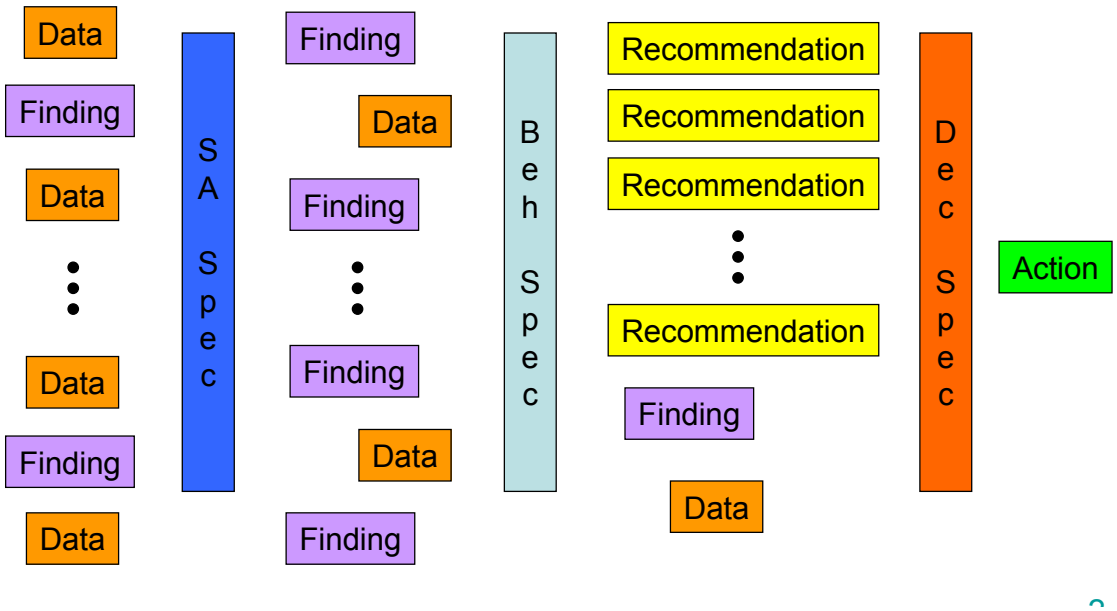


Figure 3-2. Schematic portrayal of Adaptive Planning Framework Concept of Operation.

Behavior Use Case Template
(Deliberative)

Scenario Description:

Assumptions:

Constraints:

Entry Conditions:

Exit Conditions:

Inputs Consumed:

Outputs Produced:

Steps for Deliberative Behavior:

Step #	Action	Contingency Action
1	Action to take for 1st Step	Action to take if 1st Step fails
2	Action to take for 2nd Step	Action to take if 2nd Step fails
3	…	…

Figure 3-3. Use Case Template for Deliberative Behaviors.

Behavior Use Case Template
(Reactive)

Scenario Description:

Assumptions:

Constraints:

Entry Conditions:

Exit Conditions:

Inputs Consumed:

Outputs Produced:

Steps for Reactive Behavior:

Step #	Action	Contingency Action
1	Action to take for 1st Step	Action to take if 1st Step fails
2	Action to take for 2nd Step	Action to take if 2nd Step fails
3	Apply Reactive Behavior Model	

Reactive Behavior Model:

Priority	Action	Stimulus
1	Action to take	while Stimulus 1 is True
2	Action to take	while Stimulus 2 is True
…	…	…
Last	Action to take when no stimuli are true	Monitor for any available Action

Figure 3-4. Use Case Template for Reactive Behaviors.

Findings Worksheet

Specialist:

Finding: **Type:**

Possible Values:

Rule(s)/Algorithm(s):

Element	Comments
Rule/Algorithm for finding 1st Possible Value	
[optional Rule/Algorithm for alternate ways of finding 1st Possible Value]	
Rule/Algorithm for finding 2nd Possible Value	
…	

Figure 3-5. Findings Worksheet Template.

Decision Broker Protocol Worksheet

Name of Protocol:

Goal of Protocol:

Assumption(s):

Input Parameter(s):

Entry Conditions:

Exit Conditions:

Wait State Timeout:

Travel Speed Tolerance:

Protocol:

Action Steps	Contingency Steps
1. 1st Step	1. Contingency if 1st Step fails
2. 2nd Step	2. Contingency if 2nd Step fails
3. …	3. …

Figure 3-6. Decision Protocol Template.

CHAPTER 4
REFERENCE IMPLEMENTATION AND FIELD TESTING

This chapter describes the fielding of the Adaptive Planning Framework as a foundation technology for the Team Gator Nation entry for DARPA's Urban Challenge. The architecture for the Urban Challenge version of the NAVIGATOR includes extensive adoption of the Adaptive Planning Framework and this section presents how the framework described in Chapter 3 was reduced to practice on an operational Autonomous Ground Vehicle. Above all else, the framework as presented in Chapter 3 benefited from the stress and refinement opportunities provided by this exercise.

Reference Implementation Architecture and Design

The initial Milestone for Team Gator Nation was to achieve the autonomous selection and switching between unique behaviors in a JAUS-compliant fashion. This goal was used to craft the architecture and design for the Reference Implementation. The resulting architecture is shown in Figure 4-1, depicting this two-behavior system. Appendix C contains a set of documents, based on the Knowledge Representation Tools discussed in Chapter 3, that define the Adaptive Planning Framework Reference Implementation.

Behaviors Identified

The two behaviors chosen for implementation were basic Roadway Navigation (RN) and an n-Point Turn (NPT). The RN is a deliberative behavior evolved from the Receding Horizon Planner that was used in the DARPA Grand Challenge 2005 (Crane et al. 2006). It receives a goal waypoint and a tessellated traversability grid and then uses an A* search algorithm to find the lowest cost path from the current vehicle position to the goal. The instantaneous steering effort needed to follow that path is then sent to the JAUS Primitive Driver (PD) component. This entire process is performed iteratively at approximately 20 Hertz. Modifications were made

to the Receding Horizon component design to incorporate the Adaptive Planning Framework infrastructure, to enable it to be controlled by the JAUS Subsystem Commander component, and to enable it to interactively take and release control of the PD. The "Roadway Navigation Behavior Use Case" included in Appendix C provides additional insight into the operation of the RN behavior.

The NPT behavior is reactive in nature and, thus, has no planning or searching element. It follows a hierarchically organized set of actions, each triggered by a specific positive circumstance (provided by the "Close Range Safety Specialist" discussed in the next subsection). The NPT behavior will execute the highest priority action whose enabling circumstance is valid. Its basic operation is to drive forward in a full-left turn; if that motion is (or becomes) blocked by an obstacle or the edge of the road, then it begins to drive backward in a full-right turn. If that motion is (or becomes) blocked, then it begins to drive straight backward. If all three potential actions are unavailable, it causes the vehicle to sit motionless, waiting for any one of the actions to be available. The NPT behavior will apply this strategy ~20 times per second until the Decision Broker places it into Standby state. Naturally, the NPT behavior must ensure that the vehicle is stationary whenever it attempts to shift from forward to reverse gear or vice versa. The "n-Point Turn Behavior Use Case" included in Appendix C provides additional insight into the operation of the NPT behavior.

Specialists and Findings Identified

Three Specialists were identified for the Reference Implementation, the Roadway Navigation Behavior Specialist, the n-Point Turn Behavior Specialist, and the Close Range Safety Specialist. The paragraphs that follow describe each of the Specialists and the Findings for which they are responsible. The "Findings Worksheets" included in Appendix C provide additional insight into these Specialists and their Findings.

The Roadway Navigation Behavior Specialist is tasked with monitoring the operation of the RN behavior and is responsible for three Findings. The first is the rnPlanningState which reports whether the RN's intrinsic planner has been successful in finding a valid path ("Succeeded") or has reached its final waypoint ("Goal Achieved"). If neither of these cases applies, then the Finding is reported as "Failed." Similarly, the rnMobilityState reports whether the vehicle is able to execute its plan ("Operational"). If not, it further determines whether the non-operational state is due to an obstacle ("Blocked") or some other situation ("Stuck"). Finally, the Roadway Navigation Behavior Specialist uses its other Findings to determine the overall suitability of the RN behavior in terms of whether the rnRecommendation is "OK," "Faulted," or "Need New Plan." Note that the first two Findings use various internal states of the RN behavior as their inputs while the third Finding is based solely on the Findings of the other two.

The n-Point Turn Behavior Specialist has just one Finding and it is based primarily on the Findings of the Close Range Safety Specialist. If any one of the three input Conditions is Present, the nPTRecommendation is found to be "OK." Otherwise, the n-Point Turn Behavior Specialist considers additional circumstances to render its opinion as whether the behavior is "Waiting," "Blocked," or "Unsafe." Comparing these two Behavior Specialists highlights how Findings can be based on a variety of combinations of internal and external states.

The Close Range Safety Specialist uses preprocessed LADAR range data to determine three Findings: forwardLeftSafeCondition, reverseRightSafeCondition, and reverseStraightSafeCondition. Each of these Findings is associated with one of the Reactive Actions delineated in the n-Point Turn Use Case. All three are in the form of Conditions, which means that there must be conclusive evidence that they are "Present"; otherwise, they will be

deemed "Absent." If the evidence is missing or unavailable, then the Condition is deemed to be "Unknown." The range data is reported in terms of three sectors in front of and three sectors behind the vehicle (the sector angles are configurable, but must add up to 180 degrees). The range data in each sector is compared to a configurable safe buffer distance for that sector and the targeted Reactive Action. If the range data for a given sector is greater than its associated buffer, then that sector is safe. If all three sectors are determined to be safe, then the relevant Safe Condition is reported as "Present." This design is portrayed in Figure 4-2.

Decision Protocols Identified

Five Decision Broker Protocols were identified for the Reference Implementation, one that provides the overarching monitoring of behaviors and invocation of other Protocols and two pairs that transition into and out of the two available behaviors. The paragraphs that follow describe each of the Protocols, with more detailed information provided in the "Decision Broker Protocol Worksheets" included in Appendix C.

The Monitor/Select Behavior Protocol assesses the suitability of each available behavior and selects the behavior that is to control the operation of the vehicle. Thus, this Protocol is executed in every cycle of the Subsystem Commander (SSC) component and could be considered as an Executive Protocol. It provides the essence of the Decision Broker's functionality. For this Reference Implementation, the RN behavior is always preferred if it is available and safe, causing the nature of this Protocol to be one of selecting the nPT behavior by exception. The remainder of its job is to methodically transition into and out of the RN and nPT behaviors as appropriate and to stimulate the creation of new path plans when needed.

Should the RN behavior be blocked for a reasonable period (e.g., long enough for a temporary obstruction to clear itself), this Protocol can spawn a request for the Mission Planner to create a new path plan from the vehicle's current position to the goal. A special case is

encountered when the vehicle reaches its destination, spawning a request for the Mission Planner to create an entirely new plan. Note that in either case, the autonomous Mission Planner component has not been designed or implemented, so there is not yet a way to devise a RePlan Current Mission Protocol or a Plan New Mission Protocol; for now, new path plans are generated using a manually operated planner.

The protocols for transitioning into and out of behaviors are devised such that they are mutually exclusive and can be set into motion in parallel with the continued operation of the Monitor/Select Behavior Protocol. In other words, the Monitor/Select Behavior Protocol continues to monitor the situation even while it attempts to place a given behavior into its Ready State or Standby State. In keeping with this design pattern, note that each of the transitional protocols is exited when its associated behavior has achieved the commanded state.

The Transition to Roadway Navigation Behavior Protocol and the Transition to n-Point Turn Behavior Protocol each ensure that the vehicle is stationary and that its associated Behavior Specialist still recommends its use. It then calls for a Resume message to be sent to the behavior component. Once it verifies that the behavior component is indeed in the Ready State, the Protocol is exited.

Similarly, the Exit from Roadway Navigation Behavior Protocol and the Exit from n-Point Turn Behavior Protocol each ensure that the vehicle is stationary and then calls for a Standby message to be sent to the behavior component. Once it verifies that the behavior component is indeed in the Standby State, the Protocol is exited.

Reference Implementation Messaging Design

In order for a Specialist to publish its Findings to its subscribers, a JAUS-compatible messaging mechanism was needed. This was accomplished by introducing the concept of Meta Data and incorporating a set of messages and supporting data structures and utility functions into

the CIMAR JAUS library. The requirements for this design were captured by the author in a JAUS Interface Control Document, which has been reproduced as Appendix D.

Although the driver for a Meta Data implementation for the Adaptive Planning Framework was the transmission of Findings, this implementation addresses broader team needs to marshal other types of information and data among the various components. Specifically, the Meta Data message set described here can be used for any data that needs to be transmitted from one component to another. However, its intended use is for data not already included in an existing JAUS message.

JAUS-based Meta Data Message Set

The decision was made to use the publish/subscribe design pattern (as opposed to a centralized knowledge store) because this was more in keeping with how other repetitive information is distributed in JAUS (referred to as JAUS Service Connections). Thus, in addition to a message for transmitting the Meta Data ("Report Meta Data"), two additional "subscription" messages were required ("Meta Data Changed Event Setup" and "Meta Data Changed Event Confirmation"). The setup message is sent by the "subscriber" component to the "publisher" component asking it to start (or stop) sending its Meta Data. The publisher then adds the subscriber to its list of components to which it sends Meta Data and replies to the subscriber with the confirmation message. From that point forward, the publisher compiles a Report Meta Data message whenever its Meta Data has changed significantly (as determined by the designer of the publishing component) and sends it to all of the components on its subscriber list.

The Report Meta Data message was crafted to be powerful and flexible, which also required its design to be rather complex. Specifically, the message had to be designed to package a flexible number of Meta Data Elements and to accommodate an assortment of valid JAUS data types. The number of data elements is handled by using the first field in the message

to indicate the number of Meta Data Elements to expect in the remainder of the message. Each Meta Data Element requires four fields to fully convey its current information, so the first field tells the message parser or packer how many sets of four to process.

The issue of flexible data types contained within the message was addressed by extending the notion of JAUS Type Codes that was introduced by the JAUS Working Group as part of a series of multi-organizational experiments. One of the goals of these experiments was to enable payload components to autonomously disclose to a third-party (arm's length) Operator Control Unit how to display information from and send commands to it. This led the author to develop the Variant type for use in a JAUS message. With this approach, the current value of a Meta Data Element is conveyed via two fields in the message: the Data Type Code field that uses a single byte to enumerate which of the defined Type Codes applies to the data value that is to follow, and the Value field, whose data type is Variant, indicating that the Data Type Code field must be referenced in order to determine its true type and, thus, its field size. This technique allows for a Report Meta Data message to contain data of any permissible type, arranged in any combination, and assembled extemporaneously by the publishing component and correctly parsed by the subscriber.

In addition to the Data Type Code/Value pair, each Meta Data Element also includes its Name as a NULL-terminated string and a Time Stamp as an unsigned integer whose bit field interpretation is prescribed by the JAUS Reference Architecture. One restriction on the use of the Meta Data message set is that, by agreement among the component designers (i.e., there is no enforcement in the software), the component ID combined with the Meta Data Element name must be unique within the domain or namespace. While there are schemas and approaches for automating this constraint, such were not pursued for this implementation.

Extensions to CIMAR Messaging Infrastructure

The CIMAR JAUS library (libJausC) is a set of C-language source and header files that provides the JAUS-compliant infrastructure used throughout the lab. In order to support the messages described in the previous section, this infrastructure had to be extended to incorporate the notions of Meta Data and the Variant data type, as well as to manifest the new messages themselves. To that end, the Reference Implementation required the design, development, and testing of five software modules within libJausC (jausVariant.c/h, jausMetaData.c/h, metaDataChangedEventSetupMessage.c/h, metaDataChangedEventConfirmationMessage.c/h, and reportMetaDataMessage.c/h).

The jausVariant type was introduced by first defining the allowable JAUS Type Codes to be implemented. The enumeration published by the JAUS Working Group (JAUS-OPC 2005) to support its interoperability experimentation was reviewed and adopted. Table 4-1 lists the data types defined for JausVariant and indicates which of these were fully implemented for the Reference Implementation. Next, a JausVariant data structure was devised that encapsulates both the Type Code and the appropriately typed data value (using a C union of all of the valid data types). The String data type is a special case in that the data stored in the stringValue element of the structure is actually a pointer to the string rather than the string itself. The structure element name and true data type are also indicated in Table 4-1. Finally, three utility functions were incorporated to support the use of the jausVariant data type in JAUS messages. newJausVariant simply creates a new, empty JausVariant structure and returns it to the calling function. jausVariantToBuffer packs up an existing JausVariant structure into a serialized byte stream ready for use in a JAUS message. jausVariantFromBuffer parses a serialized byte stream extracted from a JAUS message and uses it to populate a JausVariant structure.

Next, it was necessary to define two more data structures: JausMetaDataElement for representing a Meta Data Element and JausMetaData for holding a collection of Meta Data Elements. A Meta Data Element represents a single unit of Meta Data (for purposes of the Adaptive Planning Framework, this is equivalent to saying that a Meta Data Element represents a distinct Finding of a distinct Specialist). JausMetaDataElement contains the elements necessary to support the Report Meta Data message (metaDataName, timeStamp, and elementData), as well as additional elements to facilitate the management of the Meta Data (componentId and changedFlag). Since the elementData is a jausVariant data type, it will contain both the typeCode and Value needed by the Report Meta Data message. The componentId is populated with the assigned JAUS ID of the component that houses the Specialist that produces the Finding named in the metaDataName element. This allows the utility functions to examine the metaDataName in combination with the componentId to create a unique key to the Meta Data Element. For example, if two Smart Sensor components both produced a Finding with the same metaDataName, their componentId would provide a way to differentiate the two Findings. Finally, the changedFlag is included mainly for the benefit of the publisher of the Meta Data and is set by whatever algorithm is used by the component to determine that a significant (and therefore reportable) change has occurred. This flag is then used to trigger the production and distribution of the Report Meta Data Message, after which the flag is cleared.

Since components will likely have to support multiple Meta Data Elements, the notion of a Meta Data Element collection evolved and a Meta Data structure was devised. JausMetaData has just two elements, one is a vector of pointers to Meta Data Elements and the other is collection-level changedFlag. If the changedFlag of any of its members is set, then this flag will also be set and all flags will be cleared each time a Report Meta Data message is sent.

The Meta Data utilities include constructors and destructors for Meta Data Elements and the Meta Data collection, functions to set and clear the changedFlag at both levels, and functions to set and get the value of the time stamp element. Three, more complex functions are available to manage the Meta Data Elements themselves. jausAddMetaDataElement creates a new Meta Data Element structure, adds it to a collection, and returns the pointer to the element. jausCopyMetaDataElement creates a new Meta Data Element structure, copies the data contained in the source element, adds the new element to a collection, and returns the pointer to the new element. jausGetMetaDataElement returns the pointer to the element that matches the given metaDataName and componentId within the given collection.

