12-2005

A Comparison of the Sensor Brick Concept as a Modular System Architecture to the Realtime Control System as the Operational Architecture

James Robert Wilson
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by James Robert Wilson entitled "A Comparison of the Sensor Brick Concept as a Modular System Architecture to the Realtime Control System as the Operational Architecture." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Mongi Abidi, Major Professor

We have read this thesis and recommend its acceptance:

David Page, Seong Kong

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by James Robert Wilson entitled "A Comparison of the Sensor Brick Concept as a Modular System Architecture to the Realtime Control System as the Operational Architecture." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Mongi Abidi
Major Professor

We have read this thesis
and recommend its acceptance:

David Page

Seong Kong

Accepted for the Council:

Anne Mayhew
Vice Chancellor and
Dean of Graduate Studies

(Original signatures on file with official student records)

A Comparison of the Sensor Brick Concept as a Modular System Architecture to the Realtime Control System as the Operational Architecture

A Thesis
Presented For The
Master of Science Degree

The University Of Tennessee, Knoxville

James Robert Wilson

December 2005

ACKNOWLEGMENTS

The completion of my master's work has required sacrifice and toil on my behalf that otherwise would not have been possible without the support and assistance of many people including, but not strictly limited to, those mentioned below.

The unceasing professionalism and commitment displayed by my fellow graduate research assistants in the IRIS Laboratory has been truly outstanding. Their character, integrity, and intellect contribute to an environment rife with the exhilaration that can only be a result of the passion we all have for our work. Particularly, I am most indebted to my very fine colleagues at IRIS-West that directly provided me support. Their synergy of teamwork and skills resulted in polished and professionally presented demonstration events on every occasion. Namely, I would like to express my appreciation to Mr. Nikhil Naik; Mr. Chang Cheng; Mr. Chung-Hao Chen; Ms. Roselyne Barreto; and (IRIS staff member) Mr. Doug Warren.

Faculty members who graciously agreed to serve on my defense committee included Dr. Seong-Gon Kong, and Dr. David Lon Page. Dr. Kong's objectivity and insight resulted in a stronger, better thesis. It was also my privilege to have taken several courses led by Dr. Kong, whose academic instruction was always well prepared, robust and insightful. Dr. David Lon Page, in addition to serving on my committee, acted as my immediate faculty advisor. Dr. Page has been an unwavering and constant advocate that not only mentored my research, but shared many outstanding and well-rounded world views in the many discussions that we had over the course of the nearly two years working together. To both gentlemen, I would like to express my appreciation for your support, advice and hard work that helped me in the completion of my academic endeavors.

I would like to express my most sincere appreciation to the IRIS Laboratory director, Dr. Mongi Abidi. The experience that I gained through the association with the IRIS Laboratory has irrevocably changed my life and my perception of it. I was honored that Dr. Abidi asked me to be the principle student architect to formally express his vision on modularly designed, sensor-based intelligent agents. The infusion of this knowledge will serve as a springboard for the rest of my professional life. For this, and all the opportunities that were afforded to me through the experience at the IRIS Laboratory, I am profoundly grateful to you, Dr. Abidi.

Finally, I would like to thank the most special people in my life, my family: my wonderful fiancée, Jillian; my devoted sisters, Sue, Linda, and Peggy; my loving mother, Margaret; and my late father, Jim. My success has been due to your constant love and belief in me.

ABSTRACT

This thesis describes the application of modular design into an intelligent mobile robotic system. The Sensor Brick System, a mobile robotic system infused with artificial intelligence, is derived through the concept and application of modular design. Modular design drives the system's characterization by defining, identifying and selecting the essential composite of sub-systems that form the system architecture. The principles of modular design designate grouping the technologies into modules that facilitate maintenance of the system components and allow the rapid integration of future technological developments. Developing the operational control of the system requires an examination and comprehension of the theories of intelligent control. The operational architecture of the system, an application derived from the theories of intelligent control, describes the tasks, operational units, and information flows of the system in a scope of high resolution. Generally, the application of a control structure directs the evolution of and for the intelligent systems by providing a framework for the system as it achieves increasing degrees of autonomous operations. The conceptual implementation of an operational control structure in the form of the Realtime Control System into the Sensor Brick System describes the system in a future, more developed state, and constitutes the penultimate contribution of this research. A demonstrable result of the Sensor Brick System describes the system as it is applied to an under-vehicle inspection scenario. The application of the system highlights the attributes of the system as it extracts information about the state of the vehicle using various sensor bricks.

CONTENTS

FIGURES

1 INTRODUCTION

1.1 Motivation

Increasing the range of applications in which mobile robotic sensor units are of use and their effectiveness is the motivation for the development of the *Sensor Brick System* (SBS). The Sensor Brick *system architecture* is based on the philosophy of *modular design*, which emphasizes operational independence, an interoperable control and payload (sensor) capability, and a standardized approach in the development of the system. Implementing a highly modular approach to the design and construction of a system provides a method that overcomes the deficiencies of uniquely built systems such as many of the commercial systems presently in production. These commercial systems use proprietary software and hardware that can not be interchanged, and have no interoperable control capability. Interoperability and interchangeability are key concepts that enable a system (or system-of-systems) to be used in greater ranges of application and effectiveness. To facilitate this concept, the SBS is based on the concept of *modularity*. Modular systems are composed of smaller sub-systems that interact with each other. This methodology is advantageous in many ways. By using readily available off-the-shelf components from commercial vendors, the systems and subsystems become more reusable and reconfigurable. This allows the system to be dynamically organized, enabling it to suit a wider variety of applications than other types of systems. Maintenance and reliability are improved because there is no need for specialized components. The systems' effectiveness and reliability is also improved due to the ease of configurability and the interchangeability of components.

At present, many mobile robotic systems are being produced both in the private and government sectors. Manufacturers include Remotec, which produces ANDROS, Foster-Miller which produces TALON, Tank Automotive Command Research and Development Engineering Center (TARDEC) which produces (ODIS), as well as many others. These systems are designed by examining the expected mission parameters for the system, and then building the mobile robotic unit to suit specific operational requirements in the expected areas of application. Each system is constructed using its own proprietary control system that is not directly translatable to any other system; payloads and manipulation appendages are generally not interchangeable.

ANDROS, shown in Figure 1, is manufactured by Remotec, of Clinton, Tennessee. It is a commercially manufactured mobile robotic system used in many military and police units around the world. Articulated tracks allow ANDROS to maneuver over rough terrain and

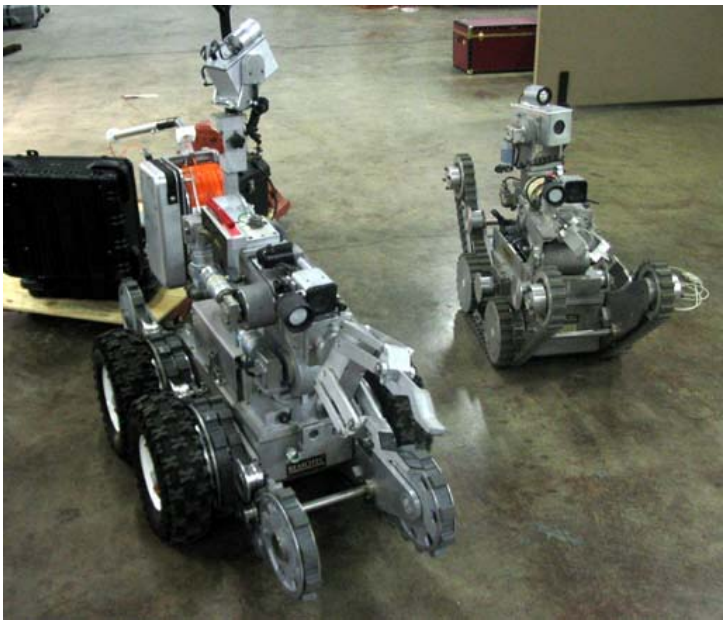

Figure 1: ANDROS, manufactured by Remotec.

obstacles, climb stairs, and cross ditches as wide as 24 inches. The vehicle is environmentally sealed to operate in any weather condition, and in areas of high temperature/humidity. Locomotion over smooth terrain is accomplished by attaching four (4) quick-release wheels with pneumatic tires onto the fixed (inner) hubs of ANDROS. Locomotion and mobility requires utilizing a customized control unit through wire cable, fiber optic cable, or radio frequency links. Auxiliary devices, such as sensors, require either a mounting kit, or an assembly that must be robot-ready modified. ANDROS is designed to be a physically robust system; it can stand the shocks produced by shotguns, disruptors and street sweepers. It is equipped with an integrated precision manipulator, color, black and white, and infrared cameras, two-way audio, and sensor support for detecting radiation and chemical threats. ANDROS can be optionally equipped with a variety of tools and accessories and can be used in explosives handling, nuclear surveillance, hazardous materials response, and SWAT operations [22].

Figure 2 shows the TALON system manufactured by Foster-Miller Incorporated, Waltham, Massachusetts. TALON is widely used for explosive ordnance disposal (EOD), reconnaissance, communications, sensing, security, defense and rescue. The TALON system has all-weather, day/night and amphibious capabilities and can navigate virtually any terrain.

Figure 2: TALON, manufactured by Foster-Miller [21].

The TALON is specifically designed so that it is human-portable, e.g., it weighs less than 100 lb (45 kg) so that it can be easily transported becoming instantly ready for operation. It is highly mobile; Foster-Miller claims that TALON has highest payload capacity and payload-to-weight ratio as well as being the fastest mobile robot on the market at present. Like ANDROS, TALON is designed to be a physically robust system. The largest TALON is used for EOD/IED missions, while smaller versions are often used for both armed and unarmed reconnaissance. For this purpose they are equipped with a variety of day/night color cameras and listening devices. TALON can be configured with a variety of weapons payloads as well as off-the-shelf chemical, gas, temperature, and radiation sensors that can be read simultaneously, remotely and in real-time via one integrated hand-held display unit developed exclusively for TALON by Foster-Miller. TALON is designed to withstand repeated decontaminations. Mobility and navigation control for TALON uses a teleoperated radio frequency link [21].

ODIS (Omni Directional Inspection System), developed by Tank-Automotive Research and Development Engineering Center, Warren, Michigan, is shown in Figure 3. ODIS is a teleoperated mobile device that sends images of a vehicle underside to the human operator.

Figure 3: ODIS (Omni Directional Inspection System) developed by Tank-Automotive Research and Development Engineering Center (TARDEC) [9].

It is a man-portable system that uses a military standard BB390 12 volt battery for the robot and the control apparatus. ODIS is a monolithic device; it has been developed strictly for under-vehicle inspection. The present incarnation of ODIS performs under-vehicle inspections to detect explosives and contraband. With auxiliary mounting hardware, it can be fitted with radiological, biological and chemical sensors. ODIS enables soldiers to perform inspections from a safe stand-off distance, rather than using mirrors on sticks. Figure 4 shows an operator with the ODIS teleoperation control system [9].

The current versions of the systems described above suffer from expensive, proprietary components that are non-interchangeable. These characteristics also contribute to the difficulties associated with interoperable control. In direct contrast to these systems, the modular SBS proposed in this thesis utilizes sensor payloads that are easily interchangeable and require no special mounting system or equipment. By utilizing modular design, a component failure does not adversely affect the operational status of the system; an entire module consisting of an off the shelf component, quickly and easily replaces the failed component in a plug-and-play capacity. Interoperability can also be more readily implemented with a modular designed system.

Figure 4: The operator control system for ODIS [9].

In summary, commercial systems have flaws that the modular SBS does not have. Those systems are built for specific purposes, have no interoperable control capability, and the (sensor) payloads are not readily interchangeable.

1.2 Sensor Brick System Applications

Identifying the wide variety of situations that can be addressed by the SBS capabilities provides ample motivation for this research. The targeted missions for these systems include: under-vehicle threat assessment, standoff checkpoint inspections, scout surveillance, intruder detection, obstacle-breach situations, and render-safe scenarios [12]. These mission types are applicable in a vast range of scenarios which include but are not limited to:

- Border Crossing Posts,
- Customs Inspection Points,
- Border Security and Integrity,
- Stadiums,
- Power Plants,
- Chemical Facilities,
- Nuclear Plants,
- Petroleum Factory,

- Parking Lots Under High Rises,
- Open or High Rise Parking Lots,
- Render-Safe Situations,
- Search, Rescue, and Recovery,
- Post-Event Inspection,
- Bridges,
- Landmarks,
- Tunnels,
- Hydroelectric Dams.

The advantages inherent to modularly designed robotic systems like the SBS become apparent as the applicability and usefulness to more and more of these scenarios are realized. As an example, consider a scenario where the operational hardware of the SBS serves as first-line security detection (sentries). For this duty, the robotic system safeguards the facility by scanning the environs of the facility in order to reconnoiter for any anomalous action or entity.

If the facility under consideration is gated, the SBS acts as preventative agents by systematically performing access control checks, or by undertaking reconnaissance of the outlying perimeter or neighborhood of the facility (e.g., an airport boundary, a border, a fence-line, etc.). By utilizing a mobility and drive system that uses semi-autonomous or autonomous navigation systems, the brick becomes a mobile intelligent agent, capable of *searching* or *learning* about its environment.

1.3 Contributions of Research

The direct contributions made by this research to the existing knowledge base are outlined in the following sections. An iconic representation of the contributions of this research is shown in Figure 5.

1.3.1 The Sensor Brick Concept

The modular approach to development of the SBS is characterized by a system that has been divided into smaller sub-systems that interact with each other. Previous work [18, 19, and 20] in the IRIS Laboratory has initially proposed the SBS concept. A major contribution of this research is the formal definition of specific architectures that define and control the SBS. A comparison of the SBS to the operational control architecture, the Realtime Control System, segments, and synchronizes the sub-systems so that they are compatible to the structure required in RCS. In the comparison, the system-of systems, the SBS, the sensor bricks, the brick's four (4) sub-systems (or 'blocks'), as well as both higher, and lower levels of the *system architecture* that define and compose any mobile system are compared to their role in RCS.

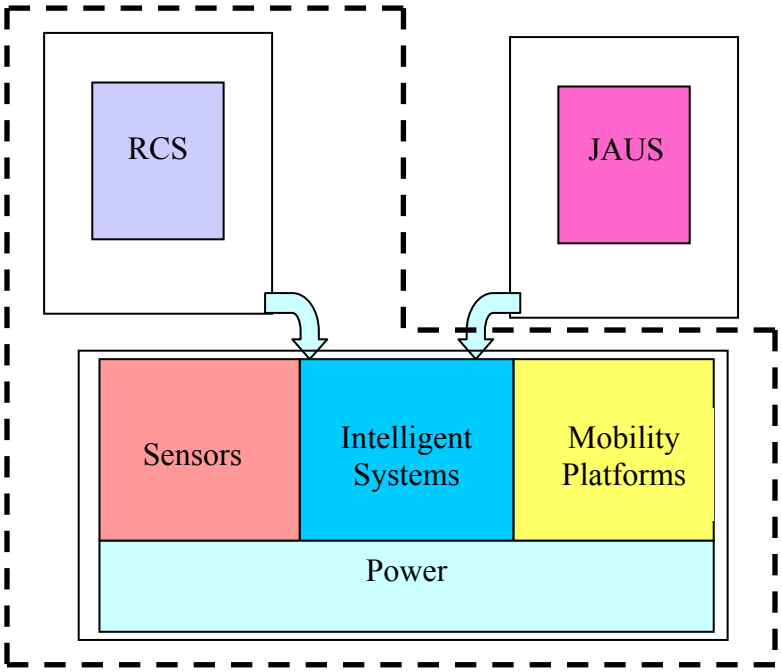

Figure 5: An iconic representation of the research contribution.

1.3.2 The Sensor Brick System as a Systems Architecture

The system is based on the concept of a *Sensor Brick*. Each modularly designed Sensor Brick utilizes four primary subsystems to acquire data, to provide locomotion, as well as the artificial intelligence and computing resources that are necessary for navigation, data processing, and communication. Additionally, each brick has an independent power supply system which consists of any batteries, converters, or other devices pertaining to the transmission and supply of power. These essential elements of the brick are encapsulated into modules of technology that are reliable, readily available, and inexpensive. The system architecture formulated into the SBS follows a modular design that provides one solution to the demands placed on it by the operational and technical architectures of the system. The development of a growing and evolving system requires the selection of an operational control structure that can be developed with the expectations of the system. This method will facilitate the evolution of the SBS into a more autonomous capability. Several criteria are important in the selection of the control structure. The most fundamental question is how well the architecture allows the robot to deal with its environment. Other important issues involve the ease of use and accessibility of the architecture including programming tools and expertise as well as the performance of the system under the selected control structure (does it act in real-time, does it get the job done, is it failure-prone?). It is important to note that no matter how the different

architectures are compared and evaluated, the comparison and evaluation is specific to the system design, the working environment(s), and the tasks the system is expected to perform. It is important to note that not all tasks, environments and designs are comparable.

1.3.3 A Comparison of the Sensor Brick System and the Realtime Control System

Examining the theory of *operational architecture* (system interaction and control) leads to the selection of a specific *reference architecture* which defines how the system will perform its tactical movements with respect to strategic goals. This interaction is accomplished by the installation of a control structure (a command and control hierarchy) that is based on modeling the expected activity, defining operational elements, decomposing tasks, and structuring the information flow.

Chapter 2 is devoted to the overview of the theory of operational architecture and is used in conjunction with the expected applications of the SBS to derive fundamental attributes of the system. The Realtime Control System (RCS) is a sophisticated operational *reference architecture* developed at the National Institute of Standards and Technology (NIST) by James Albus. Reference architectures are specific formulations that are applicable to real systems, such as the SBS. The application of RCS into the SBS is intended to demonstrate the capabilities of RCS, as well as providing a roadmap for the system as it moves toward a more autonomous state of operation. Chapter 5 utilizes the conceptual application and comparison of RCS into the SBS.

1.4 Document Organization

The thesis is divided into three (3) broad sections. The first section consists of Chapters 1 and 2. Chapter 1 serves as a general introduction and describes the format for the remainder of the report. The major concepts of the *hierarchical, reactive, and hybrid architecture paradigms* concepts are discussed in Chapter 2. The survey examines the three (3) different types of control structures that have been identified for use in intelligent mobile robotic systems. The theories and fundamental aspects of each particular architectural paradigm including the elemental descriptors, the attributes and characteristics, and the advantages and disadvantages of each style are recounted. Chapter 2 concludes with an overview of the Joint Architecture for Unmanned Systems (JAUS), an important *technical architecture*.

The second section includes Chapters 3 and 4. Chapter 3 provides a thorough overview of the *system architecture* of the SBS and explains the symbiotic inter-relationship of the operational, technical and system architectures. The focus is on the advantages provided by modular design, and the Sensor Brick concept with the essential subsystems. Chapter 4 is devoted to an overview of the Realtime Control System, developed by James Albus and others at the National Institute for Standards and Technology.

The third section (Chapters 5 and 6) examines the implementation of the Realtime Control System, an application of the SBS using RCS in an under-vehicle inspection scenario. Chapter 5 compares the components of RCS to its host system, the SBS and develops a conceptual implementation into the SBS. Chapter 6 describes an application of the SBS in a real-world, under-vehicle inspection scenario. Descriptions of the actions and manifestations of the system for one inspection cycle are included that demonstrate the various stages of development of the SBS. Chapter 7 summarizes the conclusions and future directions of the research.

2 LITERATURE SURVEY

2.1 The Hierarchical Paradigm

An architectural *paradigm* is a philosophy or set of assumptions or techniques, which characterize an approach to a class of problems. *Sensing*, *planning*, and *acting* serve as the *primitives* for all three (3) major paradigms, including the *hierarchical paradigm*.

Each paradigm is classified by one of two different methods:

- According to the relationship between the three primitives – *sense, plan, act*; and,
- By the way sensory data is processed and distributed through the system.

In the hierarchical paradigm, shown schematically in Figure 6, the robotic system executes a never-changing sequence of actions. It first *senses* the world. Secondly, it constructs a *global world map*. In the last step the system, with no additional sensing (without any rearrangement of the global world map, e.g., the robots 'eyes' are closed) *plans* the directives needed to reach the goal. In last step of the sequence, the robot *acts* to carry out the first directive. After the robot has carried out the initial *sense, plan, act* sequence, it begins the cycle again. Figure 7 depicts the unidirectional sequence of the hierarchical paradigm in terms of the inputs and outputs of each respective stage. The output, then acts as input to the next stage, until after the output of the 'act' stage is reached. Then the cycle begins again. The robot reopens it 'eyes', senses the consequence of its first action, re-plans the directives (even though the directives may not have changed), and then acts (again). Operationally, this cycle repeats over and over until a terminus is reached.

The hierarchical paradigm is characterized by the following important points:

- The *flow of control* between the components is *unidirectional and linear*. Information flows from sensors to the world model, then to planning, then to acting (effectors); *never in the reverse direction*.
- The execution of *this paradigm* is analogous to execution of a *computer program*.
- The *information* is in the *composite structure*, not the primitives**.** The *intelligence of the system* lives in the *planner* or the programmer, *not the execution mechanism* [2].

2.1.1 Attributes and Fundamental Issues

The hierarchical paradigm is an all-encompassing structure: that is, all of the sensor observations are fused into one global data structure. The 'planner' accesses the global

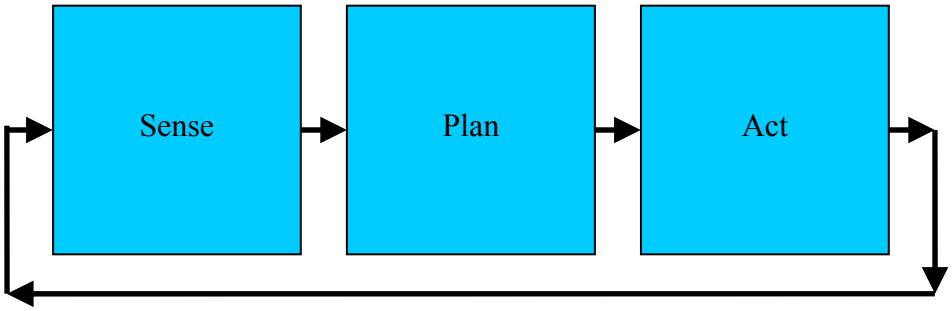

Figure 6: The Hierarchical Paradigm [2].

Primitives	Input	Output
Sense	**Sensor Data**	**Information**
Plan	**Information**	**Directives**
Act	**Directives**	**Actuator Commands**

Figure 7: Data - Information - Directive - Command stream of the Hierarchical Paradigm [2].

data structure, which is generally referred to as a *world model*. The term world model is a very general term; 'world' can mean both the outside world, and whatever meaning the robot credits to it. In the hierarchical paradigm, the world model typically contains:

1. An *a priori* representation of the environment the robot is operating in (e.g., a map of the *environment);*
2. *Sensing* information (e.g., Robot starts in a known location and based on where it has traveled, the robot must be in this particular location at the present);
3. Any *additional cognitive knowledge* that might be needed to accomplish a task (e.g., all widgets must be inspected).

Typically, the hierarchical paradigm is characterized as having a *horizontal decomposition* - the tasks are ordered and sequential, one after the other.

Two (2) issues pervade the hierarchical paradigm: the *closed world assumption*, and the *frame problem*. The closed world assumption says that the world model contains everything the robot needs to know; this implies that there can be no 'surprises'. The creation of a single representation which can store this information can be very demanding. Since it is very easy to not put the vital details into the world model, essentially the success of the mission depends on how well the human programmer can think of everything. The dependence of the global world model is related to the frame problem. Since the hierarchical paradigm is a monolithic or unchanging structure, simple tasks may not be segmented and executed as such; instead, the entire cycle must be completed before the next iteration on the task at hand may be executed. Planning is also problematic in the hierarchical paradigm. If the programmer could come up with each possible case, the resulting world model would be huge. The number of facts (or axioms) that the program would have to sort through for each pass of hierarchical execution becomes intractable for realistic applications. The problem of representing a real-world situation in a way that was computationally tractable is referred to as the *frame problem*. The opposite of the closed world assumption is known as the *open world assumption*. When a robot must function in the 'open world' it is then assumed that the closed world assumption cannot be applied to that particular domain.

Formally representing the world and maintaining changes about it is non-intuitive. This idea helps justify the use of a closed-world assumption because it becomes too difficult to modify the planning algorithm if the world model suddenly changes [2].

2.1.2 Advantages and Disadvantages

2.1.2.1 Advantages

The hierarchical paradigm's primary advantage is that it *orders* the relationship between sensing, planning, and acting. By its nature, hierarchical architectures facilitate the focusing of attention [2]. In multi-level hierarchical structures, the higher levels have broader perspective and a longer planning horizon. They can determine what is important

with respect to achieving high-level or long-term goals. This information can then get passed down to lower levels in the form of priorities, modes of behavior, and objects of interest. Lower levels use this information to focus attention on objects and tasks that are relevant to high-level goals. Planning resources at each level can be focused on selecting sub-goals that contribute to the high-level goals. Additionally, at each level attention can be used to mask, filter, and window sensor data and to focus sensor processing and world modeling resources on objects and events that are important to high-level goals. This enables the entire (intelligent) system to think and act toward a single unified set of goals. Behaviors throughout the entire hierarchy, down to the lowest level are optimized and focused in relation to achieving those goals [1].

2.1.2.2 Disadvantages

The main disadvantage of the hierarchical paradigm is *planning*. For every cycle, the robot must update the global world model and then do some type of planning. The sensing and planning algorithms, especially before 1990 were relatively slow, compounding the problem. This restriction imposed a significant bottleneck with respect to the functional progress of mobile robotic systems of this era.

Another disadvantage involves the process of *searching* for a plan to reach a goal. This can be a computationally intensive and time-consuming process. The number of hypothetically possible plans is the number of possible actions at each step in the plan raised to the power of the number of steps of the plan. Even in the case of relatively simple finite-state systems, the space of possible plans can easily become too large to search in a practical amount of time. For example, the number of possible plans for the game of chess is estimated to be on the order of 10^{120} (more than the number of microseconds since the big bang).

An additional disadvantage is that sensing and acting are always disconnected. This eliminates any stimulus-response types of actions (e.g., 'a rock is crashing down on me, I should move *anywhere*'). The dependence on a global world model is related to the frame problem. The *reference architecture* Nested Hierarchical Controller (NHC) (discussed fully in a later section) represents an attempt to mitigate this issue by dividing-up the world model into pieces best suited for a particular type of action. Unfortunately, the decompositions of the 'plan' or 'act' primitives used by NHC are dependent on a specific application.

Uncertainty is also an issue that is inadequately addressed by the hierarchical paradigm. Uncertainty comes in many forms, such as issues in semantics (how close does 'next to' actually mean?), sensor noise, and actuator errors. Another important aspect of uncertainty is action completion: did the robot actually accomplish the action [2]?

2.1.3 Nested Hierarchical Controller (NHC) – A Reference Architecture

2.1.3.1 Overview

Architectural paradigms consist of the structure of components, their relationships, and principles of design, including the assignment of functions to subsystems and the specification of the interfaces between subsystems. *Reference architectures* take the general relationships assigned by an architectural paradigm, and formulate the entire collection of functions, entities, events, relationships, and information flow involved in interactions between and within subsystems. A reference architecture is a sufficiently specific formulation that hypotheses can be constructed, tested, and either validated or disproved. Once validated, reference model architectures can form the basis for an engineering methodology that can be used to design and build an intelligent system [1].

Alex Meystel developed early versions of Nested Hierarchical Control (NHC) from *nested hierarchical algorithms* that are used in the development of models of computer vision and general signal processing [36]. The core of nested hierarchical control is based on the concept that nested hierarchical knowledge organization enables efficient practice of design and control using nested search methods in state space [2].

The earliest versions of nested control hierarchies are based on the idea of system *partitioning*. Saridis initially conceptualized the essential elements in the area of hierarchical control revealing some of the major features typical for hierarchical control systems: a controller at the top of the system controls the process as a whole; controllers at the bottom control the sub-processes and coordinate the actions taken with the decisions that are made at higher levels [37]. The controller at the top is the least precise. The controller at the bottom is the most precise [6].

The NHC architecture has components that are easily identified as sense, plan, or act, as shown in Figure 8. The robotic system begins by gathering observations from sensors, and combines those observations to form the *world model* through the sense activity. The *world model* may contain a priori knowledge about the world, such as maps of a building, or rules that mandate certain behaviors. After the *world model* has been created or updated, then the robot can plan what action it should take. Planning for navigation has a local procedure consisting of three steps executed by the *mission planner, navigator, and pilot*. Each of these modules has access to the *world model* in order to compute their portion of planning. The last step in planning is for the *pilot* module to generate specific actions for the robot to do (e.g., turn left, turn right, etc…). These actions are translated into actuator control signals by the low-level controller. Together, the low-level controller and actuators form the act portion of the architecture.

The *mission planner* then accesses a map of the building and locates where the robotic system is and where the goal is located. The *navigator* takes this information and generates a path from the current location to the goal. It generates a set of intermediate

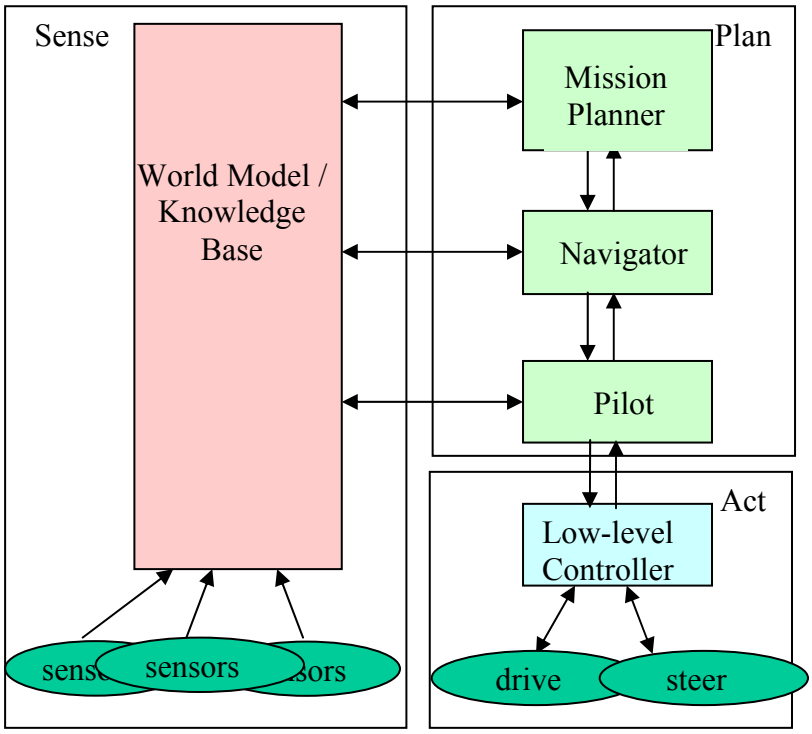

Figure 8: Nested Hierarchical Controller [2].

waypoints (straight line paths) for the robot to follow. The path is passed to the *pilot*. The *pilot* takes the first straight line segment and determines what actions the robot has to do to follow the path segment.

For instance, the robot may need to turn around to face the waypoint before it can start driving forward. In the event that a long path segment is given by the *pilot*, the robot is not necessarily walking around with its eyes closed, unlike a strictly hierarchical architecture. After the *pilot* gives the low-level commands and the controller sends actuator signals, the robot polls its sensors again, updating the *world model*. However, the entire planning cycle does not repeat. Since the robot has a plan, it doesn't need to rerun the *mission planner* or the *navigator*. Instead, the pilot checks the *world model* to see if the system has drifted off the path sub-segment (in which case it generates a new control signal), if it has reached the waypoint, or if an obstacle has appeared. If it has reached its waypoint, the *pilot* informs the *navigator*. If the waypoint is the goal, then the *navigator* informs the *mission planner* that the robot has reached the goal. The *mission planner* may then issue a new goal (e.g. return to the starting place). If an obstacle is encountered to its path, the *pilot* passes control back to the *navigator*. The *navigator* must compute a new

path and sub-segments, based on the updated *world model*. Then it gives the updated path sub-segments to the *pilot* to carry out (Figure 9) [2].

2.1.3.2 Advantages and Disadvantages

The NHC has several advantages. It differs from a pure hierarchical controller in that it interleaves planning and acting. The robot comes up with a plan starts executing it then changes it if the world is different than it expected. The decomposition is inherently hierarchical in intelligence and scope: the *mission planner* is 'smarter' than the *navigator*, who is smarter than the *pilot*. The *mission planner* is responsible for a higher level of abstraction then the *navigator*, etc. Other architectures in both the hierarchical and hybrid paradigms make use of the NHC organization. A disadvantage of the NHC decomposition of the planning function is that it is appropriate only for navigation tasks. The division of responsibilities does not relate as well to tasks such as picking up a box (rather than just moving over to it).