With these structures and utility functions in place, the structures and functions for handling the actual messages can be described. As for all JAUS messages handled in libJausC, there is the standard slate of functions for packing and unpacking these messages, as well as constructors and destructors for the message structures. The only thing needed to extend a generic message into the three needed Meta Data messages is to insert the fields defined in the tables found near the end of Appendix D. The metaDataChangedEventSetupMessage structure has an additional setupFlag field and the metaDataChangedEventConfirmationMessage has an additional confirmationFlag field. The reportMetaDataMessage has two additional fields: numberMetaDataElements and jausMessageMetaDataCollection. The latter represents a complete set of Meta Data Elements and the former tells the software how many elements to expect in the collection.

Reference Implementation Development

This section discusses how the Reference Implementation architecture and design was implemented in software. It is important to note that much of this work was performed by fellow CIMAR graduate students. The strategy was to use NAVIGATOR, as-built for the 2005 DARPA

Grand Challenge and enhanced for JAUS OPC 3.0 experiments, as the starting point and baseline for the development effort. This led to a need to modify some of the NAVIGATOR components and to create some new ones.

Modifications to Existing NAVIGATOR Components

Since the majority of functionality needed for the Roadway Navigation behavior was available in the Receding Horizon Planner, a handful of modifications was sufficient to implement the basics of the Roadway Navigation component. The notion of having the Subsystem Commander component place it into the Standby and Ready states was incorporated, the code for taking and releasing control of the Primitive Driver component was made more flexible (i.e., it is not an error condition to lose control of the Primitive Driver) and relocated, the ability to autonomously change gears was added, and the notion of "nudging" when the vehicle had become blocked for an extended period of time (a behavior of last resort) was removed. Then, the Roadway Navigation Behavior Specialist was added to the RN state machine as a function call (processOutputMetaData) which, in turn, manages its three Findings and the processing of its Meta Data. Finally, the software needed to enable the RN to accept subscribers and publish Meta Data to them was added. As a convenience during testing, a keyboard-based toggle feature was added to allow a test engineer to introduce a simulated obstacle. This feature enabled one to execute a test that includes responding to a blocked roadway without having to physically block the road.

The Primitive Driver component also required modification,[1] even though it did not need to handle Meta Data or play a direct role in the Adaptive Planning Framework by supporting Specialists. The PD software was modified to respond appropriately when multiple behaviors

[1] Eric Thorn was the Lead for this effort

were attempting to control it and to tolerate a brief time span where no behavior was controlling it (to accommodate the switchover, there will be at least one iteration where the previously controlling behavior has released control and the target controlling behavior is attempting to take control). Also, the ability to autonomously shift gears and to process the Discrete Devices JAUS message set was added.

Creation of New Components

The NPT behavior was created as a completely new component.[2] The RN component was used as an *ad hoc* template and portions of its PID controller functionality were reused, which provided a degree of consistency between the two behaviors. Nonetheless, the bulk of the NPT Behavior Use Case had to be implemented in code from scratch. As must be done for all behaviors, the ability to be controlled by and respond to the SSC, to control the PD when appropriate, and to set up and process Meta Data were all incorporated. Since the NPT Behavior Specialist uses Findings as stimuli and also publishes its own Findings, the NPT component was implemented to perform as both a subscriber and a publisher of Meta Data.

The Subsystem Commander originally was targeted for the DARPA Grand Challenge 2005 architecture, but after several months of testing, it was decided to move its functionality to cohabitate with the Planar LADAR Smart Sensor (PLSS). This move was necessary because the Meta Data infrastructure had not yet been invented and some of the data produced by the PLSS was needed by the Specialists but were not included in any JAUS message. However, this early implementation of the SSC was resurrected and extended to accommodate Meta Data and to play the role of both publisher and subscriber of Findings. It also had to be extended to take control of the RN and NPT behaviors and given the ability to place them into the Standby and Ready

[2] Greg Garcia was the Lead for this effort

states. The most significant change was the implementation of the Decision Broker and its Protocols. Each Protocol was implemented as a C-language function (sscSelectBehavior(), sscResumeRN(), sscPauseRN(), sscResumeNPT(), and sscPauseNPT()). The logic and use of these five functions were orchestrated such that each Protocol delivers its intended action individually and that they work together to deliver the intended action as a group.

The Close Range Safety Specialist was originally targeted to cohabitate with an expanded version of the PLSS that could perceive finer details, such as curbs, and would have visibility behind the vehicle. However, scheduling conflicts prevented having this component operational in time to support this work. Instead, the ability to simulate the Findings of the Close Range Safety Specialist was added to the SSC. Keyboard events on the computer attached to the SSC process are used to manually toggle the reported values of the three Conditions. This feature allows a test engineer to press a key whenever he or she wants to change the action taken by the NPT behavior. This intervention is completely at the discretion of the test engineer, but it is anticipated that it would be conducted in fashion that simulates what the LADAR sensors would have seen and, thus, what the Close Range Safety Specialist would have reported.

Field Testing

With the NAVIGATOR vehicle in good working order and updated software in place, field testing could begin. This section describes the test plans and results.

Test Plans

Although more complex scenarios and path/blockage geometries can be conceived, all comprehensive tests would follow this basic outline:

1. Set the vehicle in motion along a planned plan using the RN behavior
2. Contrive a blockage along the path that cannot be overcome (real or simulated)
3. Give the RN a new path that requires a reversal of direction

4. Ensure that the location of the blockage and the geometry of the roadway prevent the RN from simply planning a U-turn
5. Observe that the SSC switches control from the RN to the NPT behavior
6. Use the keyboard to simulate the Findings of the Close Range Safety Specialist, which in turn, stimulate the n-Point Turn actions
7. Observe that the n-Point Turn actions are appropriate
8. Observe that, once the NPT component has sufficiently reversed the direction of the vehicle, the SSC autonomously switches control back to the RN behavior
9. Observe that the RN begins following the new path

Naturally, each behavior must be tested and tuned independently before a comprehensive test can be executed. Independent testing typically requires only a subset of the outline above.

Much of the early testing was accomplished with the vehicle up on blocks in the CIMAR Lab. As part of the 2005 DARPA Grand Challenge testing, the team had created a component that simulates two key JAUS components: the Global Pose Sensor (GPOS) and the Velocity State Sensor (VSS). This enables the testing of components whose operation requires JAUS messages from these two components in order to work properly (or even enter the Ready state at all). As new functionality became available, it could usually be tested in the Lab using this technique. Even though the vehicle never actually moves forward, the simulated position and velocity messages report that it has. Because the front wheels of the vehicle are safely up off the ground and the engine is not running, the commands to the steering, brake, throttle and shifter and resulting actions can be observed (for example, when the NPT behavior commands the PD to execute the reverse, full-right-turn action, one can observe the brake pedal going to the full down position, the gear shift moving into Reverse, the steering wheel rotating to the full-right position, the brake coming up, and the throttle going down).

During this phase of testing, the test plans were *ad hoc* in nature, dictated by the specific needs of the new/changed software being introduced by the developer. In some cases, all that was needed was to exercise and prove one subtask of the operation, such as that one component could take control of another. In other cases, exercising and proving a series of several steps would be required to accomplish the testing goal, such as changing gears (take control, verify the vehicle is stationary, command the gear change, and verify the gear has changed).

In many cases the testing process took advantage of a feature of the libJausC infrastructure called cDebug. The cDebug function provides an indexed printing capability and can be added anywhere in the source code where a printf() function is allowed. The ability to tag a print statement with an index means that the print statements can be associated with each other by topic across multiple functions and processes and the output can be filtered accordingly. This feature also allows the test engineer to log the printf results, display them on the screen, or both. Examples of such logs are included in Appendix E.

Field testing took place at the UF Energy Research and Education Park (often referred to as the Solar Park), the Road Course at the Gainesville Raceway, and the UF IFAS Research Farm near Citra, Florida. The venue at the Solar Park is all grass, so roadways exist only in terms of the series of waypoints and corridor widths established in the NAVIGATOR software files. The main reason for testing at the Solar Park is its proximity to the Lab, thus providing ease of logistics. The Road Course has a network of paved roadways that are more like what would be encountered on real roadways; however, they are not painted and do not have curbs, so the perception aspects of testing the n-Point Turn behavior still require external assistance. The Citra facility has both open, grassy areas and graded roads, but again, no strong features to

demark the edge of the roadways. Unfortunately, a completely realistic testing venue that was safe and legal was never found.

Test Results

Object 4-1 links to a movie made during field testing of the Adaptive Planning Framework on the NAVIGATOR at the UF Citra facility on October 23, 2006. This particular run coincides with the logs contained in Appendix E. The Test Plan for this run was contrived to need only one *a priori* path plan, since the ability to automatically swap out such plans was never implemented. This method of testing enables a continuous autonomous flow between behaviors, thus avoiding the need to pause the test while manual interventions take place. The technique follows these basic steps:

- Set the vehicle in motion on the planned path (in this case, a simple, narrow, straight corridor); since this is the nominal case, the Decision Broker will select the RN behavior to control the vehicle
- After 20 meters or so of travel, artificially block the vehicle so that the SSC will direct the NPT behavior to take over
- Once the vehicle has reached an approximate right angle to the corridor, the artificial blockage can be removed since the RN behavior will not be able to find a success path due to the geometry of the narrow corridor
- As the NPT behavior continues to rotate the vehicle, it will eventually come back close enough to the original heading for the RN behavior to find a successful plan
- At this point, the Decision Broker realizes that the NPT behavior should be placed into standby and the RN behavior resumed
- The vehicle continues on the end of the corridor

A software tool, dubbed the Adaptive Planning Framework – Test Control Unit, was developed to support field testing of an AGV. It allows a test engineer to describe the setup and step-by-step instructions of a test in a structured xml file based on a test plan template, and then displays that test in an interactive Graphic User Interface. The Citra Test Run outlined above is

more fully described in the series of screen shots taken from the Test Control Unit shown in Figure 4-3 and Figure 4-4.

Several key concepts were demonstrated and their viability confirmed during the course of testing the Adaptive Planning Framework Reference Implementation:

- The notion of a Decision Broker interactively and autonomously orchestrating the behavior of a complex, full-scale AGV
- The notion of Specialists, implemented as software entities, autonomously determining, using, and exchanging their Findings
- The use of the Meta Data representation and transfer mechanism to enable the storage and exchange of Findings
- A hybrid of both deliberative and reactive behaviors cooperating to pursue a mission
- The use of a granular, distributed knowledge representation scheme
- The use of a granular, distributed reasoning mechanism operating in near-real-time

Object 4-1. Video of Successful Adaptive Planning Framework Test Run at UF's Citra Facility 10/23/2006 (97 MB, citra_composite.mpg, 124 seconds).

Table 4-1. JAUS Type Codes Supported by JausVariant Data Type.

Enumeration Value	JAUS Type Code	JausVariant Element Name	C-language Data Type	Remarks
1	Short	shortValue	short	
2	Integer	integerValue	Int	
3	Long	longValue	long	
4	Byte	byteValue	unsigned char	
5	Unsigned Short	uShortValue	unsigned short	
6	Unsigned Integer	uIntegerValue	unsigned int	
7	Unsigned Long	uLongValue	unsigned long	
8	Float	floatValue	float	
9	Double	longFloatValue	double	
10	Scaled Unsigned Byte	longFloatValue	double	Scaled values are stored as doubles typically [+]
11	Scaled Short	longFloatValue	double	Scaled values are stored as doubles typically [+]
12	Scaled Unsigned Short	longFloatValue	double	Scaled values are stored as doubles typically [+]
13	Scaled Integer	longFloatValue	double	Scaled values are stored as doubles typically [+]
14	Scaled Unsigned Integer	longFloatValue	double	Scaled values are stored as doubles typically [+]
15	Scaled Long	longFloatValue	double	Scaled values are stored as doubles typically [+]
16	Scaled Unsigned Long	longFloatValue	double	Scaled values are stored as doubles typically [+]
17	Enumeration	enumValue	unsigned short	Indexes into a previously stored comma-delimited string[+]
18	Boolean	booleanValue	enum	Either TRUE or FALSE[+]
19	String	stringValue	char *	NULL-terminated
20	Unsigned Byte Tuple	uByteTuple	unsigned char	Two of them in a struct
21	Unsigned Short Tuple	uShortTuple	unsigned short	Two of them in a struct
22	Unsigned Integer Tuple	uIntegerTuple	unsigned int	Two of them in a struct

[+] Not implemented in this release

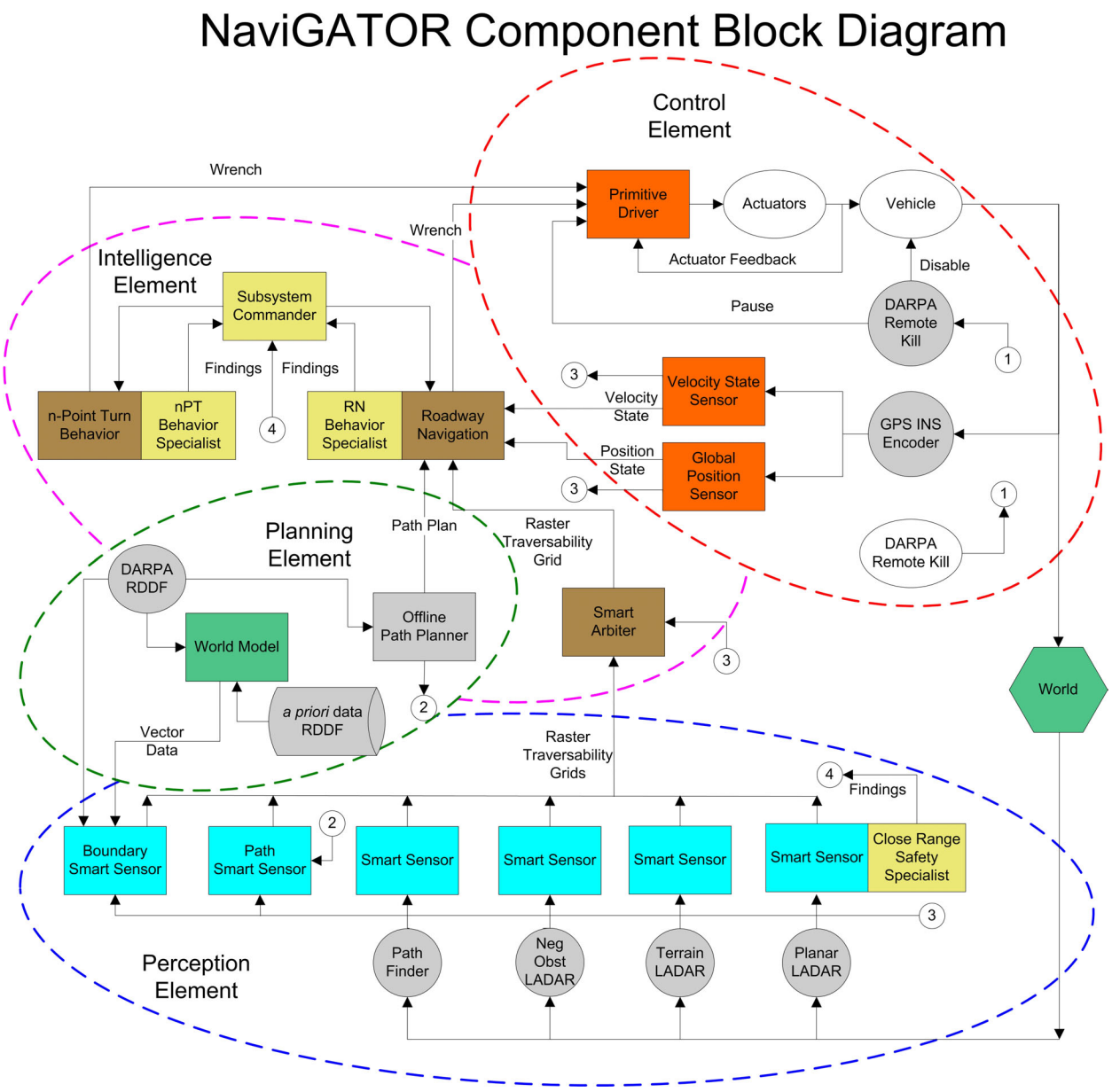


Figure 4-1. Simplified NAVIGATOR Architecture for a Two-Behavior system.

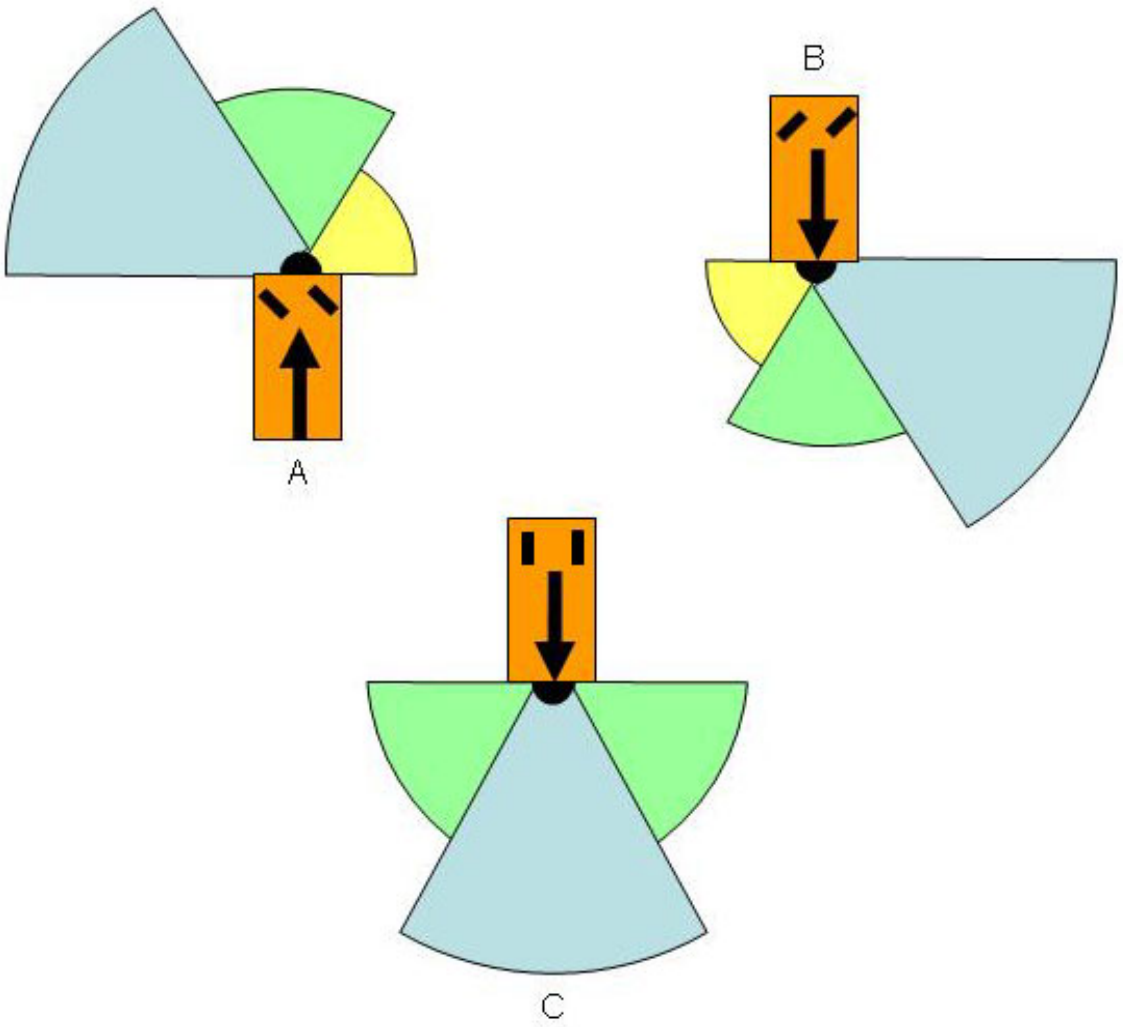


Figure 4-2. Portrayal of Safety Buffers for the Three n-Point Turn Reactive Actions. A) forward left, B) reverse right, and C) reverse straight.