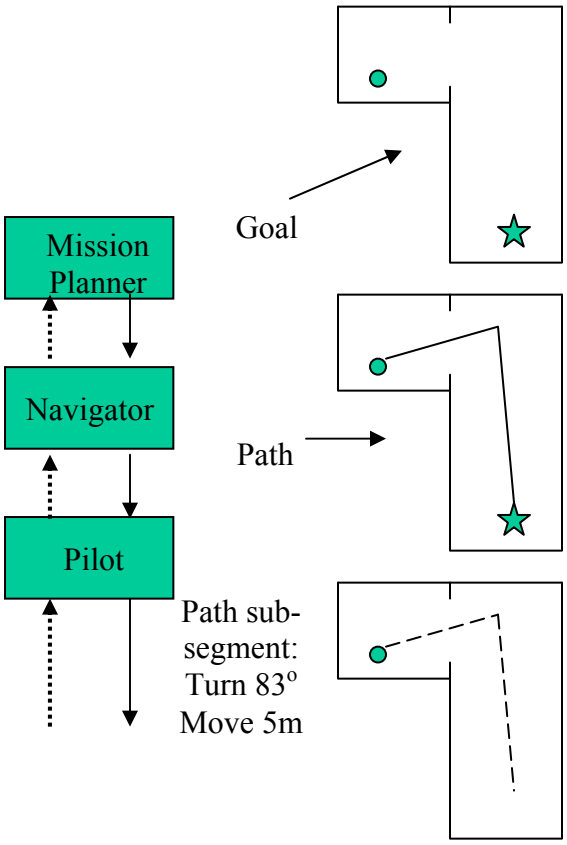

Figure 9: Planning components in the NHC architecture [2].

Also, the role of a *pilot* in controlling end-effectors is not clearly delineated. At the time of its initial development, NHC was never implemented and tested on a real mobile robot; hardware costs during this period forced most roboticists to work within simulations [2].

2.1.3.3 NHC Encapsulation

The theory of NHC resulted from the development of theories of multi-resolutional image representation, and multi-resolutional signal representation into the domain of control theory. These theories, as incorporated in NHC, solve numerous problems of control in systems with incomplete and/or inadequate information, in large systems, and in autonomously controlled systems, particularly in the area of intelligent machines. The general structure of NHC is shown in Figure 10. NHC-theory treats design and control processes as a design-control continuum. Thus, planning is a stage within this continuum. This in effect becomes 'off-line finding' the open-loop control sequence. The open-loop and closed-loop control structure is considered to be a module in the multi-resolutional hierarchy of the NHC-controller. Algorithms of multi-resolutional consecutive refinement have been developed that allow for time-efficient computation of control sequences at each level. Its use, when compared to dynamic programming algorithms is advantageous [6].

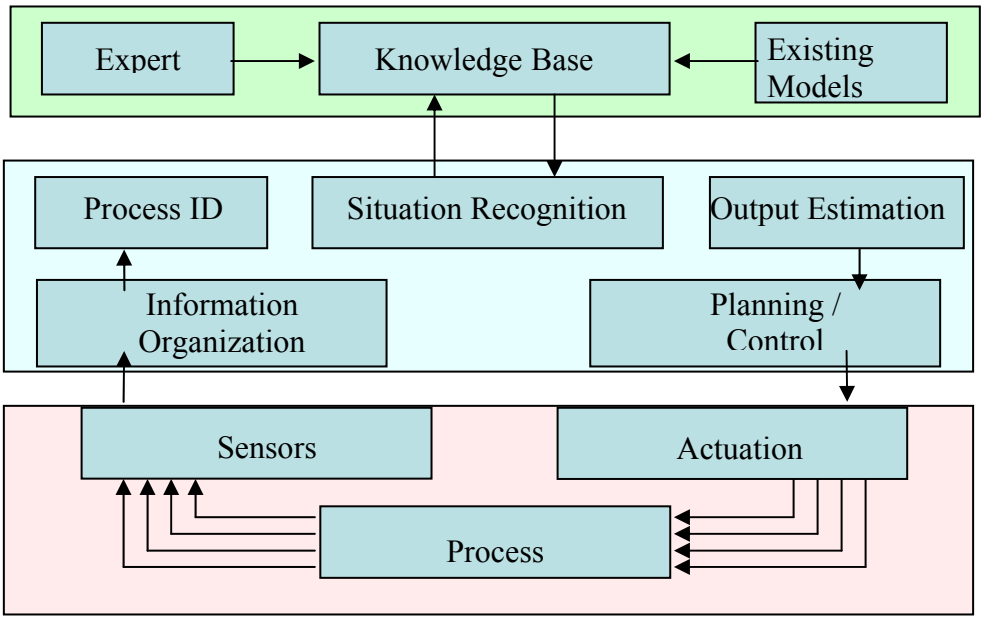

Figure 10: General Structure of the NHC-Controller [6].

2.1.4 Summary

True hierarchical systems have largely fallen out of favor except for the NIST Realtime Control Architecture (RCS), which is technically a hierarchical-based *hybrid* architecture. The decline in popularity is due in part to the hierarchical paradigms focus on particular applications. Being well-suited for a particular type of application can have the effect of reducing true modularity and portability. One often overlooked property of most hierarchical architectures is that they tend to support the evolution of intelligence from semi-autonomous control to fully autonomous control [2]. In the future, the complexity of a system model coupled with demanding closed-loop system performance requirements will call for the use of more complex and sophisticated controllers. Additionally, with increasing amounts of uncertainty (corresponding to decreases in how well the problem is structured or how the control problem is formulated) and the need for human intervention in the control process, the use of sophisticated control methodologies will be necessary. Controller complexity and sophistication is driven by both the complexities of the system to be controlled as well as the control design requirements. These ideas suggest a *hierarchical* ranking of increasing controller sophistication on the path to intelligent autonomous controls [4].

2.2 The Reactive Paradigm

The reactive paradigm first emerged in the late 1980's. It grew out of dissatisfaction with the hierarchical paradigm and with the arrival of ideas from ethology (the study of animal behavior). Reactive systems may or may not strictly adhere to principles of biological intelligence; however, they generally mimic some aspect of biology.

The reactive paradigm is important to study for at least two reasons:

- Robotic systems in limited task domains are still being constructed using reactive architectures.
- The reactive paradigm forms part of the basis for the hybrid reactive-deliberative paradigm.

This horizontal decomposition of the hierarchical structure is in direct opposition to the formulation proposed from the study of ethology.

Evidence taken from this field suggests that intelligence is layered in a *vertical decomposition*. Under a vertical decomposition, an agent (robotic system) starts with primitive survival behaviors and *evolves* new layers of behaviors which either reuse the lower, older behaviors, inhibit older behaviors, or create parallel tracks of more advanced behaviors. These parallel tracks can be thought of as *layers* of behaviors that occur simultaneously and independently. A depiction of vertical layering, indicative of the reactive paradigm, is shown in Figure 11. Each layer has access to sensors and actuators that are free from interference from any other layer. Lower level behaviors continue to operate if anything happens to an advanced behavior.

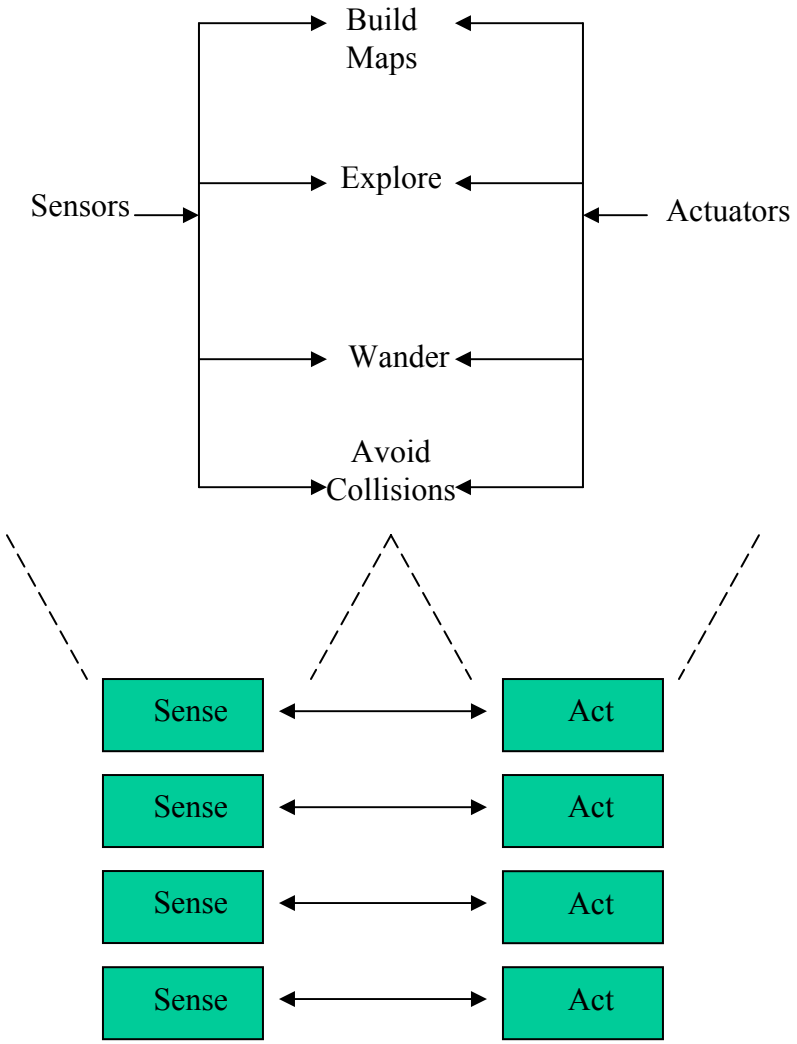

Figure 11: Vertical decomposition of task into a Sense-Act organization, associated with the reactive paradigm [2].

A return to a lower level imitates the degradation of autonomous functions in the brain; functions in the brain stem (such as breathing) continue independently of higher order functions (such as counting, face recognition, or task planning). This, for instance, is what allows a person who has traumatic brain damage to continue to breathe [2]. Early work in this area focused on defining behaviors, and on mechanisms for correctly handling situations when multiple behaviors are active simultaneously. The reactive paradigm initially met with skepticism and resistance from traditional customers of robotics, particularly the military and nuclear regulatory agencies. The end-users were uncomfortable with the reactive paradigm for two reasons. The first is the imprecise way in which discrete behaviors combine to form a rich *emergent* behavior. Secondly, reactive behaviors are not amenable to mathematical proofs showing they are sufficient and correct for a task. Ultimately however, the rapid execution times associated with reflexive behavior led to its adoption. Figure 12 depicts multiple, independent behaviors that may occur simultaneously.

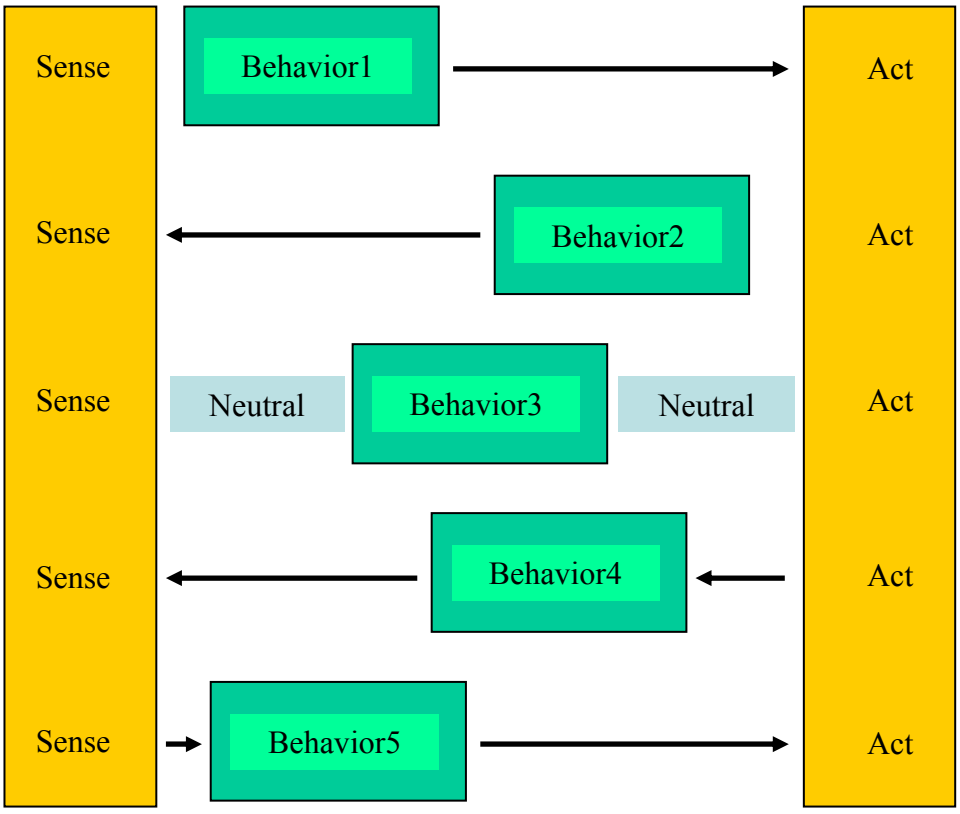

Figure 12: Sense-Act organization of the reactive paradigm into multiple, concurrent behaviors [2].

Brooks built insect-like robots with behaviors captured in hardware circuitry [38]. The control of the behaviors utilized a reactive reference architecture conceived by Brooks called *Subsumption* [2]. Subsumption has had a profound influence on many of the architectures developed by academic and NASA researchers for robotic systems [1]. Although reactive systems are composed of behaviors, the meaning of a behavior may be slightly different with respect to particular styles or applications of reactive architecture.

Purely reactive architectures do not use an internal representation of the world model. In the late 1980's, it was believed that if a programmer knew enough about an environment, a set of stimulus-response pairs sufficient to cover every possibility could be produced. This restriction provides some of the controversy over the use of internal representation. "One reason for not choosing a behaviorist architecture is the lack of a rich internal model of the world. Behaviorist architectures are often not goal-directed and typically do not generate plans for future behavior" [1]. Many researchers believe that a robotic system cannot assign meaning to its actions or environment without representing them, even if indirectly. Others believe that reliance on internal representation prevents a robot's ability to act quickly across domains [2]. Thus, reactive systems are best characterized by a direct connection between sensors and effectors. Control is not mediated by a model; it occurs as a low level pairing between stimulus and response.

Strategies which require that action be mediated by some symbolic representation of the environment are called *deliberative*. In contrast, reactive strategies do not exhibit a dedicated reliance on internal models. Instead, they displace some of the role of representation onto the environment itself. Instead of responding to entities within a model, the robot can respond directly to perception of the real-world. It may make sense to use a deliberative approach if a task is highly structured and predictable. For example, if an intelligent agent is embedded in an entirely virtual environment, it is often possible to encode aspects of the environment with some semantic representation.

However, In complex, real-world domains where uncertainty cannot be effectively modeled, robotic systems must have a means of reacting to an infinite number of possibilities. For more complicated domains it is necessary to find an appropriate balance between reactive and deliberative control.

Systems that completely avoid internal representation are ill-equipped for the many tasks that require memory or communication. On the other hand, systems that must transmit perception and action through an internal model will be perplexed in a new environment. The key to successful implementation is that the model should not drive development. Control should be built from the bottom up and distributed across the system. For a reactive design methodology to work, it is necessary that behavior be decomposed into the smallest possible components. Often, design will include a developmental phase during which these components can be honed, and then joined together. This is

accomplished by first designing, building and testing a minimal system, and then using an ongoing loop to evaluate performance and add new competence [2].

2.2.1 Attributes and Fundamental Issues

2.2.1.1 Impact of Biology

Behavior-based robotics use biology as the model for understanding intelligence. Most behaviorist-based roboticists do not model biological organisms directly. Instead, they look to nature for insight and direction. Some researchers have adopted the idea that high-level cognition is an impractical, debilitating goal, and have begun to model the lower animal world. Biological models seem to offer the best hope for creating adaptive behavior.

The science of biology directly contributes to the field of mobile intelligent systems by providing the actual robot hardware and sensors. A household fly navigates using a compound eye comprised of 3,000 facets which operate in parallel to monitor visual motion. This idea has been used to build an artificial robot eye with 100 facets that can provide a 360-degree panoramic view. Other research artifacts include artificial bees that can simulate the dance patterns and sounds of real bees well enough to communicate with other, real bees. Robot 'ants' can be built that are capable of leaving and detecting pheromone trails [2].

The influx of biological based ideas and frameworks contends that if these approaches work well-enough, then by the same evidence, this supports the behavior-based approach with respect to mobile intelligent robotics (if most animals do not rely on cognition to act, why should robotics). One opinion is that roboticists' preoccupation with high-level semantic thought merely reflects the anthropomorphic bias of human designers. One example that supports this contention resides in an experiment using the nervous system of a frog. If the connection between a frog's spine and brain is severed, it can be shown that even if the centralized control is removed, physical actions can be produced reactively and without 'thought' by stimulating particular points along the spinal cord. In this experiment, the behavioral architecture of a low-level animal's nervous system supports the hypothesis that says the behavior of a frog is encoded directly into the spine (at least to some degree). Various locations along the spine react with a different, essential motion. The stimulation in one location prompts the frog to wipe its head. Stimulating another location causes the frog to jump. Stimulating the spine in two points simultaneously makes it is possible to combine behaviors resulting in the production of a more complex form of behavior. This finding supports a fundamental premise of the behavior-based approach: *sophisticated, high-level behavior can emerge from layered combinations of simple stimulus-response mappings*. Instead of careful planning based on modeling, high-level behavior such as flocking or foraging can be built by blending low-level behaviors such as dispersion, aggregation, homing and wandering. Strategies can be built directly from behaviors, whereas plans must be based on an accurate model [2].

2.2.1.2 Fundamental Aspects

In the reactive paradigm actions are accomplished though behaviors. As in ethological systems, behaviors are a direct mapping of sensor inputs to a pattern of motor actions that are then used to achieve a task. From a mathematical perspective, behaviors are simply a transfer function, transforming sensor inputs into actuator commands. In this sense, the reactive paradigm does not use the 'plan' primitive. The sense and act components are *tightly coupled* into behaviors, and all robotic activities emerge as the result of these behaviors operating either in sequence or concurrently. However, the 'sense-act' organization does not specify how the behaviors are coordinated and controlled.

The reactive paradigm restricts sensing into either a local behavior, or behavior-specific modules (Figure 13). Every behavior has its own dedicated sensing, but in many cases it is implemented as one sensor and perceptual schema per behavior. In other cases, more than one behavior can take the same output from a sensor and process it differently (via the behavior's perceptual schema). Behaviors are independent of each other; one behavior does not know what another behavior is doing or realizing.

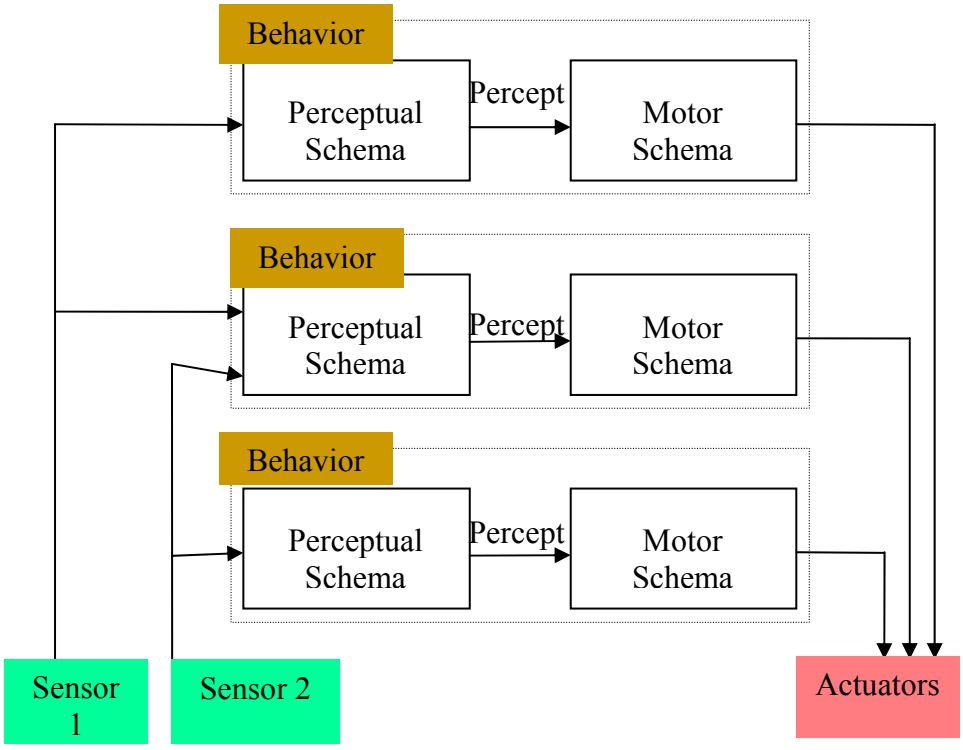

Figure 13: Behavior-specific sensing organization in the reactive paradigm: sensing is local, sensors can be shared, and sensors can be fused locally by a behavior [2].

Early implementations of the reactive paradigm utilized the idea of 'one sensor, one behavior,' but to achieve more advanced behaviors, fusing the output of multiple sensors within one perceptual schema increased the precision or resulted in a better measure of the strength of the stimulus. As long as the fusion is local to the behavior, this type of sensor fusion is permitted [2].

2.2.1.3 Attributes

A reactive robotic system can be decomposed by mapping functionality into behavior(s). Directly, each behavior affects the tight coupling of perception to action. Indirectly, this implies that there is no need to use an intervening abstraction of a global representation.

The reactive paradigm can be described by two major characteristics. The first is that the reactive architecture produces robotic systems that execute rapidly enough to permit real-time (autonomous) operation. Behaviors may be implemented directly in hardware (tightly coupled sensor-actuator loop), or through low computation complexity algorithms. Secondly, purely reactive systems have no memory. This limits reactive behaviors to pure stimulus-response reflexes.

Although many behaviors exhibit a fixed-action pattern type of response (the behavior persists for a short period of time without the direct presence of the stimulus), behaviors are controlled by what is happening in the world. This parallels the innate releasing mechanisms in primitive creatures. Whereas systems with memory execute this by storing the data necessary and remembering what the robot did last.

Additionally, architectures that follow the reactive paradigm can be characterized by most of the following five attributes:

1. *Robots are situated agents operating in an ecological niche*. A situated agent means that the robotic system is an integral part of the world. The system has its own goals and intentions. When it acts, it changes the world, and receives immediate feedback about the world through sensing. What the robot senses affects its goals and how it attempts to meet them, generating a new cycle of actions. Likewise, the goals of the system, the environment it operates in, and how it perceives the world, form the ecological niche of the system.
2. *Behaviors serve as the basic building blocks for robotic actions, and the overall behavior of the robot is emergent*. Behaviors are independent computational entities and operate concurrently. The overall behavior is emergent; there is no explicit 'controller' module which determines what will be done, or function which may call other functions. There may be a coordinated control program in the schema of a behavior, but there is no external control of all behaviors for a task. As with animals, the 'intelligence' of the robot is in the eye of the beholder, rather than in a specific section of code. Since the overall behavior of a reactive robot emerges from the way its individual behaviors interact, the major differences between reactive

architectures is usually the specific mechanism for interaction. These mechanisms include combination, suppression, and cancellation.

3. *Only local, behavior-specific sensing is permitted*. The use of explicit abstract representational knowledge in perceptual processing, even though it is behavior-specific, is avoided. Any sensing which does require representation is expressed in ego-centric (robot-centric) coordinates. Consider obstacle avoidance. An ego-centric representation means that it does not matter that an obstacle is in the world at coordinates (x,y,z); the only concern is its relative location to the robot. Sensor data, with the exception of GPS, is inherently ego-centric (e.g., a range finder returns a distance to the nearest object from the transducer), so this eliminates unnecessary processing to create a world model that extracts the position of obstacles relative to the robot.
4. *These systems inherently follow good software design principles*. The modularity of these behaviors supports the decomposition of a task into component behaviors. The behaviors are tested independently, and behaviors may be assembled from primitive behaviors.
5. *Animal models of behavior are often cited as a basis for these systems or a particular behavior*. Unlike in the early days of mobile robotics where there was a conscious effort to not mimic biological intelligence, it is very acceptable under the reactive paradigm to use animals as a motivation for a collection of behaviors [2].

2.2.2 Advantages and Disadvantages

Using the reactive paradigm to create a robotic system is often referred to as programming by behavior since the fundamental component of any implementation is a behavior. Programming by behavior has a number of advantages, most of them consistent with good software engineering principles. Behaviors are inherently modular and easy to test in isolation from the system (i.e., they support unit testing). Behaviors also support incremental expansion for the capabilities of a robot. A robot becomes more intelligent by having more behaviors. The behavioral decomposition results in an implementation that works in real-time and is usually computationally inexpensive, although duplicating specialized detectors (like optic flow) can be slow. If the behaviors are implemented poorly in the robotic system, then a reactive implementation can be slow. Generally, the reaction speeds of behaviors in reactive robotic systems are the same as stimulus-response times in animals [2].

The principle weakness for not choosing behaviorist architecture is the lack of a rich internal model of the world. Information from different sensors is nowhere fused into a single best estimate of the state of the world. Instead, sensors are typically connected directly to action. Each sensor-actuator system generates its own estimate of what behavior is most appropriate, and arbitration heuristics are used to select a single ‘best’ behavior for execution. Behaviorist architectures are often not goal-directed and typically do not generate plans for future behavior [1].

2.2.3 Subsumption – A Reactive Reference Architecture

2.2.3.1 Overview

Although reactive control can be useful, how is reactive control actually accomplished? Early researchers focused on planning modules. Because many of the robotic systems operated in a virtual world, the part of the architecture that controlled the motors and sensors was de-emphasized. These architectures constrained development by forcing a distributed approach where behaviors function in parallel rather than in a step-wise, linear fashion.

The subsumption architecture, originally developed by Rodney Brooks in 1986, provided a method for structuring reactive systems from the 'bottom-up' using layered sets of rules [39]. Bottom-layer behaviors such as 'avoid-collision' should be the most basic (at the lowest or most precise level) and should have the highest priority. Top-layer behaviors such as 'go to goal' encapsulate high-level intentions and may either be built from lower behaviors or function only when lower behaviors such as 'avoid collision' are satisfied. When behaviors react minimally, complexity is reduced. The idea is that each should function simultaneously but asynchronously with no dependence on the others. This independence should reduce interference between behaviors and prevent over-complexity.

Successes using subsumption architectures include six-legged walking robots, vacuuming agents and robots that collect cans. The layered approach promotes the fault tolerance and robustness necessary for such agents. For instance, although a robot designer cannot accurately predict component failure, well-designed subsumption architecture will allow behaviors to sequence and re-sequence according to unforeseen problems. Subsumption architectures do not require an explicit plan of action.

One of the biggest challenges in designing a subsumption architecture is giving the system the ability to automatically select its behaviors (arbitration). While the ability to assign priorities to behaviors already affords some organization, it is necessary to have a program that can smoothly transition between states. For instance, a robot that has finished vacuuming a room must end some behaviors and begin some others before it can exit and dump its trash. The situated automata strategy produced a reactive system where finite state automata are used to identify and react to discrete events. Other possibilities include a winner-take-all approach where spreading activation converges to a specific behavior or a voting architecture such as DAMN, where each behavior has some voice in deciding a single output [2].

2.2.3.2 Advantages and Disadvantages

Four interesting aspects of Subsumption in terms of releasing and control are:

1. Modules are grouped into layers of competence. The layers reflect a hierarchy of intelligence, or competence. Lower layers encapsulate basic survival functions such as avoiding collisions, while higher levels create more goal-directed actions such as mapping. Each of the layers can be viewed as an abstract behavior for a particular task.
2. Modules in a higher layer can override, or *subsume* the output from behaviors in the next lower layer. The behavioral layers operate concurrently and independently, so there is a need for a mechanism to handle potential conflicts. The solution in subsumption is a type of winner-take-all, where the winner is always the higher layer.
3. The use of internal state is avoided. An internal state in this case means any type of local, persistent representation which represents the state of the world, or a model. Because the robot is a situated agent, most of its information should come directly from the world. If the robot depends on an internal representation, what it believes may begin to dangerously diverge from reality. Some internal state is needed for releasing behaviors like being scared or hungry, but good behavioral designs minimize this.
4. A task is accomplished by activating the appropriate layer, which then activates the lower layers below it. However, in practice, Subsumption-style systems are not easily taskable; that is, they can't be ordered to do another task without being reprogrammed [2].

Subsumption has had a profound influence on many of the architectures developed by academic and NASA research for robotic systems. Behaviorist architectures (like subsumption) emphasize a direct path from sensing to acting as well as deliberately minimizing or bypassing the issue of internal representations of the external world. In this sense, Brooks considers the world to be its own model that can be sensed when necessary to enable behavioral decision-making [39]. Behaviorist architectures are primarily reactive and typically do not include mechanisms for planning or problem solving that anticipate and avoid future difficulties and optimize future results They emphasize stimulus-response mechanisms and largely ignore concepts such as goals, plans, and symbolic reasoning. The principal contribution of behaviorist architectures is that they have demonstrated how complex behaviors can be generated by simple reactive systems operating in a complex environment. They stand in sharp contrast to older artificial intelligence architectures like SOAR that apply complex reasoning mechanisms to relatively simple environments. Bekey has applied a version of Subsumption to controllers for robot hands and helicopters. Maes and Mataric have applied it to a variety of mobile lab vehicles. Payton has applied it to undersea vehicles [1].

2.2.3.3 An Example

In this example, the bottom-most layer is represented by a robotic system that is capable moving forward while not colliding with anything (Level 0). This is depicted in Figure 14. The system has multiple range finders each providing ranges in different directions. There are two actuators, one for driving and one for turning. The *range module* reads the

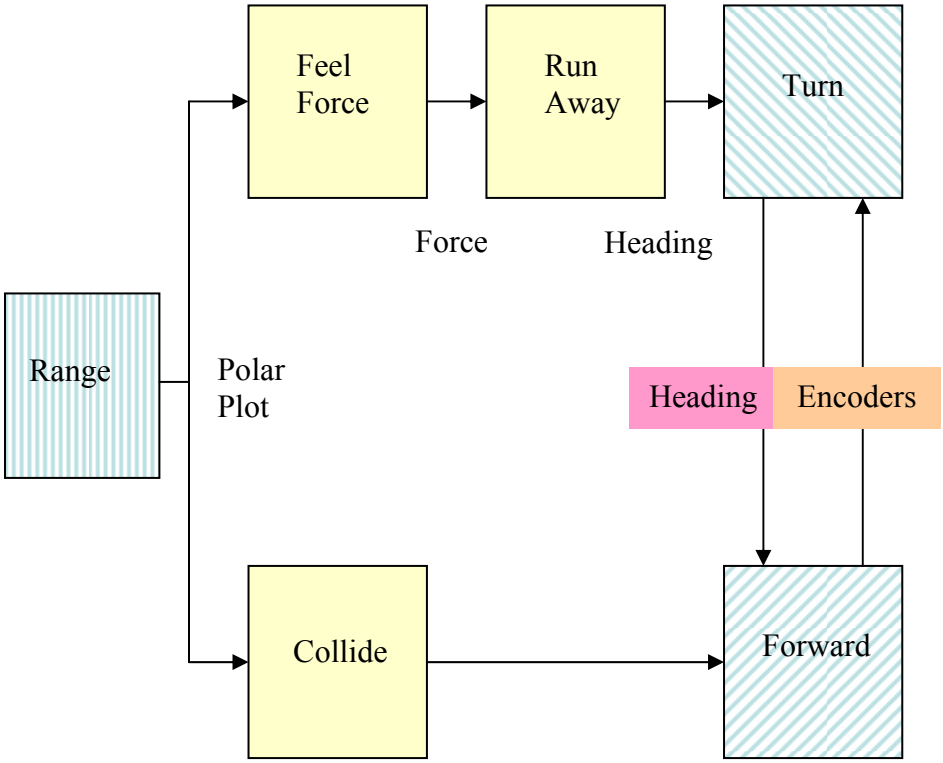

Figure 14: The modules at Level 0 in the Subsumption architecture [2].

range data and produces a polar plot representing the range readings in polar coordinates (r, Θ).