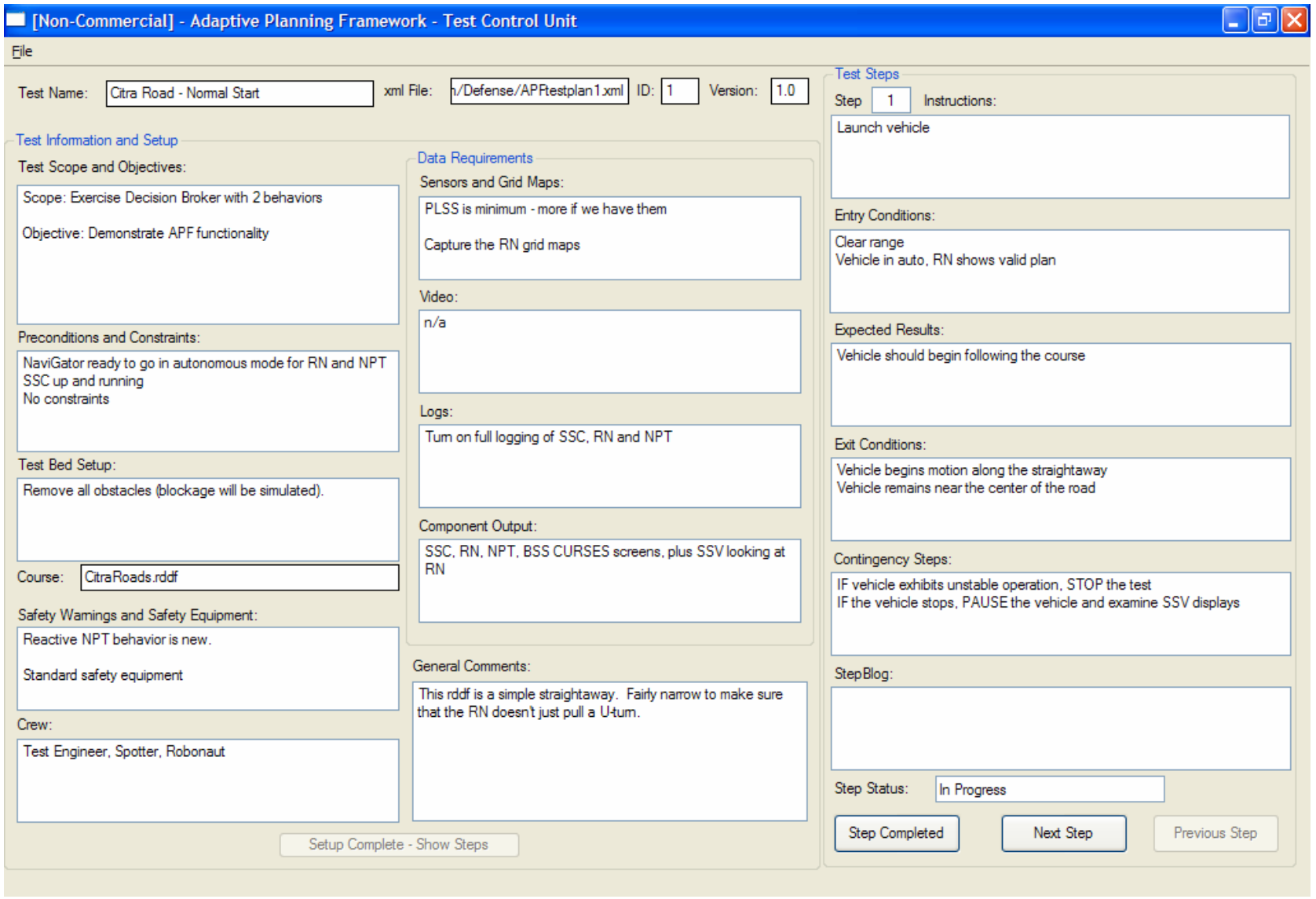


Figure 4-3. Screenshot of the Citra Test Run, including Setup and 1st Step.

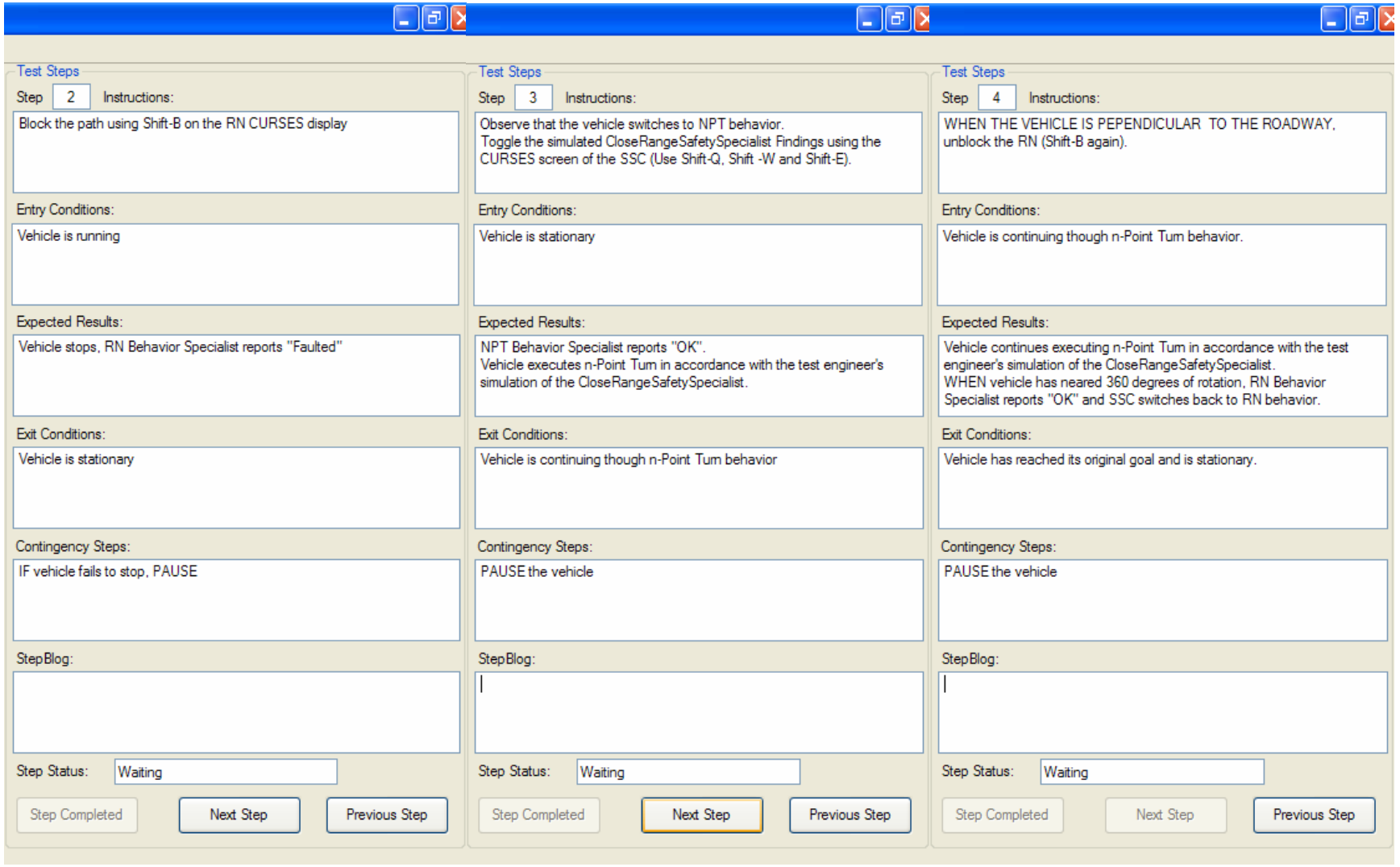


Figure 4-4. Screenshot Collage of Remaining Steps of the Citra Test Run.

CHAPTER 5
DISCUSSION AND FUTURE WORK

This chapter presents the author's assessment of the research conducted and outlines a series of additional research opportunities that would further enhance the Adaptive Planning Framework and its implementation on autonomous ground vehicles.

Assessment of the Adaptive Planning Framework

The Adaptive Planning Framework has been shown to be both a viable method for representing and managing complex, situation-dependent behavior on an Autonomous Ground Vehicle and a valuable contribution to researchers tasked with developing and fielding such a vehicle. The viability of the architecture and design was demonstrated by the Reference Implementation and the accompanying laboratory and field testing of it. The value of the Adaptive Planning Framework can be measured by the major role it is playing in the architecture and design of the AGV being fielded by Team Gator Nation for competing in the 2007 DARPA Urban Challenge.

To underscore this latter point, it is useful to mention the initial architecture that was presented to DARPA by Team Gator Nation. Figure 5-1 shows how the Adaptive Planning Framework was incorporated into the preliminary architecture of the 2007 DARPA Urban Challenge version of the NAVIGATOR. Note the growth of behaviors, and the Behavior Specialists needed to assess them, compared to the 2005 version (Figure 1-3). Likewise, there is an extensive proliferation of Situation Assessment Specialists needed to derive the many Findings needed to properly understand and respond to the situation at hand. This architecture (and its use of the Adaptive Planning Framework) continues to evolve as the team migrates towards a detailed design and as the team members become more directly involved in the details of how the framework operates and how it should be used.

Having a team of researchers dialoging about Meta Data, Findings, Specialists, and Decision Protocols further underscores the viability and usefulness of the framework. This emergent adoption of the ideas and innovations resulting from the subject research has already strengthened and improved the framework. For example, the notion of allowing the duties of the Decision Broker to be distributed into layers of abstraction (i.e., using Decision Protocols within a Behavior to select sub-behaviors) was a direct result of team discussions. In addition, many of the topics discussed in the Future Work section are either currently being addressed by members of the team or will be soon.

Future Work

Naturally, during the course of the current work there were a number of areas identified that present opportunities for further research. Some are general application of ideas and concepts discussed in Chapter 2 to the Adaptive Planning Framework. However, several that stand out as particularly important are summarized below, categorized by whether the opportunity relates more to the theoretical aspects of the Adaptive Planning Framework or its implementation.

Theoretical Opportunities

One ongoing research topic is how the framework will address conflict resolution, such as would be the case if two Specialists were arriving at opposing or incompatible conclusions. For conflicts that are foreseeable, this can be addressed by devising rules that explicitly resolve the conflict. This might be appropriate when two different styles of perception could reach conflicting Findings. The conflict resolution rule would need to take into account which sensor to trust under various (measurable) situations and then apply that knowledge at run time to select the appropriate one. Further research is needed for resolving conflicts that cannot be (or were not) foreseen (and therefore will not have any rules to divine them). Conflict resolution

strategies to be explored might include the use of general knowledge to break the conflict (e.g., if it is dark, trust LADARs more than cameras), and the use of probabilistic techniques and evidential reasoning (such that the degree of agreement or conflict can always be observed and used for discernment). For the (relatively) simple case of the Reference Implementation, the proper treatment of foreseeable conflicts was validated by review of the design and the absence of unforeseeable ones was confirmed by field testing.

Another research area is that of truth maintenance, which refers to the viability and "shelf life" of Findings and decisions over time. For Conditions, this potential problem is partially resolved by the requirement to re-prove their presence at every computational cycle of their hosting entity. All other Findings are stateful, which means that there must be conclusive evidence that a new state is preferable to the current state. What is not currently being addressed is the case where the current state is no longer the correct one (perhaps due to an undetected change in circumstances), but for some reason, the rules or algorithms for selecting the correct state do not succeed. Future researchers may want to explore the benefits of periodic confirmation of the current state and selecting a "safe" or "conservative" default state when no state can be definitively chosen. A related area that affects all Findings (Conditions included) is devising the proper response when input data needed by the rules or algorithms are not available. A condition being "Absent" because its presence cannot be proven could be due to a failure of one of the sensors that provides an input. In this case, the truth is that the Specialist does not know whether the Condition is present or absent. The current framework was extended to allow "unknown" as a legal value for a Condition (or any other Finding for that matter) in order to allow downstream users of that Finding to differentiate between a definitive result and an inability to reason. Use of this technique is not needed if it can be assured that the downstream

consumers will reach the same conclusion whether a Finding was reached explicitly or as a "default" due to missing input information.

Truth maintenance also relates to the Decision Broker and its Protocols. Once a new Protocol has been launched, but before it completes its duties and exits or transitions to another Protocol, circumstances could obviate its correctness. There is currently no general method available to externally abort a Protocol once it has begun execution. (Recall that the Contingency Steps can be used to exit the Protocol for situations that are planned for during the design of the Protocol.) This issue is somewhat mitigated by the use of high cycle rates as was the case for the Reference Implementation. If the Protocol can exit naturally in one or two tenths of a second, a mechanism for aborting a Protocol may be of little value. If this does surface as a problem, future researchers should devise a means to safely and stably recall a Protocol if the Decision Broker determines that another Protocol would be preferable.

Because of the heavy use of rules in design of both Findings and Protocols, it is possible and desirable to devise a robust explanation facility and accompanying man-machine interface. Future researchers should attempt to create tools that allow the system (and especially the Decision Broker) to use the inferencing chain to extemporaneously assemble an explanation of how a certain conclusion has been reached, perhaps augmented by a visualization of that chain. Such a capability would be of enormous benefit not only during testing and validation but also in helping operators understand the intent and reasoning of the system as it operates.

A final area of continuing research has to do with the assurance of continuity, stability, and safety during behavior transitions. The framework in its current state does not address how the transition from one behavior to another would affect the performance of the vehicle (or its individual components) during the transition. This issue did not surface as part of the Reference

Implementation because all transitions required the vehicle to be stationary. It was only when the team of Urban Challenge researchers began discussing how to use the Adaptive Planning Framework to manage the transition between behaviors that require the vehicle to be in motion that this issue became a dominant one. For now, the onus is on the various controllers and drivers to formulate proper commands to the vehicle actuators if their input commands would result in an unsafe or unstable condition. Future researchers should attempt to devise a mechanism that incorporates the resolution of discontinuities and instabilities into the Adaptive Planning Framework as they pursue the design of more complex behaviors and contemplate how one would transition among them.

Implementation Opportunities

The Reference Implementation led to the creation of a significant body of software in terms of stand-alone contributions to the CIMAR C libraries (libJausC), new software components, and additions to existing software components. There are several opportunities for future researchers at UF to build upon and improve this body.[1]

One area has to do with enhanced debugging and software validation. Since so many decisions are either time (or timing) dependent or only persist for a brief period of time, this software would benefit from some form of temporal "instrumentation" scheme. Future researchers are encouraged to pursue a standardized way of incorporating clocks and timers into the software entities and an automated technique for managing their results. The availability of such a utility would be quite useful to assist in understanding timing issues and the sequencing of events. It would be especially useful if incorporated along with a system wide, centralized clock

[1] This limitation is mainly due to the massive learning curve required for the CIMAR implementation of JAUS as well as the lack of general availability of its extended features. Since the standard portions of the CIMAR implementation have been placed in an Open Source repository (www.openjaus.com), this restriction to UF researchers could be relaxed for a highly motivated outside researcher.

so that all hardware nodes would be reporting using the same time frame. Similarly, automating the logging of intermediate conclusions and Findings (perhaps intrinsic to the Meta Data software utilities) would be of great value during testing and validation.

This leads to a second area that would benefit from further refinement of the software developed for the Reference Implementation. The incorporation of the Adaptive Planning Framework and Meta Data structures and function calls into an existing software component is tedious, vulnerable to typing mistakes, and subject to multiple (and perhaps incompatible) interpretation and extension by multiple developers. There is an opportunity for future UF researchers to develop a centralized Meta Data Manager as an integral part of the CIMAR JAUS implementation. There is a precedent for this in the handling of JAUS Service Connections and JAUS Services. Such a tool would simplify the creation and use of Specialists and their Findings (and any other Meta Data) while providing higher quality software and more productive developers. If a Meta Data Manager were available, much of the code currently seen in the Reference Implementation components could be reduced through function calls out to the Meta Data Manager. This same toolset could also house improvements and extensions to the Meta Data utilities already in place.

While the Meta Data Manager focuses on improving the use of the Adaptive Planning Framework at run-time, this final opportunity area for future research deals with improving it at design-time. Because the interactions among Findings and the Specialists that produce them can be quite intricate and because of the need to manage and standardize nomenclature in the domain namespace, there is a strong need for development of an Adaptive Planning Framework visualization and validation toolkit. It would be highly beneficial if system designers could visualize the connections between the publishers and subscribers of Findings, and how their

various possible values were used in rules, algorithms, and Protocols. Similarly, an underlying data base containing the current dictionary of Specialists, Findings, and their enumerated possible values would help designers conform to the growing body of definitions while avoiding namespace collisions. In addition, automation of the Knowledge Representation tools and templates would improve designer productivity and perhaps even lead into automatic generation of compliant source code and documentation. Future UF researchers are encouraged to pursue one or more of these Adaptive Planning Framework designer's workbench areas.

Conclusion

The Adaptive Planning Framework makes a significant contribution to advancing the state of the practice of intelligent systems in general and AGVs in particular. Its adoption by Team Gator Nation means that it will be improved and extended by future researchers. If that occurs, this work will have been the genesis of contributions in the future that are even more significant, and the catalyst of a new way of achieving more intelligent and more autonomous ground vehicles.