If the range reading (Figure 15) for the 'forward' direction is less than some minimum threshold, the *collision module* declares that a collision is imminent and sends a 'stop' signal to halt the forward drive actuator if it was moving. Simultaneously, the *feelforce module* receives the same polar plot. By treating each force as a *repulsive vector*, and summing the range vectors results in a new resultant vector. The *turn module* takes the direction part of the resultant vector and passes it to the steering actuators. It sends the magnitude of the resultant vector to the *forward module* so that it determines the magnitude of the next forward motion. This robotic system will sit and wait until it senses an obstacle within a predetermined threshold. This in effect, allows a human to 'herd' the robotic system. The obstacle may be motionless or moving; the response is re-computed at each sensor update. If part of the obstacle or another obstacle lies straight ahead (similar to herding the robot into a wall), it will stop and apply the results of the *run-away module* by stopping, turning and then moving forward again. Stopping prevents the system from brushing against the obstacle as it turns or moves forward.

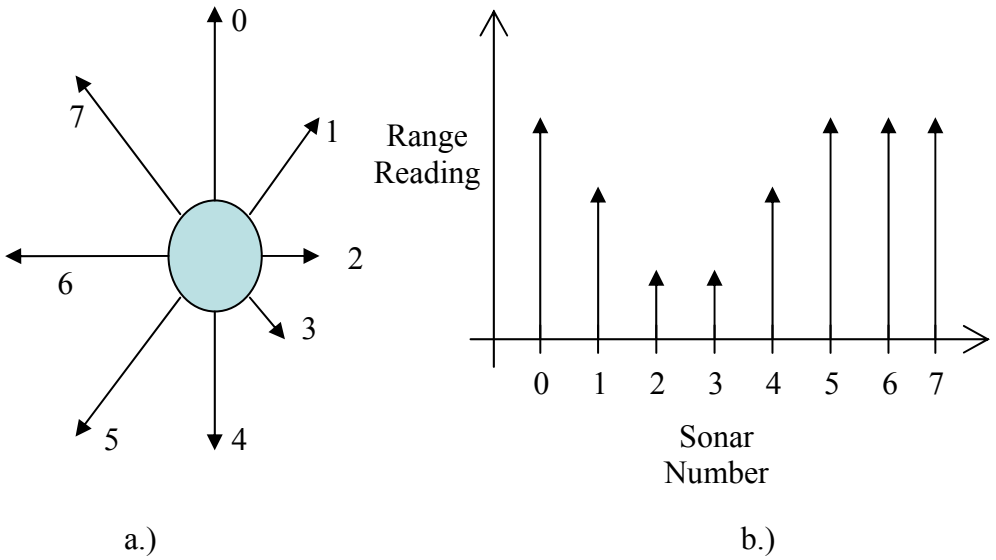

Figure 15: Eight range readings from a.) The robotic system's point of view and, b.) Plot of range reading vs. sensor number [2].

This example demonstrates a situation where complex actions emerge from a relatively simple set of modules. As seen in Figure 13, sensor data flows through the concurrent behaviors to the actuators, and the independent behaviors cause the robot to take the correct action. The *range module* is considered a global interface to the sensors, while the *turn* and *forward modules* are considered a part of the actuators. For this example, a behavior is defined as consisting of a perceptual schema and a motor schema. Perceptual schemas are connected to sensors, while motor schemas are connected to actuators.

At Level 0, the simplest possible state, the perceptual schemas are contained in the *feelforce* and *collide* modules. A depiction of Level 0 recast into primitive behaviors is shown in Figure 16. The motor schemas are in the *runaway* and *collide* modules. The *collide* module combines both perceptual processing and the pattern of action. These primitive behaviors create a rich obstacle avoidance behavior, or a *layer of competence*.

Now consider a similar robotic system that *wanders* (see Figure 17). Under Subsumption, a second layer of competence is added. Level 1, the next level of complexity consists of a *wander* module which computes a random heading every n seconds. A decomposition of Level 1 into primitive behaviors is shown in Figure 18. The random heading is a vector that can be passed to the turn and forward modules, but can not be directly passed to the *turn* module. If the random heading were to be passed directly to the *turn* module, it

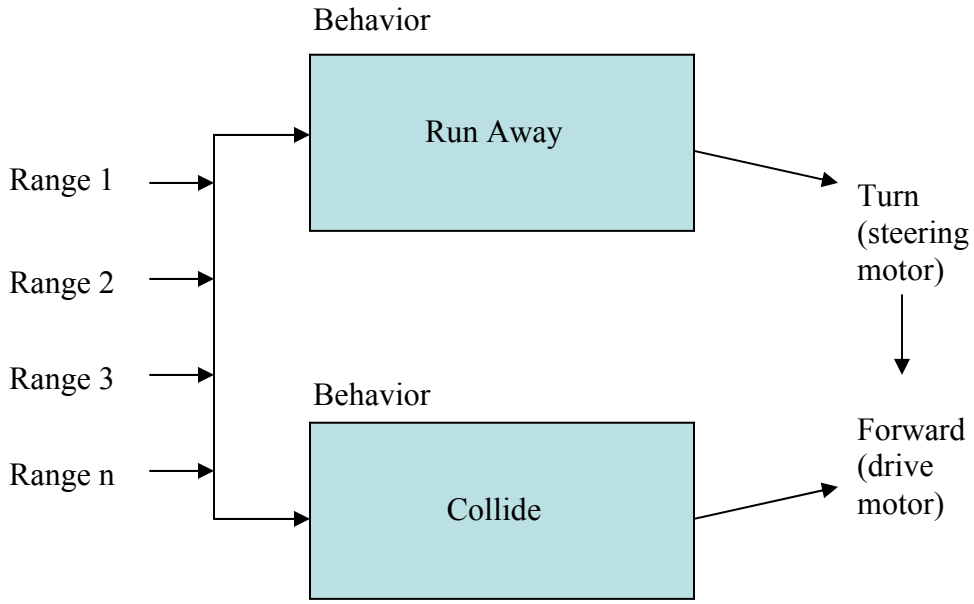

Figure 16: Level 0 recast into primitive behaviors [2].

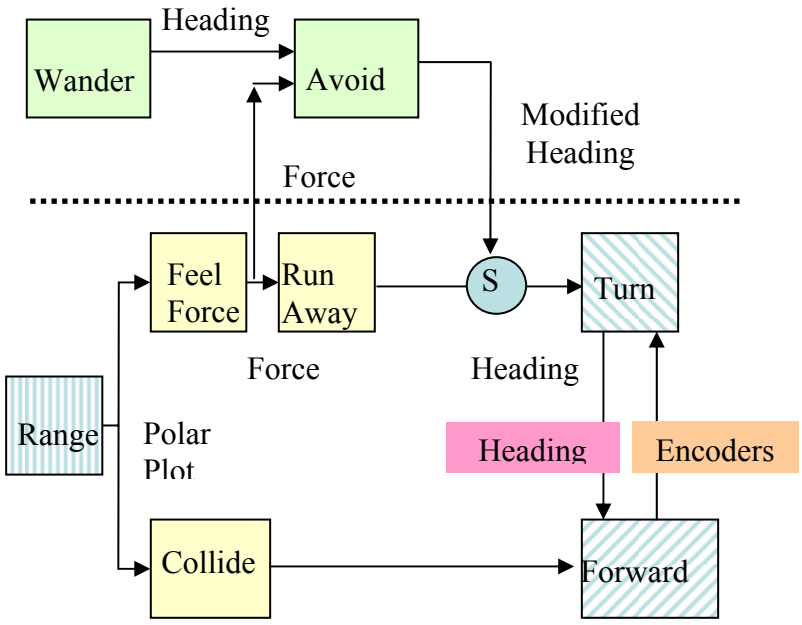

Figure 17: Adding Level 1: The Wander and Avoid Module [2].

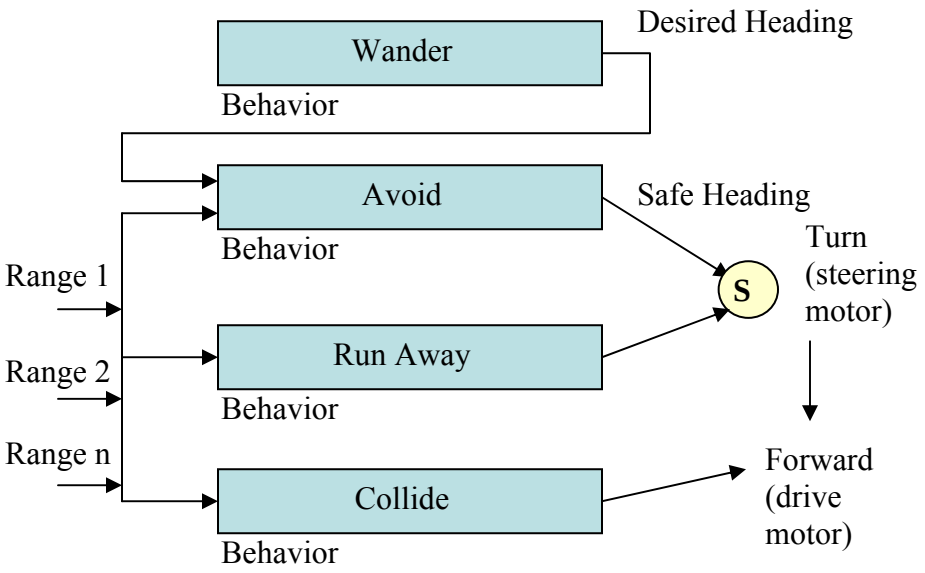

Figure 18: Level 1 recast into primitive behaviors [2].

would not be able to avoid obstacles. This level integrates the *feelforce* vector and the *wander* vector. The addition of this module creates a more sophisticated response to obstacles. The combination of the direction of the force of avoidance with the desired heading results in the actual heading being mostly in the right direction rather than having the robot turn around and lose forward progress. The *avoid* module effectively eavesdropped on a component of the next lower layer. The heading output from the *avoid* module has the same representation as the output of the runaway module, so the *turn* module can accept an input from either source.

Under subsumption, the issue regarding the selection of the heading vector automatically goes to the higher layer; the higher layer *subsumes* the lower one. Subsumption is accomplished by one of two ways. It may be done by *inhibition*. Inhibition connects the output of another '*dummy*' module. If the output of the subsuming module is 'on' or has any value, the output of the subsumed module is blocked or turned 'off.' Inhibition acts like a valve turning output streams on and off. Subsumption is also realized by *suppression*. Suppression connects the output of the subsuming module to the input of another '*dummy*' module. If the output of the subsuming module is on, it replaces the normal input of the subsumed module. Suppression acts like a switch that alternatively chooses one input stream or the other.

In this example, the circled 'S' (Figure 18) indicates that the *avoid* module suppresses the output from the *runaway* module. The *runaway* module is still executing, but its output is not used. Instead, the output from the *avoid* module goes to the *turn* module. The use of the subsumption architecture allows new layers to be built on top of a less competent layer without modifying the lower layers. This type of structure is very conducive to the principles incorporated in software engineering regarding modularity. The layering also has the effect of introducing a degree of robustness. If Level 1 is disabled, Level 0 still has the ability to at least flee from approaching obstacles [2].

2.2.4 Potential Fields

Another style of reactive architecture is based on the idea of *potential fields*. Although the specific architectures that use some type of potential fields methodologies are numerous; a generalization is presented here. Potential Fields styles of behaviors always use vectors to represent behaviors and vector summation to combine vectors from different behaviors to produce an emergent behavior.

There are five (5) basic potential fields: *uniform, perpendicular, attractive, repulsive*, and *tangential*. In a uniform field, the robot would feel the same force no matter where it was. Regardless of placement or orientation, it would 'feel': a 'need' to turn to align itself to the (uniform) vector's direction as well as the need to move in the direction of the vector at a velocity proportional to the length of the arrow. A uniform field is often used to capture the behavior of 'go in direction x.' A perpendicular field is used to force the robot to orient perpendicularly to some object (or wall, or border, etc.).

The field may be directed away, or toward an object. An attractive field is characterized by a circle with all the arrows pointing toward the center. Wherever the robot is, the robot will 'feel' a force relative to the object. Attractive fields are useful for representing a taxis or tropism, where the agent is literally attracted to light or food or a goal. The opposite of an attractive field is a repulsive field. Repulsive fields are commonly associated with obstacles, or things the agent should avoid. The closer the robot is to the object, the stronger the repulsive force away from it becomes. The last potential field is the tangential field. The field is a tangent around the object. Tangential fields can 'spin' either clockwise or counter-clockwise. They are useful for directing a robot to go around an obstacle, or having a robot investigate something.

Potential fields styles of architecture have many advantages. The potential field is a continuous representation that is easy to visualize over a large region of space. As a result, it is easier for the designer to visualize the robot's overall behavior. It is also easy to combine fields. Computer languages such as C++ support making behavior libraries. Additionally, the potential field can be parameterized: their range of influence can be limited and any continuous function can express the range in magnitude over some distance. Furthermore, a two-dimensional field can usually be extended into a three dimensional field, and so behaviors developed for 2D will work for 3D.

Building a reactive system with potential fields is not without disadvantages. The most commonly cited problem with potential fields is that multiple field can sum to a vector with zero magnitude; this is called the local minima problem. In practice however, there are many elegant solutions to this problem. One early method utilizes a motor schema that produces vectors with a small magnitude from random noise. The noise in the motor schema would serve to 'bump' the robot off the local minima [2].

2.2.5 Summary

Reactive systems are limited to applications which can be accomplished with reflexive behaviors. They cannot be transferred to domains where the robot needs to do planning, reasoning about resource allocation, etc. These practices led to the development of the hybrid paradigm.

The organization of the reactive paradigm is sense-act, with no plan component. Sensing in the reactive paradigm is local to each behavior. Each behavior has direct access to one or more sensors independently of the other behaviors. A behavior may create and use its own internal world representation, but there is no global world model (like in the hierarchical paradigm). Reactive systems typically execute faster than other types of systems.

There are several major characteristics of robotic systems constructed under the reactive paradigm. Behaviors serve as the basic building blocks for robotic actions, even though different designers may have different definitions of what a behavior actually is. When behaviors are used, as in reactive systems, the overall behavior is emergent. Only local, behavior-specific sensing is permitted. The use of explicit representations in perceptual processing, even locally, is avoided in most reactive systems. Explicit representations of the world are often referred to as maintaining the state of the world internally (an internal state). Instead, reactive behaviors rely on the world itself to maintain state. Animal models are often cited as a basis for a behavior or the architecture. Behaviors and groups of behaviors which were inspired by or simulate animal behavior are often considered desirable and more interesting. Reactive systems exhibit good software engineering principles, and are inherently modular from a software design perspective. Behaviors can be tested independently, since the overall behavior is emergent. More complex behaviors may be constructed from primitive behaviors, or from mixing and matching perceptual and motor components [2].

The main reason for not choosing this architectural type is the lack of a rich internal model of the world according to critics of behavior-based architectures. The information acquired from different sensors is not fused into a consolidated estimate of the state of the world, since sensors are typically connected directly to action. The sensor-actuator systems generate their own estimates of what behavior is most appropriate. Then, arbitration heuristics select a single best behavior for execution. "Behaviorist

architectures are often not goal-directed and typically do not generate a plan for future behavior [1].

2.3 The Hybrid Deliberative/ Reactive Paradigm

By the end of the 1980's, the use of the reactive paradigm was the trend in the design and programming of artificially intelligent robotic systems. The reactive paradigm allowed robots to operate in real-time using inexpensive, commercially available processors with no memory. However, the cost of reactivity was a system that eliminated planning to any functions which involved the use of memory (remembering or reasoning) about the global state of the robot relative to its environment. This shortcoming prevents planning used to compute optimal trajectories (path planning), making maps, monitoring performance, or selecting the best behaviors to use to accomplish a task (general planning). Not all of these functions involve planning per se; map making involves handling uncertainty, while performance monitoring (and the implied objective of what to do about degraded performance) involves problem solving and learning. In order to differentiate these more cognitively oriented functions from path planning, the term *deliberative* was coined.

The reputation of reactive paradigms also suffered somewhat because most researchers found that designing behaviors so that the desired overall behavior would emerge was an art, not a science. Techniques for sequencing or assembling behaviors to produce a system capable of achieving a series of sub-goals also relied heavily on the designer. Inherently, this implies that robotic systems need to be made smart enough so that it selects the necessary behaviors for a particular task and generates how those behaviors should be sequenced over time.