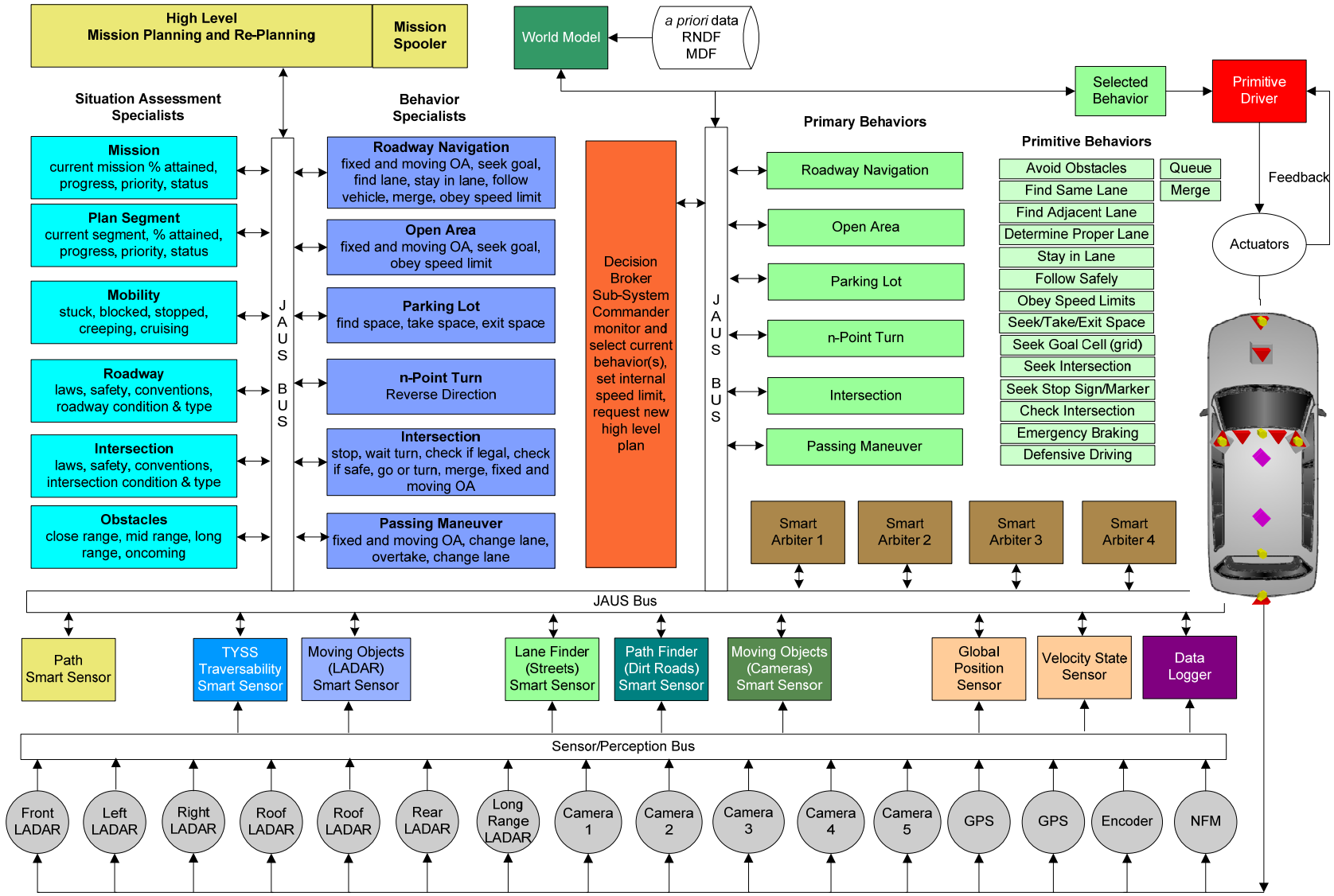


Figure 5-1. A Preliminary Version of the Component Block Diagram proposed by Team Gator Nation for DARPA Urban Challenge 2007.

APPENDIX A
PROOF OF CONCEPT PROTOTYPE

A Proof of Concept prototype of the Adaptive Planning Framework was developed early on to help clarify and validate the idea. Although quite limited in scope, it served its purpose well and became the springboard for the subsequent work.

Scope of Prototype

A simple version of a LISP-based Intelligent Situation Assessment System (ISAS) was built to support a simulated autonomous ground vehicle. This initial attempt was merely a prototype of the envisioned system and, as such, operated "on the bench," using manually entered input data that crudely simulated the operation/behavior of sensors on an autonomous ground vehicle.

Since the emphasis for this prototype was to establish a first cut at Situation Assessment, only the following Conditions, States, and Events were included in the scope:

- Conditions:
  - Rugged Terrain
  - Close-Range-Obstacle
  - Long-Range-Obstacle

- States:
  - Mission-Mode is {Ahead-of-Schedule | Nominal | Behind-Schedule}
  - Mission-Goal is {Optimize-Speed | Optimize-Risk}
  - Operating-Mode is {Low-Speed | High-Speed}
  - Sensor-Mode is {Low-Res | High-Res}
  - Sensor-Confidence is {Low | High}

- Events:
  - Sensor Object-Detection is {True | False}

Inputs to the ISAS prototype included the following list:

- Derived non-visual sensor readings (e.g., rate-of-change of heading, roll, pitch)
- Sensor metadata (e.g., whiteout/blackout, closest object detected)
- Planning and control elements (e.g., mission goal completion rate)
- Previous findings of ISAS

The design used a blackboard as the source and sink of all information used by/produced by the ISAS prototype. Simulating the operation of the vehicle simply required new information to be externally placed onto the blackboard by the user. The effort also required creation of a data structure for blackboard elements and creation of a rule structure to allow the ISAS prototype to reason upon the facts/findings placed on the blackboard.

Approach

In order to provide the forward-chaining inference engine needed by ISAS, a basic implementation was adopted from (Winston and Horn 1989). This was followed by implementing the extensions and modifications needed to achieve the specific requirements of ISAS. There were several main areas that had to be addressed beyond the functionality provided in the text:

- Control flow and User Interface for running in a 'continuous mode' and for adding new facts
- Truth Maintenance (for the retraction of previous findings in light of new findings)
- Inclusion of a very simple device for resetting Conditions to their default value before each inferencing iteration
- Inclusion of Predicate Tests in rule antecedents (instead of only patterns)

Concept of Operations

The ISAS prototype employs a forward-chaining inference engine that systematically attempts to match the antecedents of rules stored in the Rule Base with facts stored on the Blackboard. Each time a match is found, it treats that antecedent as satisfied. When it has found all of the antecedents of a rule to be satisfied, it adds the consequent of the rule to the Blackboard. In order to maximize the usability of the base code supplied by Winston & Horn, their data structures for representing rules and facts were adopted. Thus, the ISAS prototype

Blackboard is actually the system variable *assertions* which is used extensively (and explicitly) throughout their code.

Winston & Horn also included use of LISP Streams to hold the rules and facts used by the inference engine, so the ISAS prototype does too. Likewise, Winston & Horn introduced a pattern-matching concept that allows for variables in the matches. This proved to be very useful and was embraced by the ISAS prototype (and was extended, as discussed under Predicate Testing). The form of a Rule in the ISAS prototype is as follows:

```
(<Rule Name/ID>
                    (<fact 1>)
                    (<fact 2>)
                         :
                         :
                    (<fact n>)
                    (<consequent>))
```

Thus, each rule may have one or more antecedents (connected by an implied AND) but only one consequent.

The form of the Blackboard, which is embodied as the *assertions* stream, is as follows:

```
((<fact 1>
 ((<fact 2>)
  ((<fact …>)
   ((<fact n>)))))
```

A rule that contains variables uses the special notation of (? variable-name) to identify the existence and placement of the variable. By having named variables, the inference engine can support multiple variables within a given rule. The first time such a variable is encountered, the engine attempts to associate the variable with a matching pattern in the blackboard. If successful, that binding is applied to any future encounters with that variable. This enables creation of very powerful rules that can be applied to multiple circumstances. Take for example the following rule from the ISAS prototype:

```
("Sensor 3"
                  ((? sensor) white-out is true)
                  ((? sensor) confidence is low))
```

The ISAS prototype will match this rule with the fact (radar-sensor white-out is true) and subsequently add (radar-sensor confidence is low) to the Blackboard. Thus, pattern matching allows this rule to set the confidence level to "low" of *any* sensor that reports a "white-out" condition. In English, this rule of thumb might be stated as, "If the White-out of some Sensor is True, then the Confidence of that Sensor is Low." As long as such a rule can be safely applied to all situations, the author of the rule base need not be concerned with exactly which sensors have been installed on the vehicle or what their Ids are.

Issues Identified

Conflict Resolution was *not* addressed (i.e., if Rule 1 proves that an object's state is "A" and Rule 2 proves that that same object's state is "B", whichever rule fires last, wins). Naturally, this must ultimately follow a less random process for final resolution.

An Object Oriented approach was not used. The best way to represent the various physical and conceptual entities used by ISAS is as objects. For example, a State Object could encapsulate its allowed values as well as its default value. A Sensor Object might contain not only its current value, but might locally calculate its own rate of change, as well as its current confidence level, units, and so on.

To be truly useful, such a system as this would need a robust user interface for adding/maintaining Rules and Objects and for visualizing the current findings, with traces of rationale for a finding of interest to the user.

Algorithm and Program Logic Flow

Inferencing Control Strategy and User Interface

ISAS operates in a loop that executes the following steps until the user enters (QUIT!):

- Prompt User
- Read Input
- Reset Conditions to their default value
- Add User Input to Blackboard
- Run the Inference Engine (which prints each new finding and which rule found it)
- Print Blackboard content

Before starting the main loop, it loads in the knowledge base ("forward-chain.dta"), runs the Inference Engine, and displays the initial Blackboard.

Truth Maintenance

The approach presented in Winston & Horn did not provide any mechanism for destructive operations on the Blackboard. In other words, once a fact was proven by a rule, that fact would remain permanently on the Blackboard. For classifying mammals (or really, in any static situation), that might work just fine. However, for the ISAS prototype, the situation is in a constant state of flux and, therefore, must have the ability to retract an earlier finding as it becomes obviated by a new finding (or by a new user entry). To accomplish this enhancement within the scope of a prototype, two constraints on the format of rule consequents were introduced that make it easier to introduce a degree of Truth Maintenance to the Winston & Horn inference engine:

- The “value” of a finding must appear in the last position of the list that comprises the rule consequent
- Findings must be single-valued (cardinality of 1)

The first constraint allows the ISAS prototype to match new findings about to be added to the Blackboard with existing findings that have a different value (i.e., everything matches but the

value). The second constraint means that if a new value of a finding is discovered, it is always correct to delete its previous value from the Blackboard.

The algorithm for accomplishing this is to interject a new function between the discovery of a new fact and its entry onto the Blackboard. This function, named RETRACT-OLD-ASSERTION, is called with a generalized search pattern `(append (butlast assertion) '((?x)) )` which, when applied against the Blackboard stream (*assertions*) will match that assertion even if its previous value was different than the newly found one. The function uses iteration to search the Blackboard stream until it either finds a match or the stream becomes exhausted. If a match is found, the function deletes the matching assertion from the Blackboard, using another new function named DELETE-ASSERTION, and sets a success flag to end the search loop. To create the assertion to be deleted, RETRACT-OLD-ASSERTION appends the stub of the pattern `(butlast assertion-pattern)` to the matched variable that was found `(last (first result))`.

The algorithm for DELETE-ASSERTION is to recursively test and divide the stream such that it builds a new stream with the to-be-deleted assertion absent.

Retraction of Conditions

As mentioned earlier, Conditions enjoy the benefit of needing only those rules which prove their presence, putting the onus on the inference engine to retract each condition prior to executing the forward-chaining operation, i.e., the off-normal value of each Condition must be re-proven each time the inference engine is run. This means that the ISAS prototype must have access to which potential findings on the Blackboard are Conditions.

For the prototype, a file named “conditions.dta” was created which contains a series of ADD-ASSERTION statements that asserts the default value of each condition known to ISAS.

Before each new execution of the inference engine, this file is applied to the Blackboard with the effect of retracting any Condition that was present (i.e., set its value to "Absent").

Predicate Testing in Rule Antecedents

Beyond the exact and variable pattern-matching capability provided by Winston & Horn, the ISAS prototype needed to support predicate tests within the antecedent portion of a rule. A rule that contains a predicate test in an antecedent uses the special notation of (! test) to identify the existence of a predicate test, followed by the predicate test itself. The general form of a predicate testing antecedent is `((! Test) (<predicate test>))`, where `<predicate test>` must include the predicate and the correct number and type of arguments. Consider the following example of a rule that includes predicate testing:

```
("Sensor 1"
            ((? sensor) object-detection is true)
            ((? sensor) object-distance is (? distance))
            ((! test) (> (? distance) 15))
            ((? sensor) confidence is high)
            (long-range-obstacle is present)))
```

In this rule from the ISAS prototype, the first antecedent will bind to any sensor that has detected an object. The second antecedent will extend the set of bindings to include the distance of the nearest object that was detected. With success of those two antecedents being matched and their variables bound, the predicate test can be performed 'locally' (i.e., without consultation with the Blackboard). If the distance found is indeed greater than 15, the engine will try to match the fourth antecedent and, if found to be true, will assert the consequent. If the predicate test fails, the rule will be abandoned.

The following constraints were placed on a predicate testing antecedent:

- All bindings must have been made in prior antecedents (since predicate testing antecedents will never match anything on the Blackboard, they will never create a new binding of a variable)

- Multiple predicate tests for a given rule must be bundled into a single (compound) predicate test
- The LISP validity of the test rests on the shoulders of the rule designer as no error checking/validation is performed

Incorporating predicate testing significantly extends the power of the ISAS prototype and the rules that can be crafted.

Testing Results

Test Case Scenario Set-up

To support testing of ISAS, one has to first establish some notion of the autonomous vehicle to be simulated, especially the sensors and other information that would be available to provide input to ISAS (see Tables A-1, A-2 and A-3). The Specialists that embody ISAS must also be established along with the Conditions, States, and Events about which the Specialists will be asked to render their findings, as enumerated in Tables A-4 through A-7.

Finally, the Rules that embody each Specialist must be defined. To round out the test case scenario set-up, the Blackboard initialization facts and the Condition Defaults are reproduced.

Rule Base

```
;Sensor Specialist's Rules:

("Sensor 1"
            ((? sensor) object-detection is true)
            ((? sensor) object-distance is (? distance))
            ((! test) (> (? distance) 15))
            ((? sensor) confidence is high)
            (long-range-obstacle is present))

("Sensor 2"
            ((? sensor) object-detection is true)
            ((? sensor) object-distance is (? distance))
            ((! test) (<= (? distance) 15))
            ((? sensor) confidence is high)
            (close-range-obstacle is present))

("Sensor 3"
            ((? sensor) white-out is true)
            ((? sensor) confidence is low))
```

```
("Sensor 4"
            ((? sensor) black-out is true)
            ((? sensor) confidence is low))

("Sensor 5"
            ((? sensor) white-out is false)
            ((? sensor) black-out is false)
            ((? sensor) confidence is high))

;Vehicle Specialist's Rules:

("Vehicle 1"
            (roll-rate is high)
            (pitch-rate is high)
            (rugged-terrain is present))

("Vehicle 2"
            (roll-rate is high)
            (heading-rate is high)
            (rugged-terrain is present))

("Vehicle 3"
            (heading-rate is high)
            (pitch-rate is high)
            (rugged-terrain is present))

;Mission Specialist's Rules:

("Mission 1"
            (rugged-terrain is present)
            (operating-mode is low-speed))

("Mission 2"
            (close-range-obstacle is present)
            (operating-mode is low-speed))

("Mission 3"
            (goal-completion-rate is (? percent))
            ((! test) (< (? percent) 90))
            (mission-mode is behind-schedule))

("Mission 4"
            (goal-completion-rate is (? percent))
            ((! test) (> (? percent) 110))
            (mission-mode is ahead-of-schedule))

("Mission 5"
            (goal-completion-rate is (? percent))
            ((! test) (and (>= (? percent) 90) (<= (? percent) 110)))
            (mission-mode is nominal))

("Mission 6"
            (mission-mode is behind-schedule)
            (mission-goal is optimize-speed))
```

```
("Mission 7"
            (mission-mode is nominal)
            (mission-goal is optimize-speed))

("Mission 8"
            (mission-mode is ahead-of-schedule)
            (mission-goal is optimize-risk))

("Mission 9"
            (mission-goal is optimize-risk)
            (operating-mode is low-speed))
("Mission 10"
            (close-range-obstacle is absent)
            (rugged-terrain is absent)
            (mission-goal is optimize-speed)
            (operating-mode is high-speed))

("Mission 11"
            (operating-mode is low-speed)
            (sensor-mode is high-res))

("Mission 12"
            (operating-mode is high-speed)
            (sensor-mode is low-res))
```

Blackboard initialization

```
(radar-sensor object-detection is false)
(ladar-sensor object-detection is false)
(long-range-obstacle is absent)
(close-range-obstacle is absent)
(roll-rate is low)
(pitch-rate is low)
(heading-rate is low)
(rugged-terrain is absent)
(radar-sensor white-out is false)
(ladar-sensor white-out is false)
(radar-sensor black-out is false)
(ladar-sensor black-out is false)
(goal-completion-rate is 100)
```

Condition defaults

```
(rugged-terrain is absent)
(long-range-obstacle is absent)
(close-range-obstacle is absent)
```

Test Cases and Results

Test Case 1 – initial resolution of Blackboard at time = 0

Merely running ISAS exercises several key areas of the system, including the basic forward-chaining and pattern-matching functions, as well as predicate testing.

```
Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
Current BLACKBOARD:
((RADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
  ((LONG-RANGE-OBSTACLE IS ABSENT)
   ((CLOSE-RANGE-OBSTACLE IS ABSENT)
    ((ROLL-RATE IS LOW)
     ((PITCH-RATE IS LOW)
      ((HEADING-RATE IS LOW)
       ((RUGGED-TERRAIN IS ABSENT)
        ((RADAR-SENSOR WHITE-OUT IS FALSE)
         ((LADAR-SENSOR WHITE-OUT IS FALSE)
          ((RADAR-SENSOR BLACK-OUT IS FALSE)
           ((LADAR-SENSOR BLACK-OUT IS FALSE)
            ((GOAL-COMPLETION-RATE IS 100)
             ((RADAR-SENSOR CONFIDENCE IS HIGH)
              ((LADAR-SENSOR CONFIDENCE IS HIGH)
               ((MISSION-MODE IS NOMINAL)
                ((MISSION-GOAL IS OPTIMIZE-SPEED)
                 ((OPERATING-MODE IS HIGH-SPEED)
                  ((SENSOR-MODE IS LOW-RES)
                   EMPTY-STREAM)))))))))))))))))))))

Enter a new fact, or quit! to end the session (must be in a simple list):
```

Note that several new findings are discovered on this initial run, mainly that both the Radar and Ladar sensors have high Confidence levels, the Mission-Mode is Nominal, the Mission-Goal is Optimize-Speed, the Operating-Mode is High-Speed, and the Sensor Mode is Low-Res.

Test Case 2a – Rugged Terrain is encountered

The test case requires the user to enter two facts, as if they had come from vehicle sensors: that the Roll-Rate is High and the Pitch-Rate is High. This test case adds exercising retraction of obviated facts.

```
Enter a new fact, or quit! to end the session (must be in a simple
list):(roll-rate is high)
; loading conditions.dta

Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
```

```
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
Current BLACKBOARD:
((RADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
  ((PITCH-RATE IS LOW)
   ((HEADING-RATE IS LOW)
    ((RADAR-SENSOR WHITE-OUT IS FALSE)
     ((LADAR-SENSOR WHITE-OUT IS FALSE)
      ((RADAR-SENSOR BLACK-OUT IS FALSE)
       ((LADAR-SENSOR BLACK-OUT IS FALSE)
        ((GOAL-COMPLETION-RATE IS 100)
         ((RUGGED-TERRAIN IS ABSENT)
          ((LONG-RANGE-OBSTACLE IS ABSENT)
           ((CLOSE-RANGE-OBSTACLE IS ABSENT)
            ((ROLL-RATE IS HIGH)
             ((RADAR-SENSOR CONFIDENCE IS HIGH)
              ((LADAR-SENSOR CONFIDENCE IS HIGH)
               ((MISSION-MODE IS NOMINAL)
                ((MISSION-GOAL IS OPTIMIZE-SPEED)
                 ((OPERATING-MODE IS HIGH-SPEED)
                  ((SENSOR-MODE IS LOW-RES)
                   EMPTY-STREAM)))))))))))))))))

Enter a new fact, or quit! to end the session (must be in a simple
list):(pitch-rate is high)
; loading conditions.dta

Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Vehicle 1 indicates (RUGGED-TERRAIN IS PRESENT).
Rule Mission 1 indicates (OPERATING-MODE IS LOW-SPEED).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 11 indicates (SENSOR-MODE IS HIGH-RES).
Nothing new noted.
Current BLACKBOARD:
((RADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
  ((HEADING-RATE IS LOW)
   ((RADAR-SENSOR WHITE-OUT IS FALSE)
    ((LADAR-SENSOR WHITE-OUT IS FALSE)
     ((RADAR-SENSOR BLACK-OUT IS FALSE)
      ((LADAR-SENSOR BLACK-OUT IS FALSE)
       ((GOAL-COMPLETION-RATE IS 100)
        ((ROLL-RATE IS HIGH)
         ((LONG-RANGE-OBSTACLE IS ABSENT)
          ((CLOSE-RANGE-OBSTACLE IS ABSENT)
           ((PITCH-RATE IS HIGH)
            ((RADAR-SENSOR CONFIDENCE IS HIGH)
             ((LADAR-SENSOR CONFIDENCE IS HIGH)
              ((RUGGED-TERRAIN IS PRESENT)
               ((OPERATING-MODE IS LOW-SPEED)
                ((MISSION-MODE IS NOMINAL)
                 ((MISSION-GOAL IS OPTIMIZE-SPEED)
                  ((SENSOR-MODE IS HIGH-RES)
                   EMPTY-STREAM)))))))))))))))))
```

Note that after the Roll-Rate was changed, nothing new was discovered. This is because the Vehicle Specialist rules require a high rate of change on 2 of the 3 position sensors. Once the Pitch-Rate is also changed, Rugged-Terrain is deemed Present, the Operating-Mode is set to Low-Speed, and the Sensor-Mode is set to High-Res. Even though the Mission-Goal remains Optimize-Speed, the Mission Specialist sets the system into Low-Speed mode.

Test Case 2b – Rugged Terrain no longer present

The test case requires the user to enter just one fact, as if it had come from vehicle sensors: that the Roll-Rate is Low. This test case adds resetting of Conditions to the mix.

```
Enter a new fact, or quit! to end the session (must be in a simple
list):(roll-rate is low)
; loading conditions.dta

Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
Current BLACKBOARD:
((RADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
  ((HEADING-RATE IS LOW)
   ((RADAR-SENSOR WHITE-OUT IS FALSE)
    ((LADAR-SENSOR WHITE-OUT IS FALSE)
     ((RADAR-SENSOR BLACK-OUT IS FALSE)
      ((LADAR-SENSOR BLACK-OUT IS FALSE)
       ((GOAL-COMPLETION-RATE IS 100)
        ((PITCH-RATE IS HIGH)
         ((RUGGED-TERRAIN IS ABSENT)
          ((LONG-RANGE-OBSTACLE IS ABSENT)
           ((CLOSE-RANGE-OBSTACLE IS ABSENT)
            ((ROLL-RATE IS LOW)
             ((RADAR-SENSOR CONFIDENCE IS HIGH)
              ((LADAR-SENSOR CONFIDENCE IS HIGH)
               ((MISSION-MODE IS NOMINAL)
                ((MISSION-GOAL IS OPTIMIZE-SPEED)
                 ((OPERATING-MODE IS HIGH-SPEED)
                  ((SENSOR-MODE IS LOW-RES)
                   EMPTY-STREAM))))))))))))))))))))
```

Note that after the Roll-Rate was changed back to Low, Rugged-Terrain was allowed to revert to Absent, since there was now insufficient sensor data to prove that Rugged-Terrain was

Present. Accordingly, the Operating-Mode is set back to High-Speed. This happened because the Mission-Goal remained to be Optimize-Speed.

Test Case 3a – long range obstacle detection

After reinitializing ISAS, this test case requires the user to add two facts: that an obstacle has been detected and that the distance to the object is 20 meters.

```
Enter a new fact, or quit! to end the session (must be in a simple
list):(radar-sensor object-detection is true)
; loading conditions.dta

Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
Current BLACKBOARD:
((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((ROLL-RATE IS LOW)
  ((PITCH-RATE IS LOW)
   ((HEADING-RATE IS LOW)
    ((RADAR-SENSOR WHITE-OUT IS FALSE)
     ((LADAR-SENSOR WHITE-OUT IS FALSE)
      ((RADAR-SENSOR BLACK-OUT IS FALSE)
       ((LADAR-SENSOR BLACK-OUT IS FALSE)
        ((GOAL-COMPLETION-RATE IS 100)
         ((RUGGED-TERRAIN IS ABSENT)
          ((LONG-RANGE-OBSTACLE IS ABSENT)
           ((CLOSE-RANGE-OBSTACLE IS ABSENT)
            ((RADAR-SENSOR OBJECT-DETECTION IS TRUE)
             ((RADAR-SENSOR CONFIDENCE IS HIGH)
              ((LADAR-SENSOR CONFIDENCE IS HIGH)
               ((MISSION-MODE IS NOMINAL)
                ((MISSION-GOAL IS OPTIMIZE-SPEED)
                 ((OPERATING-MODE IS HIGH-SPEED)
                  ((SENSOR-MODE IS LOW-RES)
                   EMPTY-STREAM)))))))))))))))))))

Enter a new fact, or quit! to end the session (must be in a simple
list):(radar-sensor object-distance is 20)
; loading conditions.dta

Rule Sensor 1 indicates (LONG-RANGE-OBSTACLE IS PRESENT).
Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
```

```
Current BLACKBOARD:
((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((ROLL-RATE IS LOW)
  ((PITCH-RATE IS LOW)
   ((HEADING-RATE IS LOW)
    ((RADAR-SENSOR WHITE-OUT IS FALSE)
     ((LADAR-SENSOR WHITE-OUT IS FALSE)
      ((RADAR-SENSOR BLACK-OUT IS FALSE)
       ((LADAR-SENSOR BLACK-OUT IS FALSE)
        ((GOAL-COMPLETION-RATE IS 100)
         ((RADAR-SENSOR OBJECT-DETECTION IS TRUE)
          ((RUGGED-TERRAIN IS ABSENT)
           ((CLOSE-RANGE-OBSTACLE IS ABSENT)
            ((RADAR-SENSOR OBJECT-DISTANCE IS 20)
             ((LONG-RANGE-OBSTACLE IS PRESENT)
              ((RADAR-SENSOR CONFIDENCE IS HIGH)
               ((LADAR-SENSOR CONFIDENCE IS HIGH)
                ((MISSION-MODE IS NOMINAL)
                 ((MISSION-GOAL IS OPTIMIZE-SPEED)
                  ((OPERATING-MODE IS HIGH-SPEED)
                   ((SENSOR-MODE IS LOW-RES)
                   EMPTY-STREAM))))))))))))))))))))
```

Note that the Close-Range-Obstacle has been detected, but that alone does not change the behavior of the vehicle because it is still too far away.

Test Case 3b – close range obstacle detection

This test case requires the user to add one fact: that the distance to the object is now 10 meters.

```
Enter a new fact, or quit! to end the session (must be in a simple
list):(radar-sensor object-distance is 10)
; loading conditions.dta

Rule Sensor 2 indicates (CLOSE-RANGE-OBSTACLE IS PRESENT).
Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 2 indicates (OPERATING-MODE IS LOW-SPEED).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 11 indicates (SENSOR-MODE IS HIGH-RES).
Nothing new noted.
Current BLACKBOARD:
((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((ROLL-RATE IS LOW)
  ((PITCH-RATE IS LOW)
   ((HEADING-RATE IS LOW)
    ((RADAR-SENSOR WHITE-OUT IS FALSE)
     ((LADAR-SENSOR WHITE-OUT IS FALSE)
      ((RADAR-SENSOR BLACK-OUT IS FALSE)
       ((LADAR-SENSOR BLACK-OUT IS FALSE)
```

```
((GOAL-COMPLETION-RATE IS 100)
 ((RADAR-SENSOR OBJECT-DETECTION IS TRUE)
  ((RUGGED-TERRAIN IS ABSENT)
   ((LONG-RANGE-OBSTACLE IS ABSENT)
    ((RADAR-SENSOR OBJECT-DISTANCE IS 10)
     ((CLOSE-RANGE-OBSTACLE IS PRESENT)
      ((RADAR-SENSOR CONFIDENCE IS HIGH)
       ((LADAR-SENSOR CONFIDENCE IS HIGH)
        ((OPERATING-MODE IS LOW-SPEED)
         ((MISSION-MODE IS NOMINAL)
          ((MISSION-GOAL IS OPTIMIZE-SPEED)
           ((SENSOR-MODE IS HIGH-RES)
            EMPTY-STREAM)))))))))))))))))
```

Note that now, the Close-Range-Obstacle has been detected, which, in turn, causes the vehicle to change into Low-Speed operation with High-Res sensors.

Test Case 3c – obstacle avoided

This test case requires the user to add one fact: that an obstacle is no longer detected. Once the obstacle has been avoided, the detector will have a false reading.

```
Enter a new fact, or quit! to end the session (must be in a simple
list):(radar-sensor object-detection is false)
; loading conditions.dta

Rule Sensor 5 indicates (RADAR-SENSOR CONFIDENCE IS HIGH).
Rule Sensor 5 indicates (LADAR-SENSOR CONFIDENCE IS HIGH).
Rule Mission 5 indicates (MISSION-MODE IS NOMINAL).
Rule Mission 7 indicates (MISSION-GOAL IS OPTIMIZE-SPEED).
Rule Mission 10 indicates (OPERATING-MODE IS HIGH-SPEED).
Rule Mission 12 indicates (SENSOR-MODE IS LOW-RES).
Nothing new noted.
Current BLACKBOARD:
((LADAR-SENSOR OBJECT-DETECTION IS FALSE)
 ((ROLL-RATE IS LOW)
  ((PITCH-RATE IS LOW)
   ((HEADING-RATE IS LOW)
    ((RADAR-SENSOR WHITE-OUT IS FALSE)
     ((LADAR-SENSOR WHITE-OUT IS FALSE)
      ((RADAR-SENSOR BLACK-OUT IS FALSE)
       ((LADAR-SENSOR BLACK-OUT IS FALSE)
        ((GOAL-COMPLETION-RATE IS 100)
         ((RADAR-SENSOR OBJECT-DISTANCE IS 10)
          ((RUGGED-TERRAIN IS ABSENT)
           ((LONG-RANGE-OBSTACLE IS ABSENT)
            ((CLOSE-RANGE-OBSTACLE IS ABSENT)
             ((RADAR-SENSOR OBJECT-DETECTION
                            IS
                            FALSE)
              ((RADAR-SENSOR CONFIDENCE IS HIGH)
               ((LADAR-SENSOR CONFIDENCE IS HIGH)
```

```
((MISSION-MODE IS NOMINAL)
 ((MISSION-GOAL IS OPTIMIZE-SPEED)
  ((OPERATING-MODE IS HIGH-SPEED)
   ((SENSOR-MODE IS LOW-RES)
   EMPTY-STREAM))))))))))))))))))
```

Note that the Long- and Close-Range Obstacles have been retracted and the operating modes of the vehicle and sensors have been returned to their normal states.

Table A-1. Environmental Sensors.

Sensor Name	Possible Inputs to ISAS	Legal Values of Input
Radar-Sensor	Object-Detection	True | False
	Object-Distance	Number in meters
	White-out	True | False
	Black-out	True | False
Ladar-Sensor	Object-Detection	True | False
	Object-Distance	Number in meters
	White-out	True | False
	Black-out	True | False

Table A-2. Vehicle Sensors.

Sensor Name	Possible Inputs to ISAS	Legal Values of Input
Roll-Rate	Roll-Rate	Low | High
Pitch-Rate	Pitch-Rate	True | False
Heading-Rate	Heading-Rate	True | False

Table A-3. Mission Information.

Sensor Name	Possible Inputs to ISAS	Legal Values of Input
Mission-Goal	Mission-Goal	Optimize Speed | Optimize Risk

Table A-4. Vehicle Specialist.

Condition Name	Condition Attribute	Legal Values of Condition
Rugged-Terrain	Rugged-Terrain	Absent | Present

Table A-5. Sensor Specialist (States).

State Name	State Attribute	Legal Values of State
Radar-Sensor	Confidence	Low | High
Ladar-Sensor	Confidence	Low | High

Table A-6. Sensor Specialist (Conditions).

Condition Name	Condition Attribute	Legal Values of Condition
Radar-Sensor	Long-Range-Obstacle	Absent | Present
	Short-Range-Obstacle	Absent | Present
Ladar-Sensor	Long-Range-Obstacle	Absent | Present
	Short-Range-Obstacle	Absent | Present

Table A-7. Mission Specialist.

State Name	State Attribute	Legal Values of State
Sensor-Mode	Sensor-Mode	Low-Res | High-Res
Operating-Mode	Operating-Mode	Low-Speed | High-Speed

APPENDIX B
EARLY IMPLEMENTATION AND DESERT TESTING ON THE 2005 DARPA GRAND CHALLENGE NAVIGATOR

The NAVIGATOR vehicle built for DARPA Grand Challenge 2005 carried on it a very simple implementation of the Adaptive Planning Framework. The primary duty of the Situation Assessment (SA) component was to provide supplemental speed control to the Reactive Driver (RD) component. The SA component consisted of an Obstacle Specialist and a Terrain Ruggedness Specialist. The Obstacle Specialist used a data feed from the Planar Ladar Smart Sensor (PLSS) to determine the presence of two Conditions related to whether the space directly in front of the vehicle was free of obstacles beyond the 30-meter planning horizon (i.e., 30m out to the 80m range-limit of the Ladar device). The Terrain Ruggedness Specialist used the instantaneous pitch rate and roll rate of the vehicle (provided by the Velocity State Sensor (VSS) component) to classify the current state of the terrain as "Smooth," "Rugged," or "Very Rugged." Based on the Obstacle Conditions and Terrain Ruggedness State, with appropriate hysteresis control and dampening, the permitted speed of the vehicle was selected and sent to the RD. For example, if the terrain were Smooth and no Long Range Obstacle or Short Range Obstacle were present, then the RD would be permitted to drive the vehicle up to its highest allowable speed and, thus, faster than an empirically derived Obstacle Avoidance speed of 7.2 mps (16 mph).

The following Conditions, States, and Events were included in the DGC2005 NAVIGATOR:

- Conditions:
  - Short-Range-Obstacle
  - Long-Range-Obstacle
- States:
  - Terrain is {Smooth | Rugged | Very Rugged}

- Events:
    - none

Inputs to the SA component included the following (updated ~20Hz):

- PLSS raw image array (range data at 180 degree sweep, 1 degree resolution)
- PLSS self-assessment of array health (sensor meta-data)
- Vehicle roll rate
- Vehicle pitch rate

The design placed the SA component "inside" the PLSS component, thus eliminating the need for data marshalling between the sensor component and the SA component (the Ladar array pointer and health data were passed as arguments to an SA function call). The SA component used a standard JAUS message to obtain the vehicle roll rate and pitch rate from the VSS component. The SA output was a standard JAUS Set Speed message sent to the RD. The reasoning demands for this initial implementation were simple enough to use clusters of If/Then/Else statements in C, rather than a formal inference engine, to provide the necessary logic processing to assess the conditions and states and make the output decision. The concept of operations for this implementation was, for each iteration of the PLSS, to invoke a call to the SA function that included a pointer to the latest Ladar data array and its self-assessed (Boolean) health status. The VSS input was set up via a standard JAUS service connection that stimulates an update at 20 Hz.

Test Case Scenario Set-up

To support testing of the SA component on the NAVIGATOR, one has to first establish the parameters that will be available to the component (see Tables B-1 and B-2). The Specialists that embody the SA component must also be established along with the Conditions, States, and Events about which the Specialists will be asked to render their findings, as enumerated in

Tables B-3 and B-4. The Decision Broker then has the task of assigning the appropriate speed limit, with the choices and decision logic enumerated in Table B-5.

Desert Testing and Results

Although the testing and tuning of the SA component began in the CIMAR lab and Citra test range, most of it was performed after Team CIMAR arrived at the Mohave Desert. The component was designed with tuning in mind by establishing every tunable parameter as a member of a configuration file. This allowed the tester to rapidly and easily experiment with varying parametric values. The first task was to tune the parameters for the Terrain-State. The initial settings for the Rugged and Very-Rugged thresholds were based on analysis of roll-rate and pitch-rate data collected while manually driving the vehicle in areas at Citra known to be either Rugged or Very-Rugged. Once in the desert, the vehicle was again driven (or allowed to drive itself) in Rugged and Very-Rugged areas and the findings of the Terrain-Specialist were monitored in real-time from the chase vehicle. The threshold parameters were iteratively adjusted until the desired Terrain-State results were obtained.

The tuning of the Long- and Short-Range-Obstacle-Conditions was conducted in a test area specifically designed to create obstacle readings on the PLSS at the distances and heading offsets of interest. The test course allowed initial parameters to be set while the vehicle was either stationary or moving at very low speeds. In addition, the various travel speed settings were empirically established in a controlled testing environment. The final tuning of the Obstacle-Conditions was conducted by adjusting the threshold parameters while the vehicle was allowed to navigate at full speed and operate in real time.