Putting the planning and deliberation back into robotic systems without disrupting the success of the reactive behavioral control became the challenge for AI robotics. Behavioral control seemed to be the 'correct' way to do low-level control because of its obvious success, and its elegance as a computational theory for both biological and machine intelligence. Many roboticists, however, wanted to add layers of higher, more cognitive functions to their behavioral systems, emulating the evolution of intelligence. In 1988, Arkin published work on how to add more cognitive functions to a behavioral system in the form of the *Autonomous Robot Architecture (AuRA)* [2]. AuRa is a hybrid architecture designed by Arkin that incorporates both planned and reactive behavior in a goal-directed schema-based system. AuRa combines a high-level deliberative hierarchical planner based on traditional AI techniques with a low-level reactive controller based on schema theory [1].

The concept of considering an intelligent system situated in its environment, combined with the existence proof that detailed world representations are not always necessary, led to a new style of planning. This change in planning was called reactive planning. As

many researchers who had worked in traditional AI became involved in robotics, architectures stemming from those derived in the planning field made use of their traditional AI roots. These architectures use a more top-down, hierarchical flavor with global world models, especially Saphire and TCA. Examples of other hybrid architectures stemming from the behaviorist camp are Sensor Fusion Effects (SFX), and 3T [2].

The problem of exactly how to include reactive elements can be addressed from the two perspectives of 'top-down' and 'bottom-up.' A more hierarchical, top-down approach necessitates that behavioral importance is determined by behavioral goals selected on the basis of evaluations by value judgment processes in the limbic system (brain). The intelligent system is thus driven by high-level goals and priorities to focus attention on objects specified by task goals or identified by task knowledge as necessary to accomplish task goals successfully. High-level goals and priorities can also generate expectations of what objects and events are likely to occur during the task and which of these are most important for achieving goals [1].

Regardless of the bottom-up or top-down inspiration for including non-behavioral intelligence, architectures which use reactive behaviors, but also incorporate planning, are now referred to as being part of the *hybrid paradigm*. At first, hybrids were viewed as an artifact of research, without any real merit for robotic implementations. Some researchers went so far as to recommend that if a robot was being designed to operate in an unstructured environment, the designer should use the reactive paradigm. If the task was to be performed in an easy-to-model or knowledge-rich environment, then the hierarchical paradigm was preferable, because the software could be engineered specifically for the mission. Hybrids were believed to be the worst of both worlds, saddling the fast execution times of reactivity with the difficulties in developing hierarchical models. Currently, many roboticists believe that hybrids are the best general architectural solution for several reasons. First, the use of asynchronous processing techniques (multi-tasking, multi-threading, etc.) allow deliberative functions to execute independently of reactive behaviors. A planner can be slowly computing the next goal for a robot to navigate to, while the robot is reactively navigating toward its current goal with fast update rates. Second, good software modularity allow subsystems or objects in hybrid architectures to be mixed and matched for specific applications. Applications which favor purely reactive behaviors can implement just the subset of the architecture for behaviors, while more cognitively challenging domains can use the entire architecture [2].

2.3.1 Attributes and Fundamental Issues

2.3.1.1 Fundamentals

The organization of a hybrid deliberative reactive system (Figure 19) can be described as: plan, then sense-act. The plan box includes all deliberation and global world modeling,

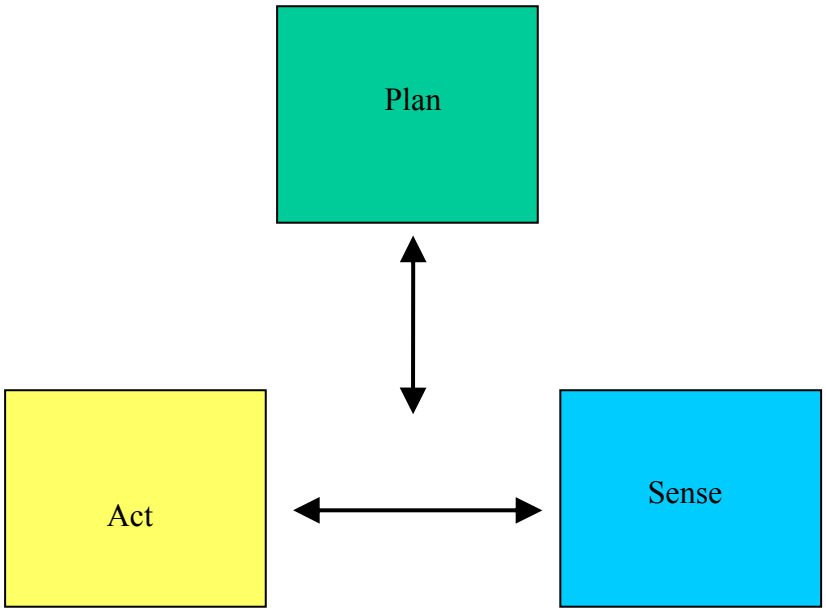

Figure 19: Plan, sense-act organization in the hybrid deliberative reactive paradigm [2].

not just task or path planning. The robot would first plan how to accomplish a mission or a task (using a global world model), then instantiate or turn on a set of behaviors (sense-act) to execute the plan (or a portion of the plan). The behavior would execute until the plan was completed, then the planner would generate a new set of behaviors.

The idea of plan, then sense-act evolved from two assumptions of the hybrid paradigm. The first assumption is that *planning covers a long time horizon and requires global knowledge*, so it should be decoupled from real-time execution (this is similar to the software engineering principle of coherence - dissimilar functions should be placed in different objects). This helps to set objectives and select methods, but does not work well for making finely grained decisions. The second assumption is that *deliberation works with symbols* (the goal is to pick up a 'Coca-Cola can'). This is in contrast to how reaction works with sensors and actuators (the percept is a 'red blob' which exerts an attractive field). Since planning and global modeling algorithms are computationally expensive, from a standpoint of practicality, they should be decoupled from real-time execution because they would slow down the reaction rate.

The organization of sensing in hybrid architecture is more complex in that sensing is truly hybrid. In the behaviors, sensing remains as it was for the reactive paradigm: the

behaviors are local and specific. But planning and deliberation require global world models. Since planning functions must have access to a global world model, the planning model is constructed from processes independent of behavior-specific sensing. However, both the perceptual schemas for the behaviors and the model making processes can share the same sensors. Furthermore, the model making processes can share the percepts created by the perceptual schemas for behaviors (eavesdrop), or it can have sensors which are dedicated to providing observations which are useful for world modeling but aren't used for any active behaviors.

The organization of the sense, plan, and act primitives in the hybrid paradigm is conceptually divided into a reactive (or reactor) portion and a deliberation (or deliberator) portion. Although many architectures will have discrete layers of functionalities within the reactor and deliberator, each architecture in the hybrid paradigm has obvious partitions between its reactive and deliberative functions [2].

2.3.1.2 Attributes

The hybrid paradigm is an extension of the reactive paradigm, and from the above description, it appears that the behavioral component is also undifferentiated, but that is not entirely true. Behaviors in the hybrid paradigm have a slightly different connotation than in the reactive paradigm. In the reactive paradigm, 'behavior' connotes purely reflexive behaviors. In the hybrid paradigm, the term 'behavior' is usually more consistent with the ethological use and includes reflexive, innate, and learned behaviors. This can be confusing, and at least one architecture uses the term 'skill' instead of 'behavior' to avoid confusion with purely reflexive behaviors. Also, hybrid implementations tend to use clusters of behaviors sequenced over time, rather than primitive behaviors. Because the hybrid implementations are interested in more complex emergent behaviors, there is more diversity in methods for combining the output from concurrent behaviors.

The term 'global' is used almost synonymously with 'deliberative' and 'local' with 'reactive.' This can lead to significant confusion, because 'global' may not always be truly global in hybrids. The deliberative portion of a hybrid architecture contains modules and functions for things which are not easy to represent in reactive behaviors. Some of these functions require a global world model; path planning and map making are probably the best examples. But other activities require global knowledge of a different sort. Behavioral management (planning which behaviors to use) requires knowing something about the current mission and the current (and projected) state of the environment. This is global knowledge in that it requires the module to know something outside of itself, as compared to a reactive behavior which can function without any knowledge of whether other behaviors are actively executing [2].

2.3.2 Advantages and Disadvantages

The advantage of the hybrid paradigm is that is combines the best of both the hierarchical and the reactive paradigms. By comparing the differences between the various hybrid deliberative reactive architectures, the advantages become clearly distinguishable. Hybrid architectures fall distinctly into three (3) areas based on the following:

- How does the architecture distinguish between reaction and deliberation?
- How does it organize responsibilities in the deliberative portion?
- How does the overall behavior emerge?

The difference between reaction and deliberation is a critical issue in building a successful, reusable object-oriented implementation. The sensing organization of the hybrid paradigm is shown in Figure 20. This determines what functionality goes in what modules, what modules have access to global knowledge (which leads to specifying public and friend classes in C++), and precisely what the global knowledge (shared data structures) should be. Likewise, it is important to subdivide the deliberative portion into modules or objects. A good decomposition will ensure portability and reusability. While hybrid architectures are most noteworthy for how they incorporate deliberation into mobile robotics, they also introduce some changes in the way reaction is organized. Many researchers found the two primary means of combining reactive behaviors, Subsumption and Potential Fields summation to be limited. Since then at least three other mechanisms have been introduced: *voting* (in the DAMN architecture); *fuzzy logic* (Saphira); and *filtering* (SFX).

The major contribution of hybrid architectures is that they combine deliberation and reactive control by various methods of interleaving the two. By 1990, algorithms that could allow robotic systems to adequately compute an optimal path for both two-dimensional and three-dimensional maps existed. These path planning algorithms exhibited several drawbacks. One was that they were all computationally expensive. This prevented the robotic system from continuously generating the path. If the robotic system tried to execute a pre-computed path, it would be vulnerable to unexpected changes in the world. That map would represent the best, currently available knowledge about the world: where obstacles are located or other impassable terrain. But a map is at best a representation of a closed world; it can't show what has changed since it was built, unless a robot goes there and senses it. Even though the robotic system can generate an optimal path, it may find an un-modeled obstacle blocking the path. If the robot was just using the pre-computed path to navigate, it would have to stop, update the map with the newly discovered obstacle, then re-plan the optimal path, resulting in slow progress.

The solution mixes path planning and reaction by having a planning algorithm in the cartographer generate a complete optimal route, decomposing the route into line segments with a *waypoint* at the end. Each waypoint is a goal to reach which is accomplished by behaviors. When the robot reaches the first goal, the sequencer agent can give the behavioral manager the coordinates or landmarks for the next goal, etc. This

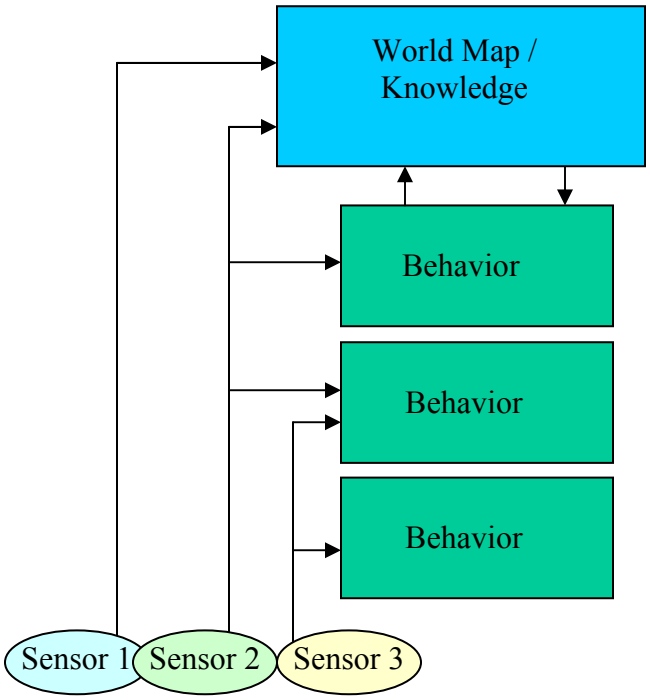

Figure 20: Sensing Organization in the Hybrid Paradigm [2].

allows computationally expensive path planning to occur only once, and then the behaviors take care of actually navigating.

This strategy, however, has drawbacks but compels the idea that mismatches in planning and reaction times are no longer valid reasons to enforce a strict separation of deliberation and reaction. However, the software engineering reasons for the partition remain the same: things which operate on symbols and global information should be in the deliberator: things which operate directly on sensors and actuators should be in the reactor. In this example of top-down interleaving of deliberation and reaction, the deliberative layers decompose the mission into finer and finer steps until it arrives at a set of behaviors capable of accomplishing the first sub-goal [2].

2.3.3 Summary

Hybrid architectural styles can be loosely divided into three categories. *Managerial* styles subdivide the deliberative portion into layers based on the scope of control (managerial responsibility) of each deliberative function. A Mission Planning module would be able to direct other, subordinate deliberative modules such as navigation, because Mission Planning (where to go) is more abstract than Path Planning (how to get there). *State*

Hierarchies use knowledge of the robot's state to distinguish between reactive and deliberative activities. Reactive behaviors are viewed as having no state, no self-awareness, and function only in the present. Deliberative functions can be divided into those that require knowledge about the robot's past state (where it is in a sequence of commands), and those that are about the future (mission and path planning). *Model-oriented* styles are less precise. They are distinguished by behaviors that have access to portions of a world model. These styles often appear as if they have returned to the hierarchical paradigm model.

The primary contribution of hybrid architectures is to provide a template for merging deliberation and reaction. However, interleaving the two functions is often task dependent. In navigation, there is the issue of interleaving path planning and reactive execution of the path. Another issue is monitoring the progress of behaviors and terminating behaviors correctly.

The hybrid paradigm uses global models only for symbolic functions. The frame problem (with respect to the hierarchical paradigm) is non-existent or very minor because execution is reactive and therefore well-suited for unstructured environments, and also by the fact that software agents can use agent-specific abstractions to exploit the structure of an environment in order to fulfill their particular role in deliberation. Global models are generally closed world, but the world is 'closed' only at the deliberative level. The robotic system under the hybrid paradigm can think in terms of a closed world, while it acts in an open world.

One of the major influences on the mobile robotic community has been the DARPA UGV projects, which advanced the state of the art in outdoor ground vehicle control and navigation. The DARPA project essentially allows the vehicle to have knowledge of a map and directions to a goal location with all the locomotion and navigation performed by autonomous control. These tasks are well-suited for a hybrid approach. The European ESPRIT agency also influenced mobile robotics sponsoring research in automating intelligent vehicle highway programs. A third source of influence has been the effort by NASA to develop autonomous planetary rovers. The rover's primary purpose is to map a planet's surface, which like path planning, is an intrinsically deliberative function [2].

2.4 The Joint Architecture for Unmanned Systems (JAUS)

2.4.1 Introduction

The purpose of JAUS is to provide interoperability between various unmanned systems and subsystems for both military and commercial applications. JAUS seeks to achieve this through the development of functionally cohesive building blocks called components whose interface messages are clearly defined. In the language of JAUS, a number of terms are used to delineate position within the overall hierarchy of the system. These

terms describe the different levels of the architecture and often imply an internal hierarchical sub-grouping. The levels are: System, Sub-System, Node, and Component.

A system consists of one or more sub-systems. A sub-system consists of one or more nodes and is usually thought of as a single vehicle. A node consists of one or more components and is typically thought of as a single computing device. A component represents the lowest level of decomposition within the JAUS reference architecture and performs a specific function. An important part of JAUS is the specification of the messaging or interfaces between components. The interface defines what information gets passed to and from the component, thereby indirectly constraining the function of the component. The interface does not and should not specify how the function is carried out. This leaves the implementation details to the various systems engineers. Implementing JAUS in a system can significantly streamline the design and prototype development with respect to the integration of all subsystems.

In addition, future upgrades can be made to the system on a *modular* basis. JAUS defines a set of reusable *components* and their interfaces. These reusable components not only reduce the maintenance costs of a system, but also dramatically reduce the development costs of any follow-on system(s). Reuse allows a component developed for one unmanned system to be readily ported to another unmanned system or to be easily replaced when technology advances. Components that are deemed necessary for the mission of the Unmanned System may be inserted simply by bundling. JAUS defines components for all classifications of Unmanned Systems from remote control to systems capable of autonomous operation, and regardless of application. As a particular system evolves, the JAUS *technical architecture* will already be in place to support evolution of the system.