The tuning of the combined decision logic was conducted by monitoring the Set Travel Speed output of the SA component from the chase vehicle. Most of this tuning took place while the vehicle was being tested in autonomous mode in support of other sensors and components.

The presence of the SA component provided a beneficial speed control oversight to the overall operation of the vehicle and achieved a balance of conservatism (not too fast) and pace (not too slow).

Table B-1. Environmental Sensor.

Sensor Name	Inputs to SA component	Legal Values of Input
Planar Ladar Smart Sensor	Range Data Array	Hit distance for each degree
	Sensor Health Flag	True | False

Table B-2. Vehicle Sensor.

Sensor Name	Inputs to SA component	Legal Values of Input
Velocity State Sensor	Roll-Rate	-32.767 to +32.767 rad/s
Velocity State Sensor	Pitch-Rate	-32.767 to +32.767 rad/s

Table B-3. Obstacle Specialist.

Condition Name	Condition Test	Legal Values of Condition
Long-Range-Obstacle	Object detected within a heading cone of ± 3° at a distance of ≥ 0.1m and < 80m ➔ Present	Absent | Present
Short-Range-Obstacle	Object detected within a heading cone of ± 3° at a distance of ≥ 0.1m and < 50m ➔ Present	Absent | Present

Table B-4. Terrain Specialist.

State Name	State Test	Legal Values of State
Terrain-State	Accumulated roll-rate or pitch-rate above user-defined thresholds for "rugged" and "very-rugged"	Smooth | Rugged | Very Rugged

Table B-5. Decision Broker.

Decision Name	Decision Logic	Legal Values of Decision
Set Travel Speed	IF Long-Range-Obstacle is Absent AND Short-Range-Obstacle is Absent AND Terrain-State is Smooth ➔ Travel-Speed is Max-Speed IF Long-Range-Obstacle is Present AND Short-Range-Obstacle is Absent AND Terrain-State is Smooth ➔ Travel-Speed is Mid-Speed IF Terrain-State is Very-Rugged ➔ Travel-Speed is Min-Speed All other combinations ➔ Travel-Speed is Obstacle-Avoidance -Speed	Max-Speed | Mid-Speed | Obstacle-Avoidance-Speed | Min-Speed

APPENDIX C
KNOWLEDGE REPRESENTATION WORK PRODUCTS FOR THE ADAPTIVE PLANNING FRAMEWORK REFERENCE IMPLEMENTATION

This appendix contains the Behavior Use Cases, the Findings Worksheets, and the Decision Broker Protocol Worksheets used to define the Reference Implementation of the Adaptive Planning Framework.

Roadway Navigation Behavior Use Case

Description: The focus of this behavior is finding the most navigable obstacle-free terrain in the general direction of the goal; it has no understanding of lanes or lane markings, so it will tend to take the center of the road surface (note that this behavior was originally designed for use on one-way, often off-road, situations).

Assumptions: A drivable surface exists. The NAVIGATOR vehicle is the one being controlled. A Traversability Grid that follows the CIMAR ICD is available.

Constraints: Obey the speed limit, avoid obstacles, and minimize 'cost'. The planning behavior needs to operate while the component is in the STANDBY State.

Entry Conditions: GPOS connection, VSS connection, SARB connection, PD Current Wrench connection, PD Current Status connection, Control of the PD, Valid Path File

Exit Conditions: Final goal node achieved (see Step 8).

Inputs Consumed:

- Report Global Position message (from GPOS)
- Report Velocity State message (from VSS)
- Report Traversability Grid message (from SARB)
- Report Wrench Effort message (from PD)
- Report Discrete Devices (Gear) message (from PD)
- Report Component Status message (from PD)
- Path File (non-JAUS)
- Resume message (from SSC)
- Standby message (from SSC)

Outputs Produced:

- Assume Control of PD message
- Set Wrench Effort message (to PD)
- Set Discrete Devices (Gear) message (to PD)
- Report Component Status message (RN to SSC)
- Report Traversability Grid message (info only, not used for control/behavior)

Steps for Roadway Navigation Behavior:

Step #	Action	Contingency Action
1	Verify required Service Connections and Take Control of PD	Go to EMERGENCY State and attempt to reinitialize
2	Verify PD is in READY State	Go to STANDBY State
3	Roll Grid (if necessary based on change in position) and set new Current Vehicle State (x, y, yaw, speed, ϕ effort, goal row/column)	
4	Calculate desired speed and ϕ effort	
5	Verify goal node is in bounds	Stop the vehicle (Set desired speed to 0 m/sec)
6	Copy SARB input grid into RD planning grid and dilate cell values such that each cell takes on the worst value of its neighbors	
7	Verify cell value of current position is not 0 or 2 (Out Of Bounds or Absolutely Non-traversable)	Set OOB status/Collision status to TRUE; if current speed is less than Minimum Allowed Speed, set Stuck status to TRUE
8	Verify that there are path segments remaining in the path file	EXIT CONDITION MET: Stop the vehicle (Set desired speed to 0 m/sec, ϕ effort to 0)
9	Verify desired speed is greater than Minimum Allowed Speed	Apply the Receding Horizon algorithm with the desired speed = Minimum Allowed Speed to get the best steering angle, but then set the desired speed to 0 m/sec; GOTO Step 12
10	Apply Receding Horizon algorithm	Determine best ϕ effort (in spite of failure to find a viable solution) and set desired speed to a lower value for the next iteration (Note: this will gradually bring the vehicle to a stop if success is not achieved)
11	Determine command speed (min of DARPA Speed Limit, SSC max speed, RH-determined desired speed)	
12	Determine (dampened) command ϕ effort	
13	Convert command speed and command ϕ effort into Wrench	
14	Send wrench to PD	
15	Send TG to visualizer	
16	Repeat (go to Step 1)	

n-Point Turn Behavioral Use Case

Scenario Description: This behavior is used by the Decision Broker to reverse the direction of the vehicle when an erstwhile navigable route becomes blocked, causing the Mission Planner to place a goal node behind the vehicle. This scenario will stay in effect until it either solves the problem (thereby allowing the Decision Broker to reenter normal operations) or cannot move for an extended period.

Assumptions: This Use Case applies to situations that cannot be solved by the more sophisticated behaviors. There will be some mechanism available on the vehicle to discern where the boundaries of the drivable surfaces are located, either by sensing a curb, sensing a painted line, interpreting *a priori* data, etc.

Constraints: This behavior is targeted to be Reactive, so the intrinsic behaviors (e.g., obey the speed limit, avoid obstacles, and minimize 'cost') do not apply. This entire Use Case is only valid while n-Point Turn Behavior Specialist reports that nPTRecommendation = OK. The vehicle must be stopped when changing gears. A given Action should remain in effect for a configurable number of seconds (as long as it is safe), even if a preferred Action becomes available to avoid thrashing between actions.

Entry Conditions: The vehicle is stationary.

Exit Conditions: The vehicle is stationary.

Inputs Consumed:

- Close Range Safety Findings (Meta Data from closeRangeSafetySpecialist)
- Report Velocity State message (from VSS)
- Report Wrench Effort message (from PD)
- Report Discrete Devices (Gear) message (from PD)
- Report Component Status message (from PD)
- Resume message (from SSC)
- Standby message (from SSC)

Outputs Produced:

- Assume Control of PD message (to PD)
- Set Wrench Effort message (to PD)
- Set Discrete Devices (Gear) message (to PD)
- Report Component Status message (nPT to SSC)

Steps for n-Point Turn Behavior:

Step #	Action	Contingency Action
1	Verify required Service Connections and Take Control of PD	Go to EMERGENCY State and attempt to reinitialize
2	Verify PD is in READY State	Go to STANDBY State
3	Apply Reactive Behavior Model	

Reactive Behavior Model for n-Point Turn Behavior:

Priority	Action	Stimulus
1	Drive forward, full-left at Minimum Travel Speed	while forwardLeftSafeCondition is Present
2	Drive reverse, full-right at Minimum Travel Speed	while reverseRightSafeCondition is Present
3	Drive reverse, straight at Minimum Travel Speed	while reverseStraightSafeCondition is Present for up to 15 meters (configurable)
4	Stop - wait for 5 seconds (configurable) before reentering Action 3	while no Stimulus is Present (monitor for any available Stimulus)

Findings Worksheet

Specialist: Roadway Navigation Behavior

Finding: rnPlanningState **Type:** State

Possible Values: Succeeded | Failed | Goal Achieved

Rule(s)/Algorithm(s):

Element	Comments
IF rnPlanPathSuccess = TRUE THEN rnPlanningState = Succeeded	The value of rnPlanPathSuccess is returned by the recedingHorizonSearch() function in the RN component
IF rnPlanPathSuccess = FALSE THEN rnPlanningState = Failed	The value of rnPlanPathSuccess is returned by the recedingHorizonSearch() function in the RN component
IF rnPath->currentSegment = NULL THEN rnPlanningState = Goal Achieved	Perform this test LAST. rnPath is a data structure in the RN component

Findings Worksheet

Specialist: Roadway Navigation Behavior

Finding: rnMobilityState **Type:** State

Possible Values: Stuck | Blocked | Operational

Rule(s)/Algorithm(s):

Element	Comments
IF rnCurrentState.speedMps < RN_DEFAULT_MIN_SPEED_MPS AND rnCurrentState.desiredSpeedMps >= RN_DEFAULT_MIN_SPEED_MPS THEN rnMobilityState = Stuck	**Side Effect:** Set operationalTimeDamper = getTimeSeconds() + CLEAR_STUCK_TIMEOUT_SEC
IF rnPlanningState = Succeeded AND rnCurrentState.desiredSpeedMps > RN_DEFAULT_MIN_SPEED_MPS THEN rnMobilityState = Blocked	**Side Effect:** Set operationalTimeDamper = getTimeSeconds() + CLEAR_STUCK_TIMEOUT_SEC **NOTE:** This test must be executed AFTER recedingHorizonSearch() for the current iteration of the roadway navigation component
IF rnCurrentState.speedMps >= RN_DEFAULT_MIN_SPEED_MPS AND rnCurrentState.desiredSpeedMps >= RN_DEFAULT_MIN_SPEED_MPS AND getTimeSeconds() > operationalTimeDamper THEN rnMobilityState = Operational	

Findings Worksheet

Specialist: Roadway Navigation Behavior

Finding: rnRecommendation **Type:** Recommendation

Possible Values: OK | Faulted | Need New Plan

Rule(s)/Algorithm(s):

Element	Comments
IF rnMobilityState = Operational AND rnPlanningState = Succeeded THEN rnRecommendation = OK	
IF rnPlanningState = Goal Achieved THEN rnRecommendation = Need New Plan	
ELSE rnRecommendation = Faulted	

Findings Worksheet

Specialist: n-Point Turn Behavior

Finding: nPTRecommendation **Type:** Recommendation

Possible Values: OK | Unsafe | Blocked | Waiting

Rule(s)/Algorithm(s):

Element	Comments
IF forwardLeftSafeCondition = Present OR reverseRightSafeCondition = Present OR reverseStraightSafeCondition = Present THEN nPTRecommendation = OK	
IF forwardLeftSafeCondition = Unknown AND reverseRightSafeCondition = Unknown AND reverseStraightSafeCondition = Unknown THEN nPTRecommendation = Unsafe	**Side effect:** Stop the vehicle
IF forwardLeftSafeCondition = Absent AND reverseRightSafeCondition = Absent AND reverseStraightSafeCondition = Absent AND Counter > {BlockedCountMax} THEN nPTRecommendation = Blocked	**Side effect:** Stop the vehicle {} indicates configurable parameter Counter starts when all Safe Conditions become Absent
IF forwardLeftSafeCondition = Absent AND reverseRightSafeCondition = Absent AND reverseStraightSafeCondition = Absent AND Counter <= { BlockedCountMax } THEN nPTRecommendation = Waiting	**Side effect:** Stop the vehicle {} indicates configurable parameter Counter starts when all Safe Conditions become Absent

Findings Worksheet

Specialist: Close Range Safety

Finding: forwardLeftSafeCondition **Type:** Condition

Possible Values: Present | Absent | Unknown

Rule(s)/Algorithm(s):

Element	Comments
IF leftFrontSectorRangeMeters > {leftForwardTurnBufferDistanceMeters} AND centerFrontSectorRangeMeters > {centerForwardTurnBufferDistanceMeters} AND rightFrontSectorRangeMeters > {rightForwardTurnBufferDistanceMeters} THEN forwardLeftSafeCondition = Present	{} indicates configurable parameter Range information is Meta Data from LADAR Smart Sensor if Specialist is not co-resident Sector sizes should also be configurable (Sector sizes on LADAR Smart Sensor should be configured to match)
IF leftFrontSectorRangeMeters = UNKNOWN OR centerFrontSectorRangeMeters = UNKNOWN OR rightFrontSectorRangeMeters = UNKNOWN THEN forwardLeftSafeCondition = Unknown	#define UNKNOWN -1 Constructor should set default value of range as UNKNOWN
ELSE forwardLeftSafeCondition = Absent	

Findings Worksheet

Specialist: Close Range Safety

Finding: reverseRightSafeCondition **Type:** Condition

Possible Values: Present | Absent | Unknown

Rule(s)/Algorithm(s):

Element	Comments
IF leftRearSectorRangeMeters > {leftReverseTurnBufferDistanceMeters} AND centerRearSectorRangeMeters > {centerReverseTurnBufferDistanceMeters} AND rightRearSectorRangeMeters > {rightReverseTurnBufferDistanceMeters} THEN reverseRightSafeCondition = Present	{} indicates configurable parameter Range information is Meta Data from LADAR Smart Sensor if Specialist is not co-resident Sector sizes should also be configurable (Sector sizes on LADAR Smart Sensor should be configured to match)
IF leftRearSectorRangeMeters = UNKNOWN OR centerRearSectorRangeMeters = UNKNOWN OR rightRearSectorRangeMeters = UNKNOWN THEN reverseRightSafeCondition = Unknown	#define UNKNOWN -1 Constructor should set default value of range as UNKNOWN
ELSE reverseRightSafeCondition = Absent	

Findings Worksheet

Specialist: Close Range Safety

Finding: reverseStraightSafeCondition **Type:** Condition

Possible Values: Present | Absent | Unknown

Rule(s)/Algorithm(s):

Element	Comments
IF leftRearSectorRangeMeters > {leftStraightBufferDistanceMeters} AND centerRearSectorRangeMeters > {centerStraightBufferDistanceMeters} AND rightRearSectorRangeMeters > {rightStraightBufferDistanceMeters} THEN reverseStraightSafeCondition = Present	{} indicates configurable parameter Range information is Meta Data from LADAR Smart Sensor if Specialist is not co-resident Sector sizes should also be configurable (Sector sizes on LADAR Smart Sensor should be configured to match)
IF leftRearSectorRangeMeters = UNKNOWN OR centerRearSectorRangeMeters = UNKNOWN OR rightRearSectorRangeMeters = UNKNOWN THEN reverseStraightSafeCondition = Unknown	#define UNKNOWN -1 Constructor should set default value of range as UNKNOWN
ELSE reverseStraightSafeCondition = Absent	

Decision Broker Protocol Worksheet

Name of Protocol: Monitor/Select Behavior (Executive Behavior)

Goal of Protocol: Maintain the vehicle in a safe state while determining the desired behavior

Assumption(s): The vehicle will be stopped when transitioning behaviors

Input Parameter(s): Travel Speed and recommendations from any operational behavior

Entry Conditions: The vehicle should be fully stopped

Exit Conditions: The vehicle should be fully stopped

Wait State Timeout: 5 seconds

Travel Speed Tolerance: 0.05 mps

Protocol:

Action Steps	Contingency Steps
1. IF rnRecommendation = Need New Plan ➔ Execute Plan New Mission Protocol (future)	
2. IF rnRecommendation = OK AND RN is in Control ➔ Do nothing	2a. IF Wait is expired ➔ Execute RePlan Mission Protocol (future)
3. IF rnRecommendation != OK AND NPT in Control ➔ Do nothing	2b. IF Wait is expired AND nptRecommendation = OK ➔ Execute Exit from Roadway Navigation Behavior Protocol
4. IF rnRecommendation = OK AND NPT in Control ➔ Execute Exit from NPT Behavior Protocol	2c. IF nptRecommendation = OK AND NPT is NOT in Control AND RN is NOT in Control ➔ Execute Transition to N-Point Turn Behavior Protocol
5. IF rnRecommendation = OK AND NPT is NOT in Control AND RN is NOT in Control ➔ Execute Transition to Roadway Navigation Behavior Protocol	2d. ELSE do nothing
6. IF nptRecommendation != OK AND NPT in Control AND Wait is expired ➔ Execute Exit from N-Point Turn Behavior Protocol	

Decision Broker Protocol Worksheet

Name of Protocol: Transition to Roadway Navigation Behavior

Goal of Protocol: Cause the vehicle to safely begin executing its roadway navigation behavior

Assumption(s): This protocol will enter from and return to the Executive Behavior; entering the Ready state requires successfully taking control of the JAUS Primitive Driver

Input Parameter(s): Travel Speed, RN Behavior Specialist's Findings, RN Component Status

Entry Conditions: The vehicle should be fully stopped and SSC has control of the RN behavior

Exit Conditions: The vehicle should be fully stopped

Wait State Timeout: 1 second

Travel Speed Tolerance: 0.05 mps

Protocol:

Action Steps	Contingency Steps
1. Verify current speed = 0 mps	Set Travel Speed = 0 and Wait
2. Verify RN Specialist reports rnRecommendation = OK	Wait, Exit Protocol if Action Step still not satisfied
3. Place RN into Ready State	
4. Verify RN is in Ready State	Wait, Place RN into Ready State
5. Exit this Protocol	

Decision Broker Protocol Worksheet

Name of Protocol: Exit from Roadway Navigation Behavior

Goal of Protocol: Cause the vehicle to safely discontinue its roadway navigation behavior

Assumption(s): This protocol will enter from and return to the Executive Behavior

Input Parameter(s): Travel Speed, RN Behavior Specialist's Findings, RN Component Status

Entry Conditions: The vehicle should be fully stopped

Exit Conditions: The vehicle should be fully stopped

Wait State Timeout: 1 second

Travel Speed Tolerance: 0.05 mps

Protocol:

Action Steps	Contingency Steps
1. Set Travel Speed = 0 mps	
2. Verify current speed = 0 mps	Wait, Set Travel Speed = 0 mps
3. Place RN into Standby State	
4. Verify RN is in Standby State	Wait, Place RN into Standby State
5. Exit this Protocol	

Decision Broker Protocol Worksheet

Name of Protocol: Transition to n-Point Turn Behavior

Goal of Protocol: Cause the vehicle to safely begin executing its n-point turn behavior

Assumption(s): This protocol will enter from and return to the Executive Behavior; entering the Ready state requires successfully taking control of the JAUS Primitive Driver

Input Parameter(s): Travel Speed, nPT Behavior Specialist's Findings, nPT Component Status

Entry Conditions: The vehicle should be fully stopped and SSC has control of the NPT behavior

Exit Conditions: The vehicle should be fully stopped

Wait State Timeout: 1 second

Travel Speed Tolerance: 0.05 mps

Protocol:

Action Steps	Contingency Steps
1. Verify current speed = 0 mps	Set Travel Speed = 0 and Wait
2. Verify nPT Specialist reports nPTRecommendation = OK	Wait, Exit Protocol if Action Step still not satisfied
3. Place nPT into Ready State	
4. Verify nPT is in Ready State	Wait, Place nPT into Ready State
5. Exit this Protocol	

Decision Broker Protocol Worksheet

Name of Protocol: Exit from n-Point Turn Behavior

Goal of Protocol: Cause the vehicle to safely discontinue its n-point turn behavior

Assumption(s): This protocol will enter from and return to the Executive Behavior

Input Parameter(s): Travel Speed, nPT Behavior Specialist's Findings, nPT Component Status

Entry Conditions: The vehicle should be fully stopped

Exit Conditions: The vehicle should be fully stopped

Wait State Timeout: 1 second

Travel Speed Tolerance: 0.05 mps

Protocol:

Action Steps	Contingency Steps
1. Set Travel Speed = 0 mps	
2. Verify current speed = 0 mps	Set Travel Speed = 0 and Wait
3. Place nPT into Standby State	
4. Verify nPT is in Standby State	Wait, Place nPT into Standby State
5. Exit this Protocol	

APPENDIX D
JAUS META DATA TRANSFER INTERFACE CONTROL DOCUMENT

This appendix contains the initial release of the Interface Control Document used to incorporate the Adaptive Planning Framework into the JAUS messaging system in place on the NAVIGATOR.

NAVIGATOR Urban Challenge Architecture

Meta Data Transfer Interface Control Document

Version 1.0

September 26, 2006

Change Summary

1. Initial Release

System Overview

This document specifies a standardized method and format for transferring data that is not otherwise accommodated in a JAUS message. The two areas envisioned are Meta Data (or symbolic data), such as Findings of Adaptive Planning Framework Specialists and numerical data of interest but not found in an existing JAUS message, such as the number of active sensors currently in use by the Smart Arbiter. The scope of this document is only for the Team Gator Nation Urban NAVIGATOR vehicles and is not intended for general use for JAUS-based systems at this time. The purpose of the method presented herein is to allow all components to report their findings or other information in a common format that is flexible, simple and efficient. By conforming to this common format, a high degree of modularity is achieved such that components and Specialists may be added and removed with minor impact on any other components or Specialists.

This approach requires that each component that produces or consumes Meta Data do so using the messaging formats described later in this document. Note: to utilize the time stamping of the message content included with the message, each component must synchronize its internal clock with that of the Subsystem Commander (or other agreed-upon source).

Concept of Operations

The Meta Data Reporting concept described here requires every component that can provide Meta Data (the "publishers") be able to process an inbound Meta Data Changed Event Setup Message, respond with a Meta Data Changed Event Confirmation Message and then create and distribute a Report Meta Data Message containing those Meta Data Elements that have changed to those components that have subscribed.

The rules and tolerances for judging that a numerical element has changed enough to merit inclusion in a report is solely the responsibility of the publisher. In other words, the onus is on the publisher rather than the subscriber to decide when a Meta Data Element ought to be published.

It follows that every component that uses Meta Data (the "subscribers") must be able to produce a Meta Data Changed Event Setup Message, and send one to every component that publishes data of interest. Likewise, it must be able to process inbound Meta Data Changed Event Confirmation Messages and Report Meta Data Messages.

The simplicity of this approach places greater demands during the design phase since every component that needs Meta Data must be explicitly programmed to subscribe to it.

The anticipated response is simply for the publisher to add the subscriber to its list of subscribers and then to send out a Meta Data Changed Event Confirmation Message so that the subscriber knows that the publisher has added it to its list of subscribers (and it can stop trying to set it up) followed by an ongoing series of Report Meta Data messages to its subscriber list whenever any of its Meta Data changes. These Reports should only include those Meta Data Elements that have experienced a change. It is also considered good practice to periodically send out a "key"

report containing a complete set of Meta Data values in case an earlier update message was not properly delivered to a subscriber. Future releases may add more intelligence to this process by allowing a component to subscribe to a specific Meta Data Element rather than all Meta Data provided by a given publisher.

Summary of Behaviors

Subscriber Event	Subscriber Behavior	Publisher Event	Publisher Behavior
Startup	Send Meta Data Changed Event Setup Message (keep sending periodically until confirmed)	Receive Meta Data Changed Event Setup Message	Add subscriber to distribution list, Send Meta Data Changed Event Confirmation Message
Receive Meta Data Changed Event Confirmation Message	Stop sending setup message	The value of one or more Meta Data Elements changes significantly	Send Meta Data Report Message containing only changed Meta Data Elements to subscriber list
Receive Meta Data Report Message	Examine Meta Data Element Name and apply new value if it's of interest		

Findings from Situation Assessment Specialists and Behavior Specialists

The software Specialists called for by the Adaptive Planning Framework share their results in the form of "Findings." The major impetus for creating this ICD and its messages is to provide a JAUS-compatible mechanism to enable these Specialists to send and receive these Findings. Findings can be in the form of Conditions, States, Events, or Recommendations, as follows:

Conditions:

- Can only have a value of "Present" or "Absent" and must be conclusively proven to be "Present" at each iteration.

States:

- From a pre-defined list of possible values; a state transition must be conclusively proven.

Events:

- Either "True" or "False" based on the occurrence of the event.

Recommendations:

- From a pre-defined list of possible values; a state transition must be conclusively proven.

General Data and Information

This message set can also be used to convey data and information that need to be transferred from one component to another but are not encapsulated in an available JAUS message. Data can be of any valid JAUS data type and its associated units of measure should be suffixed to its name. For example, if one wanted to use this message to allow a Smart Arbiter to divulge how many Smart Sensors were currently being used to produce its output grid, one might name the meta data element "smartArbiterInputCount" and give it data type of "unsignedShort." Information related to Specialists' Findings is typically going to be conveyed as a string.

Summary of Parameters

This ICD shall follow the enumeration scheme created for the Variant data type, which in turn closely follows the Type Code concept introduced in the Payload Interface ICD.

Assigned Data Type Codes

Data Type	Code	Suggested Define
Reserved	0	
Short Integer (2 bytes)	1	JAUS_VARIANT_TYPE_SHORT
Integer (4 bytes)	2	JAUS_VARIANT_TYPE_INTEGER
Long Integer (8 bytes)	3	JAUS_VARIANT_TYPE_LONG
Byte (1 byte)	4	JAUS_VARIANT_TYPE_BYTE
Unsigned Short (2 bytes)	5	JAUS_VARIANT_TYPE_U_SHORT
Unsigned Integer (4 bytes)	6	JAUS_VARIANT_TYPE_U_INTEGER
Unsigned Long (8 bytes)	7	JAUS_VARIANT_TYPE_U_LONG
Float (4 bytes)	8	JAUS_VARIANT_TYPE_FLOAT
Long Float (8 bytes)	9	JAUS_VARIANT_TYPE_DOUBLE
String {Length (unsigned short) followed by the Null Terminated ASCII string}	19	JAUS_VARIANT_TYPE_STRING
Unsigned Byte Tuple	20	JAUS_VARIANT_TYPE_U_BYTE_TUPLE
Unsigned Short Tuple	21	JAUS_VARIANT_TYPE_U_SHORT_TUPLE
Unsigned Integer Tuple	22	JAUS_VARIANT_TYPE_U_INTEGER_TUPLE

Message Set Specifications

These messages are experimental and should be tagged as such in their message header.

Code D090h: Meta Data Changed Event Setup

This message is used in lieu of a traditional Query Message to request a specific component to send its Report Meta Data messages. It has only one field to indicate whether the request is being started or cancelled.

Field #	Name	Type	Units	Interpretation
1	Setup Flag	Byte	N/A	0 – Stop sending Meta Data Reports 1 – Start sending Meta Data Reports

Code E090h: Meta Data Changed Event Confirmation

This message is used to confirm to a specific requesting component that it will begin receiving Report Meta Data messages. It has only one field to indicate whether the request is being confirmed, rejected or cancelled.

Field #	Name	Type	Units	Interpretation
1	Confirmation Flag	Byte	N/A	0 – Stop sending Meta Data Reports confirmed 1 – Start sending Meta Data Reports confirmed 2 – Request rejected

Code E091h: Report Meta Data

This message is used to report Meta Data. The message contains one or more meta data elements within the purview of the originating component/service.

Field #	Name	Type	Units	Interpretation
1	Number of Data Elements	Unsigned short	N/A	How many meta data elements to expect, n
2	Name of 1st Meta Data Element	String	N/A	Null Terminated ASCII string
3	Time Stamp	Unsigned Integer	N/A	Bits 0-9: milliseconds, range 0...999 Bits 10-15: Seconds, range 0...59 Bits 16 – 21: Minutes, range 0...59 Bits 22-26: Hour (24 hour clock), range 0..23 Bits 27-31: Day, range 1…31
4	Data Type Code	Byte	N/A	See Assigned Variant Type Codes Table
5	Value	Variant	N/A	Current value of the meta data to be reported
…				
4n - 2	Name of last Meta Data Element	String	N/A	Null Terminated ASCII string
4n - 1	Time Stamp	Unsigned Integer	N/A	Bits 0-9: milliseconds, range 0...999 Bits 10-15: Seconds, range 0...59 Bits 16 – 21: Minutes, range 0...59 Bits 22-26: Hour (24 hour clock), range 0..23 Bits 27-31: Day, range 1…31
4n	Data Type Code	Byte	N/A	See Assigned Variant Data Type Codes Table
4n + 1	Value	Variant	N/A	Current value of the meta data to be reported

APPENDIX E
REPRESENTATIVE TEST LOGS

This appendix contains the abridged log files of the Subsystem Commander, the Roadway Navigation and the n-Point Turn components for the final test at UF's Research Farm near Citra, Florida. They have been edited to remove repetitive entries or entries not material to the research results. Note that the gear indexes are 0 for Park, 1 for Drive, and 129 for Reverse and that the component state indexes are 1 for Ready and 2 for Standby.

```
CIMAR /opt/CIMAR/NaviGator-85/SSC-32/bin/ssc Log -- 01-01-2004 04:59:35
01-01-2004 04:59:35: SSC: Got message: JAUS_QUERY_IDENTIFICATION
01-01-2004 04:59:35: Ready to create GPOS SC.
01-01-2004 04:59:35: ssc: Sent GPOS SC Request
01-01-2004 04:59:35: ssc: Sent VSS SC Request
01-01-2004 04:59:35: ssc: Sent MDE Event Setup Request to 129.100.23.1
01-01-2004 04:59:35: ssc: attempting to establish MDE event setup with RN
01-01-2004 04:59:35: ssc: NPT behavior found at address: 129.100.23.1
01-01-2004 04:59:35: SSC: Got message: UNDEFINED MESSAGE: E090
01-01-2004 04:59:35: ssc msg procsr: rec'd confirmation - setting nptMDEventSetup to true
01-01-2004 04:59:35: SSC: Got message: UNDEFINED MESSAGE: E091
01-01-2004 04:59:35: ssc msg procsr: new n-Point Turn Recommendation is: OK
01-01-2004 04:59:35: ssc: Requesting control of 129.0.15.0
01-01-2004 04:59:36: SSC: Got message: UNDEFINED MESSAGE: D090
01-01-2004 04:59:36: ssc: received MD Event setup request from 129.100.23.1
01-01-2004 04:59:36: ssc: sent MD Event setup confirmation
01-01-2004 04:59:36: ssc: preparing to send meta data report
01-01-2004 04:59:36: ssc: sending meta data report to 129.100.23.1
01-01-2004 04:59:36: ssc: attempting to establish MDE event setup with RN
01-01-2004 04:59:36: ssc: RN address FOUND - attempting to establish MDE event setup with RN
01-01-2004 04:59:36: ssc: Sent MDE Event Setup Request to 129.100.15.1
01-01-2004 04:59:36: SSC: Got message: UNDEFINED MESSAGE: E090
01-01-2004 04:59:36: ssc msg procsr: rec'd confirmation - setting rnMDEventSetup to true
01-01-2004 04:59:36: SSC: Got message: UNDEFINED MESSAGE: E091
01-01-2004 04:59:36: ssc msg procsr: new Roadway Navigation Mobility State is: Operational
01-01-2004 04:59:36: ssc msg procsr: new Roadway Navigation Planning State is: Succeeded
01-01-2004 04:59:36: ssc msg procsr: new Roadway Navigation Recommendation is: OK
01-01-2004 04:59:36: RN rec: OK; NPT rec: Unsafe; RN state: 2; NPT state: 2
01-01-2004 04:59:36: SSC: Got message: JAUS_CONFIRM_COMPONENT_CONTROL
01-01-2004 04:59:36: ssc: Confirmed control of RN
01-01-2004 04:59:37: ssc: Requesting control of 129.100.23.1
01-01-2004 04:59:37: RN rec: OK; NPT rec: Unsafe; RN state: 2; NPT state: 2
01-01-2004 04:59:37: SSC: Got message: JAUS_CONFIRM_COMPONENT_CONTROL
01-01-2004 04:59:37: ssc: Confirmed control of NPT
01-01-2004 04:59:39: SSC calling Resume RN Behavior
01-01-2004 04:59:39: ssc: Sent Resume Request to RN
01-01-2004 04:59:45: SSC: Got message: UNDEFINED MESSAGE: E091
01-01-2004 04:59:45: ssc msg procsr: new Roadway Navigation Mobility State is: Blocked
01-01-2004 04:59:45: ssc msg procsr: new Roadway Navigation Planning State is: Succeeded
01-01-2004 04:59:45: ssc msg procsr: new Roadway Navigation Recommendation is: Faulted
01-01-2004 04:59:45: RN rec: Faulted; NPT rec: OK; RN state: 1; NPT state: 2
```

```
01-01-2004 04:59:45: SSC calling Pause RN Behavior
01-01-2004 04:59:45: ssc: Sent Standby Request to rn
01-01-2004 04:59:45: RN rec: Faulted; NPT rec: OK; RN state: 2; NPT state: 2
01-01-2004 04:59:45: SSC calling Resume NPT Behavior
01-01-2004 04:59:45: ssc: Sent Resume Request to npt
01-01-2004 05:00:48: RN rec: Faulted; NPT rec: OK; RN state: 2; NPT state: 1
01-01-2004 05:02:01: SSC: Got message: UNDEFINED MESSAGE: E091
01-01-2004 05:02:01: ssc msg procsr: new Roadway Navigation Mobility State is: Operational
01-01-2004 05:02:01: ssc msg procsr: new Roadway Navigation Planning State is: Succeeded
01-01-2004 05:02:01: ssc msg procsr: new Roadway Navigation Recommendation is: OK
01-01-2004 05:02:01: SSC calling Standby NPT Behavior
01-01-2004 05:02:01: ssc: Sent Standby Request to npt
01-01-2004 05:02:01: RN rec: OK; NPT rec: OK; RN state: 2; NPT state: 1
01-01-2004 05:02:08: RN rec: OK; NPT rec: OK; RN state: 2; NPT state: 2
01-01-2004 05:02:08: SSC calling Resume RN Behavior
01-01-2004 05:02:08: ssc: Sent Resume Request to RN
```

```
CIMAR ./bin/rn Log -- 01-01-2004 04:59:36
01-01-2004 04:59:36: rn: Sent GPOS SC Request
01-01-2004 04:59:36: rn: Sent VSS SC Request
01-01-2004 04:59:36: rn: PD Found at address: 129.2.33.1
01-01-2004 04:59:36: rn: Sent PD SC Shifter Request
01-01-2004 04:59:36: rn: Sent PD Wrench SC Request
01-01-2004 04:59:36: rn: Sent PD Status SC Request
01-01-2004 04:59:36: RN: Got message: UNDEFINED MESSAGE: D090
01-01-2004 04:59:36: rn: received MD Event setup request
01-01-2004 04:59:36: rn: sent MD Event setup confirmation
01-01-2004 04:59:36: rn: preparing to send meta data report
01-01-2004 04:59:36: rn: sending meta data report to 129.100.32.1
01-01-2004 04:59:46: RN: Got message: JAUS_RESUME
01-01-2004 04:59:46: RN: ssc has commanded Ready State
01-01-2004 04:59:46: rn: Requesting control of PD
01-01-2004 04:59:46: RN: Got message: JAUS_REJECT_COMPONENT_CONTROL
01-01-2004 04:59:46: rn: Lost control of PD
01-01-2004 04:59:46: RN: Got message: JAUS_REPORT_HEARTBEAT_PULSE
01-01-2004 04:59:47: RN: Got message: JAUS_REPORT_HEARTBEAT_PULSE
01-01-2004 04:59:48: ForceBlockedFlag changed to: True
01-01-2004 04:59:48: rn: preparing to send meta data report
01-01-2004 04:59:48: rn output changed flag = 0
01-01-2004 04:59:48: rn: sending meta data report to 129.100.32.1
01-01-2004 04:59:48: RN: Got message: JAUS_REPORT_HEARTBEAT_PULSE
01-01-2004 04:59:49: RN: Got message: JAUS_REPORT_HEARTBEAT_PULSE
01-01-2004 04:59:50: RN: Got message: JAUS_REPORT_HEARTBEAT_PULSE
01-01-2004 04:59:51: ForceBlockedFlag changed to: False
01-01-2004 04:59:51: rn: preparing to send meta data report
01-01-2004 04:59:51: rn output changed flag = 0
01-01-2004 04:59:51: rn: sending meta data report to 129.100.32.1
01-01-2004 04:59:51: rn: Requesting control of PD
01-01-2004 04:59:51: RN: Got message: JAUS_CONFIRM_COMPONENT_CONTROL
01-01-2004 04:59:51: rn: Confirmed control of PD
01-01-2004 04:59:51: RN: switching to Ready
01-01-2004 04:59:51: Set to DRIVE.
01-01-2004 04:59:51: Current Gear: 0
01-01-2004 04:59:51: requested gear: 1
01-01-2004 04:59:51: RN: Got message: JAUS_RESUME
01-01-2004 04:59:51: Set to DRIVE.
01-01-2004 05:00:32: Current Gear: 1
01-01-2004 05:00:32: requested gear: 1
```

```
01-01-2004 05:00:40: ForceBlockedFlag changed to: True
01-01-2004 05:00:40: rn: preparing to send meta data report
01-01-2004 05:00:40: rn: sending meta data report to 129.100.32.1
01-01-2004 05:00:40: RN: Got message: JAUS_STANDBY
01-01-2004 05:00:40: RN: ssc has commanded Standby State
01-01-2004 05:00:43: gear is: 1
01-01-2004 05:00:43: gear is: 1 trying to shift to Park
01-01-2004 05:00:43: Current Gear: 1
01-01-2004 05:00:43: requested gear: 0
01-01-2004 05:00:47: gear is: 0
01-01-2004 05:00:47: RN is releasing control of PD (SSC)
01-01-2004 05:00:47: RN: switching to standby
01-01-2004 05:00:47: RN: Got message: JAUS_TERMINATE_SERVICE_CONNECTION
01-01-2004 05:02:08: RN: Got message: JAUS_RESUME
01-01-2004 05:02:08: RN: ssc has commanded Ready State
01-01-2004 05:02:08: rn: Requesting control of PD
01-01-2004 05:02:08: RN: Got message: JAUS_CONFIRM_COMPONENT_CONTROL
01-01-2004 05:02:08: rn: Confirmed control of PD
01-01-2004 05:02:08: RN: Got message: JAUS_CREATE_SERVICE_CONNECTION
01-01-2004 05:02:08: RN: switching to Ready
01-01-2004 05:02:08: Set to DRIVE.
01-01-2004 05:02:08: Current Gear: 0
01-01-2004 05:02:08: requested gear: 1
01-01-2004 05:02:12: Current Gear: 1
01-01-2004 05:02:12: requested gear: 1
```

```
CIMAR ./bin/npt_test Log -- 01-01-2004 04:59:34
01-01-2004 04:59:34: npt_cmpt: Got message: JAUS_CREATE_SERVICE_CONNECTION from 129.10.11.1
01-01-2004 04:59:34: npt_cmpt: Sending: JAUS_CREATE_SERVICE_CONNECTION to default processor
01-01-2004 04:59:34: npt_cmpt: Sent VS SC Request
01-01-2004 04:59:34: npt_cmpt: Sent PD SC Wrench Request
01-01-2004 04:59:34: npt_cmpt: Sent PD SC Shifter Request
01-01-2004 04:59:34: npt_cmpt: Sent PD Status SC Request
01-01-2004 04:59:34: npt_cmpt: PD Found at address: 129.2.33.1
01-01-2004 04:59:36: npt_cmpt: Got message: UNDEFINED MESSAGE: E090 from 129.100.32.1
01-01-2004 04:59:36: npt_cmpt msg procsr: rec'd confirmation - setting sscMDEventSetup to true
01-01-2004 04:59:38: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 04:59:38: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 04:59:38: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 04:59:38: npt_cmpt msg procsr: forwardLeftSafeCondition is: Absent
01-01-2004 05:00:47: npt_cmpt: Got message: JAUS_RESUME from 129.100.32.1
01-01-2004 05:00:47: Trying to Resume (DB)
01-01-2004 05:00:47: npt_cmpt: Sent Control Request
01-01-2004 05:00:48: npt_cmpt: Recieved control of PD
01-01-2004 05:00:48: npt_cmpt: Entering Ready State
01-01-2004 05:00:48: Current Gear: 0
01-01-2004 05:00:48: requested gear: 129
01-01-2004 05:00:50: Current Gear: 129
01-01-2004 05:00:50: requested gear: 129
01-01-2004 05:00:54: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:00:54: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:00:54: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:00:54: npt_cmpt msg procsr: forwardLeftSafeCondition is: Present
01-01-2004 05:00:54: Current Gear: 129
01-01-2004 05:00:54: requested gear: 1
01-01-2004 05:00:58: Current Gear: 1
01-01-2004 05:00:58: requested gear: 1
01-01-2004 05:01:04: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:01:04: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:01:04: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:01:04: npt_cmpt msg procsr: forwardLeftSafeCondition is: Absent
01-01-2004 05:01:08: Current Gear: 1
01-01-2004 05:01:08: requested gear: 129
01-01-2004 05:01:11: Current Gear: 129
01-01-2004 05:01:11: requested gear: 129
01-01-2004 05:01:15: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:01:15: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
```

```
01-01-2004 05:01:15: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:01:15: npt_cmpt msg procsr: forwardLeftSafeCondition is: Present
01-01-2004 05:01:16: Current Gear: 129
01-01-2004 05:01:16: requested gear: 1
01-01-2004 05:01:19: Current Gear: 1
01-01-2004 05:01:19: requested gear: 1
01-01-2004 05:01:27: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:01:27: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:01:27: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:01:27: npt_cmpt msg procsr: forwardLeftSafeCondition is: Absent
01-01-2004 05:01:30: Current Gear: 1
01-01-2004 05:01:30: requested gear: 129
01-01-2004 05:01:33: Current Gear: 129
01-01-2004 05:01:33: requested gear: 129
01-01-2004 05:01:39: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:01:39: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:01:39: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:01:39: npt_cmpt msg procsr: forwardLeftSafeCondition is: Present
01-01-2004 05:01:40: Current Gear: 129
01-01-2004 05:01:40: requested gear: 1
01-01-2004 05:01:43: Current Gear: 1
01-01-2004 05:01:43: requested gear: 1
01-01-2004 05:01:50: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:01:50: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:01:50: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:01:50: npt_cmpt msg procsr: forwardLeftSafeCondition is: Absent
01-01-2004 05:01:54: Current Gear: 1
01-01-2004 05:01:54: requested gear: 129
01-01-2004 05:01:56: Current Gear: 129
01-01-2004 05:01:56: requested gear: 129
01-01-2004 05:02:00: npt_cmpt: Got message: UNDEFINED MESSAGE: E091 from 129.100.32.1
01-01-2004 05:02:00: npt_cmpt msg procsr: new rearStraightSafeCondition id: Present
01-01-2004 05:02:00: npt_cmpt msg procsr: new rearRightSafeCondition is: Present
01-01-2004 05:02:00: npt_cmpt msg procsr: forwardLeftSafeCondition is: Present
01-01-2004 05:02:01: Current Gear: 129
01-01-2004 05:02:01: requested gear: 1
01-01-2004 05:02:01: npt_cmpt: Got message: JAUS_STANDBY from 129.100.32.1
01-01-2004 05:02:01: Trying to go to Standby (DB)
01-01-2004 05:02:01: Brake effort: 68
01-01-2004 05:02:02: Current Gear: 129
01-01-2004 05:02:02: requested gear: 0
```

```
01-01-2004 05:02:03: Current Gear: 1
01-01-2004 05:02:03: requested gear: 0
01-01-2004 05:02:08: npt_cmpt: Entering Standby State
01-01-2004 05:02:08: Current Gear: 0
01-01-2004 05:02:08: requested gear: 0
01-01-2004 05:02:08: npt_cmpt: Got message: JAUS_TERMINATE_SERVICE_CONNECTION from 129.2.33.1
01-01-2004 05:02:08: npt_cmpt: Sending: JAUS_TERMINATE_SERVICE_CONNECTION to default processor
01-01-2004 05:02:42: Terminating PD Status service connection
```