2.4.2 Technical Constraints

2.4.2.1 Platform Independence

Analysis has shown that Unmanned Systems will be based on a variety of mission requirements, including but not limited to surveillance, reconnaissance, force protection, combat, security, and emergency response. In order for JAUS compliant components to be interoperable, no assumptions about the underlying vehicle are made.

2.4.2.2 Mission Isolation

The purpose of Unmanned Systems is to gather information, alter the state of the environment, or both. The set of these sensing and effecting tasks are called missions. The purpose of this isolation is the anticipation that many Developers will build their systems to support a variety of missions, possibly with removable mission modules.

2.4.2.3 Computer Hardware Independence

The growth in the computer industry has been enormous over the past 20 years and there are no indications that the growth will slow down. Future Unmanned Systems must be able to capitalize on commercial advancements in computing and sensor technology. The issue is two-fold:

- First, a single Unmanned System must be able to evolve over its product life cycle to accommodate new missions and greater degrees of autonomy. An architecture that imposes a specific hardware implementation reduces the opportunity to take advantage of future technical advancements.
- Second, each Unmanned System Developer should have the flexibility to design a computer hardware architecture that meets that particular system's requirements. Computer hardware that is appropriate for one Unmanned System may not be appropriate for another.

2.4.2.4 Technology Independence

The final technical constraint is technology independence. This constraint is similar to computer hardware independence but focuses more on the technical approach rather than the computer hardware. For example, an Unmanned System may use a visual system for numerous purposes. A visual system may provide feedback to the Unmanned Systems operator to support tele-operated driving. However, there may be other techniques besides visual systems and edge detection that could perform obstacle detection. Active Laser Detection and Ranging (LADAR) is one possibility. The point is that there may be multiple technical solutions to a problem. An architecture that is built around a particular technology solution may eliminate a superior alternative.

2.4.3 JAUS Nomenclature

In the language of JAUS, a number of terms are used to delineate position within the overall hierarchy of the system and must therefore, be well understood. These terms describe the different levels of the architecture and define the required internal hierarchical sub-grouping.

- System — A system is a logical grouping of subsystems. The system definition provides a functional grouping for the full robotic or unmanned capability. This grouping includes all human interface subsystems and unmanned subsystems common with robotic and unmanned applications.
- Subsystem — A subsystem performs one or more unmanned system functions as a single localized entity within the framework of the System. A subsystem shall provide one or more communication command and control capabilities. A mobile subsystem shall execute mobility commands as a single unit and retain a defined center of gravity relative to all articulations and payloads.

- Node — A JAUS Node defines a distinct processing capability within a subsystem. A node retains a set of coherent functions and shall provide a node manager component to manage the flows and controls of JAUS message traffic.
- Component — A component provides a unique functional capability for the unmanned system. JAUS messages are defined with respect to these capabilities so that context in command and control is provided. A JAUS component resides wholly within a JAUS Node.
- Instance — Duplication and redundancy of JAUS Components are provided by Component Instances. All Components are uniquely addressable using Subsystem, Node, Component and Instance Identifiers.
- Message — A JAUS message is comprised of the message header and associated data fields as defined within this document.

2.4.4 System Topology

A graphical representation of how each element fits into the JAUS system topology is shown in Figure 21.

2.4.4.1 Topology Simplified

Now that the five most elementary JAUS terms have been defined, a simple and direct statement can be made that gets to the heart of how they work together:

- *A subsystem is composed of component software, distributed across one or more nodes.*

From a systems engineering standpoint, this last statement is significant in that it implies, yet does not specify. Since configurations can be virtually unlimited, node or component interface boundaries are not dictated. This is one of the key aspects of JAUS flexibility.

2.4.4.2 Configuration

One of the principal goals of JAUS is to provide a level of interoperability between intelligent systems that has been missing in the past. Towards this end, JAUS defines functional components with supporting messages, but does not impose regulations on the systems engineer that govern configuration.

JAUS does have one absolute, unwavering requirement that can effect configuration. It is:

- *To achieve the desired level of interoperability between intelligent computing entities, all messages that pass between JAUS defined components (over networks or via airwaves), shall be JAUS compatible messages.*

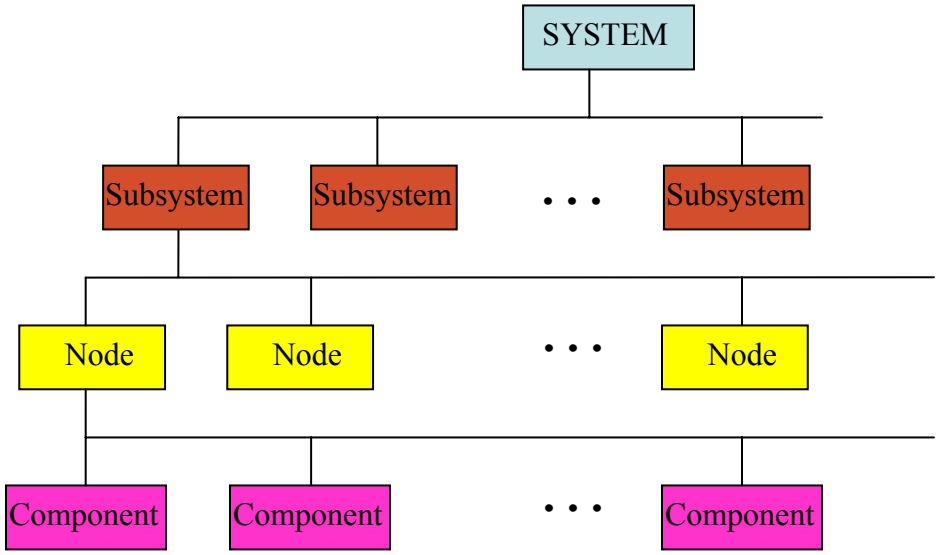

Figure 21: JAUS System Topology [40].

Consider a message as a means of exchanging information. A close examination of this statement reveals that *messages* pass between *components* via some communications medium. It is important to understand that this restriction only applies to JAUS messages [40].

3 THE SENSOR BRICK SYSTEM

Achieving the maximum possible benefit from intelligent mobile robotic systems requires the use and integration of common control structures. The proper implementation of these structures is what enables the collection, processing, analysis, integration, dissemination and reuse of information and technology, and drives the system toward a state of *maximal operational effectiveness*. The control systems provide the attributes that are functionally necessary that allow different systems to be *interoperable*, and for sub-systems of one system to be able to be used (*interchangeable*) on another system. Systems interoperability and interchangeability are a crucial component with respect to gaining *information superiority* (control of the maximum possible information from the full spectrum of information in any given situation or paradigm) [8].

3.1 System Development

Addressing the issues concerning information superiority and interoperable control require utilizing the concept of contemporary robotic *systems development* and the theory of *operational control*. The current state of system development in mobile robotics is mired by the *specificity of design* issue, discussed in Chapter 1. Commercial systems are plagued by a lack of interoperability, and a lack of interchangeable subsystems. The sub-systems they do have are not modular and are not interchangeable between systems. The Sensor Brick system (SBS) overcomes the deficiencies of the commercially-built systems by emphasizing that the modular, self-sustaining design of the system allows for the easy interchange of sensor bricks. The issues related to interoperable control are more readily addressed by a system such as the SBS, as the Joint Architecture for Unmanned Systems (JAUS), a technical architecture, may be easily included to allow the system to conform to the specification. By selecting the SBS as the *system architecture*, the issues that are the most problematic with respect to mobile robotic systems are addressed. The issues of interoperability and interchangeability, the main issues with commercially-built systems, are easily overcome due to the modular, independent nature of the SBS.

3.1.1 Influence of Operational Control Architecture

The system architecture must also be ratified by the *operational architecture*. The operational architecture drives the design of the system through integrating the software implementation of the control structures (*operational architectures*) and the standards and specifications set forth in the *technical architecture* with respect to the computing and communication equipment [8].

The evolution of intelligent mobile robotic systems, such as the SBS, must incorporate all three (3) architecture types into a comprehensive system of coordination and control. Intelligent mobile systems require advanced, adaptive, and interoperable architectures that are based on proven engineering principles and follow industry standards. These architectures must be object-oriented and be in support of object-based distributed computing. Additionally, they must support collaborative planning in an unreliable bandwidth environment, be robust enough to withstand harsh operational conditions, exhibit an acute degree of situation awareness, and be able to intelligently disseminate information to the end-user while providing course of action analysis and integration with a simulation, testing, and validation environment [11].

3.1.2 Inter-Relationship of the Control Architectures

These architectures must be easily integrated to enable many different systems to operate together and be reusable. They also must be formatted so that the presentation, use and application occur in a consistent manner. For these reasons, the control structures for any intelligent mobile robotic system have been identified and grouped in the three (3) major categories (operational architecture, technical architecture, and system architecture).

The theory and overview of operational architecture for intelligent mobile robotic systems is discussed in Chapter 2 of this thesis. Operational architecture describes the tasks, operational units, and information flows that are required to accomplish an application or mission. Essentially, it is concerned with task decomposition, the hierarchy of command, and the priorities and information flow of the system.

Technical architectures contain sets of rules that govern the organization, interaction, and interdependence of the system components. This facilitates interoperability when the system's or systems of system's components conform to the specification. The technical architecture also specifies the conceptual paradigms of the processing, database, and communications as well as the standards and data dictionary. The technical services, hardware interfaces and standards, computer and engineering specifications, and communication interfaces and standards are addressed by the technical architecture.

The system architecture describes the physical system components and interconnections that integrate for particular applications or missions. The systems architecture is constructed to satisfy operational architecture requirements from standards defined in the technical architecture. The particular mobility platforms, sensor units, and actuators are addressed by the system architecture. "*A system is realized through integrating the software implementation of the operational architecture and the standards and specification as set up in the technical architecture to the physical computing and communication equipment shown as the systems architecture*." The system, operational and technical architectures' relationship is depicted in Figure 22 [8].

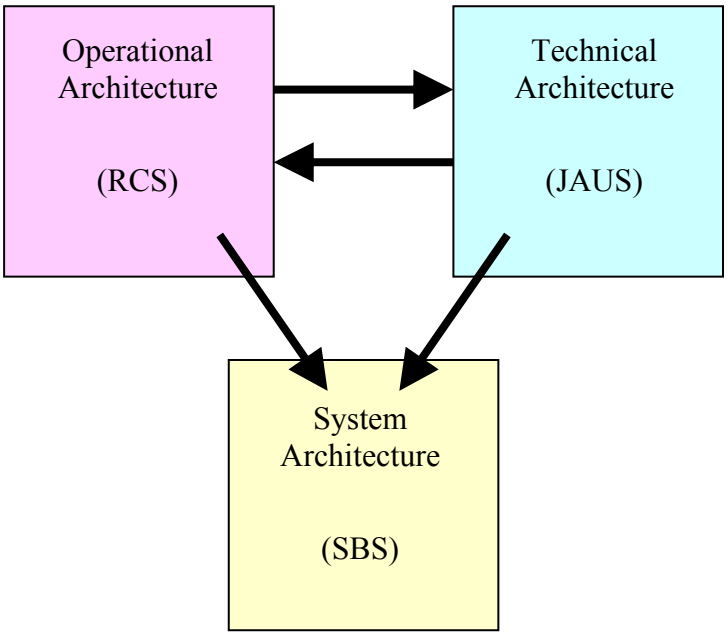

Figure 22: The relationship of the architectures for mobile robotic systems [8].

3.1.3 Formulation of the System Concept

By applying the concept of *modularity* in every aspect of development of the SBS, the range of possible applications in which robotic-sensor units (bricks) can be applied increases. The application of modular design *does not constrain* any of the operational or technical architecture requirements that drive the development of the *system architecture* of the SBS; instead it *promotes* the interoperable control and interchangeable payloads or components within and between systems, and decreases or eliminates the problems associated with the specificity of unique designs.

The SBS described herein, is a real, operational system designed and built in the University of Tennessee's, Imaging, Robotics and Intelligent Systems (IRIS) Laboratory. However, a *modular system* may take many forms. In this sense, the research described herein is a generic framework.

3.2 Theory of Modular Design

3.2.1 Overview

At present, the design of intelligent, mobile robotic systems is undergoing fundamental changes caused by developments in information technology and by the ongoing pressure

on productivity and costs. This change is defining new automation structures and principles which open up convenient capabilities in the application of these technologies.

What is of central importance in this respect is the further development of automation technology which will lead to more open and widely distributed automation systems. These developments are based on the demands frequently made by users for *openness and universality*, and the wish for *ease of use of the technology* which is becoming increasingly more complex.

The benefits for the user relate to time and cost savings when preparing and operating the machines and systems. As the demand for robotic products and customized applications increase, the companies that manufacture these products must design them in a manner so that they satisfy the market, or create a new market upon their instantiation.

For mobile robotic systems, the market calls for operationally effective systems that can be made as elementally simply and uncomplicated as possible while retaining robust and independent operationally capability with outstanding reliability and self-reliance. They must be useful in many different areas of application and be maximally effective. In general, these statements characterize the current global state of the technologies market. For intelligent mobile robotics, the creation and support of the SBS provides a template that can be used to satisfy the demand for flexible and easy-to-build intelligent mobile robotic systems for the beginning of the twenty-first century. The template does not mandate a certain style or form that has to be used; instead it prescribes a method that shows how systems can be constructed using the framework provided herein. Essentially, the principle of modularity requires *machines and systems to be split into autonomously operating sub-units which are able to coordinate with each other using a manageable number of inter-related sub-systems*.

Modularizing products provides a method to achieve a flexible product that enables the goals mentioned above. The present state of most robotic systems manufacturers requires a constant fine-tuning of the manufacturing processes in order to lower costs and maximize the efficiency of production. The result is a basis where fewer parts (or subsystems) are used to make the same or a greater amount of product variants than before the implementation of modularizing program. In general, implementing modularization of the products enables a variety of products at low cost to be offered in a market by being able to combine a set of modules in order to get the desired variant of the product.

Although initially, the modularization process forces a reorganization of the product structures, conducting research of which parts to redesign and how, reorganizing their manufacturing processes and their standard procedures. Although a total re-adaptation from no modularity to a modularized state can cause the initial changeover costs to be high, as the modularization process maturates through its completion it pays back the

initial capital investments in money and in time. Implementing modularization may soon be a requirement for the companies that wish to stay competitive because of the complex product markets of the twenty-first century. These markets will demand that the company have the ability to quickly and globally deliver a high variety of customized products. A modular approach is one method that can achieve this result [16].

3.2.2 Advantages of Modular Design

Some of the advantages of modular design include:

- Modules are easier to handle than large, complex, uniquely fabricated constructs – this improves situational clarity and allows errors to be located more easily
- Different modules that describe the same process can be exchanged – this allows object oriented testing of alternative designs
- Improvement of re-use – modules are intended to be generic – large, complex, uniquely built units are difficult to reuse
- Minimization of redundancy – this minimizes risk and exposure to inconsistent design approaches

Although there are no general rules as to how to apply modular design, and the results of modularization are not unique, three (3) key aspects must be considered as modular design is instantiated:

- In most cases, the variables, processes and systems that belong together are known. The phenomenology (in this case, intelligent mobile agents) under consideration provides the clues to the modularized approach.
- Based on mathematics and systems theory, if the number of modules is given, the sum of interactions with a module, divided by the sum of interactions between modules is constrained to a minimum which imply that the inputs and outputs of different modules which describe the same process should be equal
- To minimize the risk of inconsistencies, minimize redundancy [16].

3.3 Description of the Sensor Brick System

The essential unit of the SBS is the *sensor brick*. A central and essential principle of modular design requires the modules (sub-units or sub-systems) to be capable of autonomous and independent operation. In the SBS there are three (3) modules, or sub-systems, termed '*blocks.*' The three blocks are:

- Mobility Bricks
- Intelligent (Software, Communication, and Computer) Systems
- Sensor Bricks

Figure 23 shows a depiction of the three (3) blocks of the system which form a *sensor brick*. In the same way that the sensor bricks form the units that comprise the SBS, the blocks form the sensor bricks.