LIST OF REFERENCES

Aksit, M. (2002). *Software Architectures and Component Technology*, Kluwer Academic Press, Boston.

Balakirsky, S., and Lacaze, A. (2000). "World Modeling and Behavior Generation for Autonomous Ground Vehicle." *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, CA, 1201-1206.

Balakirsky, S., and Scrapper, C. (2004). "Knowledge representation and planning for on-road driving." *Robotics and Autonomous Systems*, 49(2004), 57-66.

Barbera, A., Albus, J., Messina, E., Schlenoff, C., and Horst, J. (2004a). "How Task Analysis Can Be Used to Derive and Organize the Knowledge For the Control of Autonomous Vehicles." *Proceedings of the 2004 AAAI Spring Symposium Series on Knowledge Representation and Ontology for Autonomous Systems*, Palo Alto, CA, 67-78.

Barbera, A., Messina, E., Huang, H.-M., Schlenoff, C., and Balakirsky, S. (2004b). "Software Engineering for Intelligent Control Systems." *Kunstiliche Intelligenz (an Artificial Intelligence journal)*, 22-26.

Batavia, P. H., and Nourbakhsh, I. (2000). "Path Planning for the Cye Personal Robot." *Proceedings of International Conference on Intelligent Robots and Systems, 2000 (IROS 2000)*, Takamatsu, Japan, 15-20.

Brooks, R. A. (1986). "A robust layered control system for a mobile robot." *IEEE Journal of Robotics and Automation*, 2(1), 14-23.

Crane, C. D., Armstrong, D. G., Touchton, R., Galluzzo, T., Solanki, S., Lee, J., Kent, D., Ahmed, M., Montane, R., Ridgeway, S., Velat, S., Garcia, G., Griffis, M., Gray, S., Washburn, J., and Routson, G. (2006). "Team CIMAR's NaviGATOR: An Unmanned Ground Vehicle for Application to the 2005 DARPA Grand Challenge." *Journal of Field Robotics*, 23(8), 599-623.

Erol, K., Hendler, J., and Nau, D. (1994). "Semantics for Hierarchical Task-Network Planning." *CS-TR-3239, UMIACS-TR-94-31, ISR-TR-95-9*, University of Maryland, Institute for Advanced Computer Studies College Park, MD.

Franklin, S., and Graesser, A. (1996). "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." *Third International Workshop on Agent Theories, Architectures, and Languages*, Budapest, Hungary, 21-35.

Hassan, H., Simo, J., and Crespo, A. (2001). "Flexible real-time mobile robotic architecture based on behavioural models." *Engineering Applications of Artificial Intelligence*, 14(5), 685-702.

Hayes-Roth, B. (1995). "An Architecture for Adaptive Intelligent Systems." *Artificial Intelligence*, 72, 329-365.

Hillenbrand, J., Kroschel, K., and Schmid, V. (2005). "Situation assessment algorithm for a collision prevention assistant." *Intelligent Vehicle Symposium, 2005*, Las Vegas, NV, 459-465.

Hoffman, H. (2004). "Adaptive Planning Overview." Presented at Military Operations Research Society, Capabilities-Based Planning: The Road Ahead, Alexandria, Virginia.

Hoffman, R. R., and Yates, J. F. (2005). "Decision(?)Making(?)." *IEEE Intelligent Systems*, 20(4), 76-83.

Howard, C., and Stumptner, M. (2005). "Probabilistic reasoning techniques for situation assessments." *Third International Conference on Information Technology and Applications, 2005 (ICITA '05)* Sydney, Australia, 383-386.

Howe, D. (2005). "The Free On-line Dictionary of Computing." http://foldoc.org/foldoc.cgi?query=ontology, last accessed October, 2006

JAUS-OPC. (2005). "OPC 2.75 Interface Control Document, version 1.0, Payload Interface section." JAUS Working Group - OCU & Payloads Committee.

JAUS. (2005). "JAUS Reference Architecture, version 3.2." JAUS Working Group.

Karacapilidis, N., and Papadias, D. (2001). "Computer supported argumentation and collaborative decision making: the HERMES system." *Information Systems*, 26(4), 259-277.

Lacaze, A. (2002). "Hierarchical Planning Algorithms." Presented at SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL.

Murphy, K., Abrams, M., Balakirsky, S., Coombs, D., Hong, T., Legowik, S., Chang, T., and Lacaze, A. (2000). "Intelligent Control for Unmanned Vehicles." Presented at World Automation Conference (WAC 2000), Maui, HI.

Musliner, D. J. (2001). "Imposing Real-Time Constraints on Self-Adaptive Controller Synthesis." *Lecture Notes in Computer Science*, 1936, 143-160.

Musliner, D. J., Durfee, E. H., and Shin, K. G. (1995). "World Modeling for the Dynamic Construction of Real-Time Control Plans." *Artificial Intelligence*, 74(1), 83-127.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, San Francisco.

NIST (National Institute of Standards and Technology). (1992). "A Real-Time Control System Methodology for Developing Intelligent Control Systems." *NISTIR 4936*, National Institute of Standards and Technology.

NIST (National Institute of Standards and Technology). (2002). "4D/RCS: A Reference Model Architecture for Unmanned Vehicle System, Version 2.0." *NISTIR 6910*, National Institute of Standards and Technology.

Panzarasa, P., Jennings, N. R., and Norman, T. J. (2002). "Formalising Collaborative Decision-Making and Practical Reasoning in Multi-agent Systems." *Journal of Logic and Computation*, 12(1), 55-117.

Payton, D. W., Rosenblatt, J. K., and Keirsey, D. M. (1990). "Plan Guided Reaction." *IEEE Transaction on Systems, Man, and Cybernetics* 20(6), 1370-1382.

Pirjanian, P. (1997). "An Overview of System Architecture for Action Selection in Mobile Robotics." Laboratory of Image Analysis, Aalborg University, Aalborg, Denmark.

Pritchett, W. (2002). "A Domain Framework for Intelligent Ground Combat Vehicle Systems." Presented at Software Technology Conference (STC 2002), TACOM (Tank-automotive & Armaments COMmand), Salt Lake City, UT.

Rauenbusch, T. W., and Grosz, B. J. (2003). "A Decision Making Procedure for Collaborative Planning." Presented at AAMAS'03 - Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia.

Reichard, K. M., and Crow, E. C. (2005). "Intelligent Self-Situational Awareness for Unmanned and Robotic Platforms." AUVSI Unmanned Systems Conference, Baltimore, MD.

Robotic Systems Technology. (1998). "Demo III Experiemental Unmanned Vehicle (XUV) Program Autonomous Mobility Requirements Analysis." prepared for TACOM.

Rosenblatt, J. K. (1997). "DAMN: A Distributed Architecture for Mobile Navigation," Dissertation, Carnegie Mellon, Pittsburgh.

Rosenblatt, J. K. (2000). "Optimal Selection of Uncertain Actions by Maximizing Expected Utility." *Autonomous Robots*, 9(1), 17-25.

Schlenoff, C., Balakirsky, S., Uschold, M., Provine, R., and Smith, S. (2003). "Using ontologies to aid navigation planning in autonomous vehicles." *The Knowledge Engineering Review*, 18(3), 243-255.

Schlenoff, C., Madhavan, R., and Barbera, T. (2004). "A Hierarchical, Multi-Resolutional Moving Object Prediction Approach for Autonomous On-Road Driving." *2004 ICRA Conference*, New Orleans, LA, 1956-1961.

Schlenoff, C., Washington, R., Barbera, T., and Manteuffel, C. (2005). "A standard intelligent system ontology." *Proceedings of SPIE - Unmanned Ground Vehicle Technology VII*, Orlando, FL, 46-56.

Scholtz, J., Antonishek, B., and Young, J. (2004). "Evaluation of a Human-Robot Interface: Development of a Situational Awareness Methodology." National Institute of Standards and Technology, last accessed October, 2006.

Seares, C. D. F. (1987). "Adaptive Mission Planning: Squeezing Out Greater Combat Capability." *Air University Review*, http://www.airpower.maxwell.af.mil/airchronicles/aureview/1987/seares2.html, last accessed October, 2006.

Touchton, R. A. (1988). "Reactor Emergency Action Level Monitor." *Artificial Intelligence and Other Innovative Computer Applications in the Nuclear Industry*, Majumdar, M. C., Majumdar, Debu, and Sackett, John I., ed., Plenum Press, New York, 189-197.

Touchton, R. A., Gunter, A. D., Wilson, K. M., Eldredge II, T. D., and Weaver, M. E. (1988). "Reactor Emergency Action Level Monitor Volume 1: REALM Technical Report." *NP-5719*, Electric Power Research Institute, Palo Alto.

US Army. (2003). "Army Universal Task List." *FM 7-15*, Washington, DC.

W3C. (2004). "Web Services Architecture." World Wide Web Consortium, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, last accessed October, 2006.

Weiss, K., Philipps, H., To, T. B., and Kirchner, A. (2005). "Environmental Perception and Situation Assessment for an Advanced Highway Assistant." *2005 IEEE Intelligent Vehicles Symposium Proceedings*, Las Vegas, NV, 472-477.

Winston, P., and Horn, B. (1989). *LISP: Third Edition*, Addison-Wesley, Reading, MA.

Yanco, H. A., and Drury, J. (2004). ""Where am I?" Acquiring situation awareness using a remote robot platform." *2004 IEEE International Conference on Systems, Man and Cybernetics*, The Hague, Netherlands, 2835- 2840.

Zhang, W., and Hill, R. W. (2000). "A Template-Based and Pattern-Driven Approach to Situation Awareness and Assessment in Virtual Humans." *Fourth International Conference on Autonomous Agents*, Barcelona, Spain, 116 - 123.

BIOGRAPHICAL SKETCH

Mr. Touchton holds a BSE (Nuclear Engineering) from the University of Florida (1974), an MS (nuclear science and engineering) from Carnegie-Mellon University (1977), and an MS (computer science) from the University of North Florida (1995). He is a licensed Professional Engineer and has authored and co-authored more than 40 publications. He is currently serving as a Graduate Research Assistant in the Center for Intelligent Machines And Robotics (CIMAR) at the Mechanical and Aerospace Engineering Department of the University of Florida. Mr. Touchton was an integral part of Team CIMAR, whose autonomous ground vehicle, NAVIGATOR, placed 8th at the inaugural DARPA Grand Challenge Event in March 2004 and 18th at their second Event in October 2005. Mr. Touchton is an active participant in the Joint Architecture for Unmanned Systems (JAUS) Working Group, a DoD-sponsored body aimed at ensuring the interoperability of robotic systems and components. Upon graduation, Mr. Touchton will be joining Honeywell Aerospace as a Managing Director in its Business Innovation Center.

Mr. Touchton has 30 years of engineering and consulting experience encompassing product development, project management, and marketing in the areas of advanced software development, system analysis and design, liaison engineering, problem-solving, risk/reliability analysis, and emergency planning. Past efforts include development of expert and knowledge-based systems, including real-time monitoring and control, dynamic scheduling, and numerous business and manufacturing applications, and participation in risk and reliability studies for nuclear power plants, fossil power plants, and the defense industry.

In 1984, Mr. Touchton co-founded PATHTECH Software Solutions, a provider of e-Commerce solutions, advanced web applications, custom software development, and IT consulting, and served in a management capacity as a "working" President, CEO, and Chief

Technology Officer. PATHTECH was acquired by Cary, NC-based Strategic Technologies in February of 2000, where Mr. Touchton continued as Chief Technology Officer and then as Vice President of Advanced Technologies. His responsibilities included business development, adoption of software engineering best practices, overseeing Strategic Technologies' entrée into emerging software technologies and the creation of an offshore programming strategy, as well as management of selected consulting and development projects.

Mr. Touchton has received numerous honors, including UNF Distinguished Alumni Achievement Award 2001, UNF Outstanding Alumnus (Computing Sciences and Engineering) 1997, and Florida's Jim Moran Entrepreneurial Excellence Award 1997.