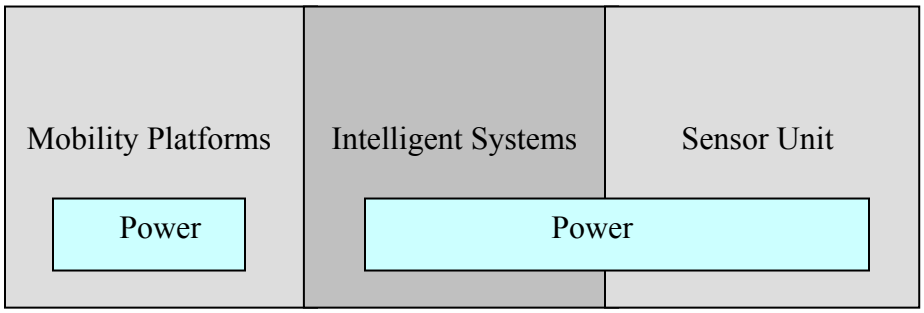

Figure 23: An iconic representation of the three (3) components in the SBS.

The Sensor Brick's modular design provides two key advantages. First, each sensor brick is constructed using inexpensive and commercially available components. Secondly, the required components are selected or designed so that they can easily be replaced or exchanged, enabling the devices to act as 'plug-and-play' modules. Each block operates as an independent, interchangeable module. This philosophy also requires that each sensor block be able to mount onto any of the mobility drives, and that any of the mobility drives are capable of having any of the sensors attach to it.

This flexible, inter-connective architecture allows the use of any sensor unit on any mobility unit. Essentially, the sensor blocks become plug-and-play devices for the mobility platforms. By its nature, highly modular design identifies and segments the operational units with respect to the required functionalities of the major subsystems. This lowers cost by minimizing the number of subsystems required and reduces the reliance on proprietary technology by using inexpensive and readily available commercial components. By requiring that the sensing and mobility units be readily interchangeable builds redundant capabilities into the functional design. Only by incorporating modular design can the system attain its full realization of physical and logical independence. The ideas of physical and logical independence enable the operational control requirements that ultimately lead to the attainment of an intelligent, self-diagnosing system. These robotic systems sense and drive autonomously, uploading or downloading various raw and processed data to the appropriate command and control structures while allowing operator intervention or other forms of control (teleoperation) if necessary. These characteristics provide the inherent advantages that allow intelligent, mobile and modularly-designed robotic systems to be effective in many different areas of application.

3.3.1 Components of the Sensor Brick System

3.3.1.1 Mobility Bricks

The *mobility bricks* consist of both in-house fabricated and commercially built units. The idea is to use the various mobility platforms by incorporating their respective control systems into the SBS. Reverse-engineering the control system can allow intelligent software applications to control the host's motors at the physical level. If the locomotion control signals can be decrypted, then it is possible to use it as a mobility block. Remotec's ANDROS can be used as a mobility block in this way.

Reverse engineering the locomotion control, and applying intelligent navigation algorithms to the navigation control, the new mobility brick is re-named *Andibot* (Figure 24). Once the control system kernel is decrypted, the mobility and drive system can be linked to the intelligent navigation systems capability, enabling it to be utilized and integrated into the SBS.

Similarly, the Segway system (Figure 25) is another mobility device that can be easily transformed into a mobility brick (*Segbot*) by employing the same reverse-engineering methods. Other commercially produced units may also be incorporated in a similar fashion; once the control system is known, it can be easily assimilated into the SBS as a mobility block. Even though these commercially manufactured platforms may not be modularly constructed, they can be used as modular mobility platforms!

The *Safebot* (Figure 26) mobility platform is a low- profile under-vehicle device which is designed and developed in a true modular fashion at the University of Tennessee, IRIS Laboratory. It has a large (approximately 45 cm^2) load bay placed between two independently powered and controlled tracks. Safebot requires an environment that has a smooth, hard surface devoid of significant debris. The physical structure of Safebot suggests this mobility brick is best when used in low-profile applications.

The modular design is incorporated in two key ways. The flexible method by which sensor blocks are loaded onto Safebot is the first key modular attribute. The sensor bricks are simply placed into the payload bay, where each independently powered sensor block may be inserted, removed, or replaced by other sensor blocks depending on the mission.

The sensor blocks power supply does not depend on the power supply of the mobility platform, and vice versa. The maximum payload weight for Safebot is on the order of 50 kg. Figure 26 depicts the Safebot without a sensor mounted in the payload bay.

Figure 24: Andibot (Developed by Roselyne Barreto).

Figure 25: Segbot (Developed by Allen Kemp).

Figure 26: Safebot, developed in the University of Tennessee IRIS Laboratory.

3.3.1.2 Intelligent Systems

With the Visual Sensor loaded in the payload bay ready for an inspection operation, Safebot's mobility and drive system may accept commands for locomotion and navigation of the necessary path. This can be accomplished in several different ways. A human can manually control the trajectory of the inspection brick, or *artificial intelligent systems (control methods)* may be utilized in part or in whole to make the unit more functionally autonomous.

The navigation of the mobility platform is accomplished by the use of algorithms that are concerned with path planning, localization, obstacle avoidance, and actuator input, output and control. The intelligent systems concerning navigation, utilize algorithms such as the A* algorithm, or other methods or their derivatives that are concerned with autonomous navigation. Input for the navigation intelligent systems is carried out by the laser range sensor, discussed fully in a later section. Other intelligent systems applications used in the system are

- Mosaicing Algorithms - used for concatenating visual and thermal images
- Pseudo-Coloring Algorithms – used to colorize grayscale images
- 3-D Reconstruction of Images (data fusion)
- Radiological Source Localization Algorithms.

The algorithms listed above are developed by students in the IRIS Laboratory, and are described subsequently in conjunction with the demonstration and use of the SBS in the

under-vehicle inspection scenario. Other intelligent systems applications make use of image feature extraction, other forms of data fusion, data combination, and other decision engines that use sensory input to help make intelligent choices concerning the system.

In general, the intelligent systems process acquired data to provide information that is used for planning or making decisions for the next instance of operational control process.

3.3.1.3 Sensor Blocks

The critical component of each brick is the sensor block. The sensor block acquires data that is used by the intelligent systems. Each of the four (4) sensor bricks of the SBS, their components and functionalities, are more fully described in the next section.

3.4 Sensor Bricks

3.4.1 Introduction to the Sensor Brick Concept

Figure 27 depicts an iconic representation of the *sensor brick*. The figure is comparable to the iconic description of the *SBS*. Sensor bricks exhibit two major aspects that are directly related to the influence of modular design. The first aspect is that only the four (4) essential subsystems that are necessary to have the sensor brick acquire, pre-process and transmit the image data, are identified as the modules, sub-systems, or '*blocks*.' These four blocks of the sensor brick are:

- Power
- Acquisition (Sensor)
- Intelligent Systems & Computing (Pre-Processing)
- Communication

The second aspect of modular design specifies that each sensor brick must be operationally independent; it does not depend on any other device(s) to maintain operation. Two aspects of the design affect this idea: the design of the *power system* and the *embedded operational directives*. Separate power supply systems keep the mobility bricks and the sensor bricks independent of each other's power supply. If one brick fails, the other brick is not affected. Designing a system that uses separate power supplies enables the sensor brick devices to be seamlessly installed or mounted on other mobility platforms, facilitating the interoperability of payload (sensors). In addition to the separate power supplies, the intelligent systems may also use *embedded directives* (in the operational architecture) to keep the block operationally independence. During mobile robotic operations, communication can not be expected to be maintained under all conditions, so the intelligent agent (in this case the sensor brick's intelligent systems) keeps an embedded higher-layer of intelligent that acts as the directives provider, that in the event of a communications outage, controls the brick continually so that its mode of execution under this condition is uninterrupted until a communication re-establishment

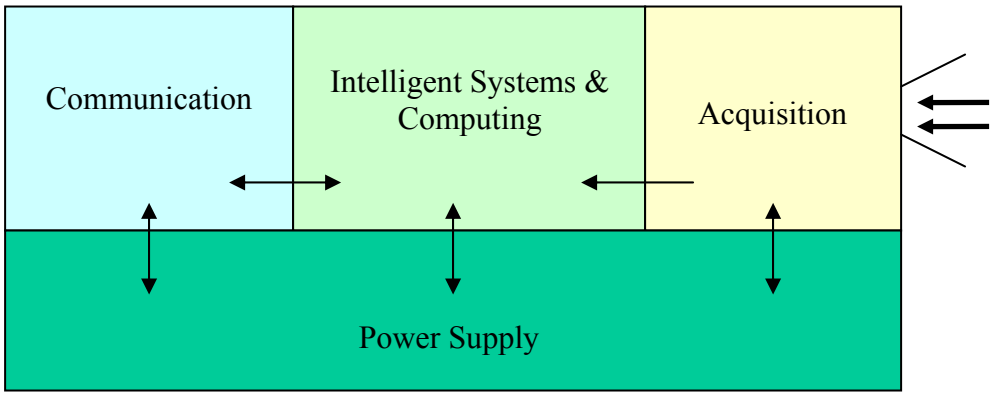

Figure 27: An iconic representation of the Sensor Brick.

The 'out-of-communication' directives sit at a high level in the hierarchical chain of command, and acts only under certain conditions (communications blackout, communications system failure, etc).

3.4.2 Power Systems

The power systems of the sensor bricks consist of a battery, DC to DC converters, fuses and wiring. The batteries in each sensor brick are conventional 12 volt, lead-acid battery. The SBS has adopted 12 volts as the standard for battery voltage The DC to DC converters act to provide the proper power supply to each of the power system components (the acquisition sensors, and the computing and communication systems). A provision is also made on each of the bricks for the use of transformed AC power for recharging and directly powering the sensor blocks.

3.4.3 Intelligent Systems & Computing

Computing systems on the sensor blocks can be any computing device that can store, pre-process and send data over a radio frequency communication link. Some pre-processing operations of the on the visual, thermal, and laser range bricks is accomplished through the use of a frame grabber card that is installed on a PCI slot of the motherboard. The frame grabber is a high speed flash memory device that shuttles groups of data taken by the acquisition devices as image frames for use as output to video screens, or for further pre-processing operations.

The intelligent systems applications concerned with image pre-processing operations make use of image feature extraction, data fusion, data combination, and other techniques and methods that use sensory input to help make intelligent choices concerning the system. Intelligent navigation is provided by input from the laser range scanning system.

A smart algorithm uses the range data to localize and sweep a specified area. All command levels are affected by intelligent systems operations. At each level of hierarchical command, the intelligent systems process acquired sensory data is used too provide information in the planning or decision points for the next instance of the operational control process.

3.4.4 Communication

Communication between the brick and operator interface is accomplished by the use of a wireless communication card (wireless Ethernet) that attaches to the expansion slot of the computer motherboard. Any number of networks can be utilized with wireless Ethernet devices. The SBS typically uses a wireless router to issue high level commands (begin operations, emergency stop and shutdown, etc.) to the various sensor bricks. The advent of technology allows communication between the sensor brick and the control structure to be a plug and play capability.

However, the advent of interoperability of control has affected the technical architecture design of the system. The Joint Architecture for Unmanned Systems (JAUS) is a technical specification that is concerned with over-riding or re-establishing communication and control of an intelligent mobile system. Implementation of JAUS into the software structure of the operational architecture addresses the issue of interoperable control and makes the SBS compliant with the JAUS standard.

3.4.5 Visual (Quad Video) Sensor Brick

3.4.5.1 Overview

The *visual brick* provides image data that is used in many operational capacities. The video sensor provides a view of the environment by capturing and transmitting images or image data to a controlling mechanism or computer. Vision for intelligent mobile robotic systems may be necessary for both autonomous and manually controlled operations. Visual sensors can serve the robotic system by providing vision to the mobility and drive systems as well as gathering data for image processing applications. The visual brick serves the mobility and drive system in two (2) important ways: it can provide vision to the robot in motion, and data it acquires can be used for path planning and navigation. Locomotive and navigation operations require clear representations of the environment in order to provide headings to waypoints, plan the trajectory or path of the system, and to observe the surrounding environment.

3.4.5.2 Acquisition, Pre-Processing & Communication

The visual brick in the SBS uses a quad-dome camera that can provide a 360^0 field-of-view and zooming capability for each of its four lenses. The *Clover Electronics USA DQ205 Color Dome Quad Camera* is the camera used as the acquisition block in the visual brick. It is a dome-type camera with four (4) cameras and a quad video splitter which allows simultaneous viewing of images acquired from all four of the cameras.

Each camera can be positioned to view specific quadrants of interest. All four (4) cameras can provide 360^0 of hemispheric coverage.

The visual sensor brick uses the quad camera to *acquire* data which outputs the data in analog format. Further processing and analysis requires the raw data to be converted into digital form. The *pre-processing* block of the visual sensor brick consists of a computer consisting of motherboard that includes central processing unit, memory (RAM), input and output capabilities and expansion cards. The ASUS P4P800-VMmini-ATX board manufactured by ASUSTeK Computer Inc. was chosen to perform this task on the visual sensor brick. It supports a Pentium 4 processor and has three (3) PCI expansion slots. One of the slots is used to interface to the camera through the capture card and another PCI slot is used for communication to the remote central control system. A *frame grabber* capture card takes in the analog input from the camera to produce digital data that is then sent to the intelligent systems (in the next phase of the preprocessing block) for further processing operations. The main function of the pre-processing block is to perform low-level image processing on the acquired data and including inversion, resizing, cropping, edge detection operations, etc. For the visual sensor brick described in this report, the *Pinnacle Studio AV/DV Version 8 Capture Card* is used as the capture card.

The pre-processed data may be used directly by the visual sensor brick or it can be passed to the *communication block* for wireless transfer to the operator interface (or any other intelligent agent that may require use of the data). The communication block of the visual sensor brick transmits the data that has been captured and preprocessed. The communication card of is a Linksys Wireless-G PCI WMP54G Card. This card supports the wireless LAN IEE 802.11g standard, setting the baud rate of the sensor brick to 54MBps (with 128-bit encryption) [20].

For a more detailed description of the visual sensor brick (Figure 28) see the IRIS Laboratory internal report, Video Sensor Brick for Modular Robotics, by Anjana Poduri.

3.4.6 Thermal (Infrared-Imaging) Sensor Brick

3.4.6.1 Overview

The *thermal sensor brick* detects objects that are outside the range of normal human vision, exploiting an expanded range of information. Figure 29 shows the thermal sensor brick loaded into the payload bay of Safebot. The thermal sensor brick uses electromagnetic radiation from the infrared spectrum to form images. Thermal sources radiate electromagnetic wave energies scintillating the sensor, a microbolometer (similar to a CCD array). The thermal radiation acquired through the lens and aperture system of

Figure 28: The visual (quad-video) sensor brick mounted in Safebot (Developed by Anjana Poduri).

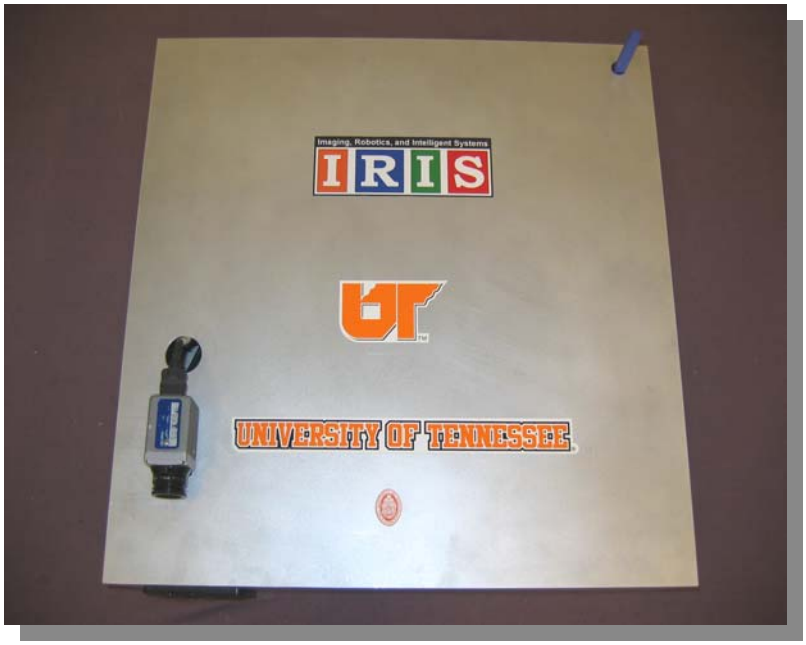

Figure 29: The thermal (infrared-imaging) sensor brick (Developed by Nikhil Naik).

the thermal camera impacts the cells of the bolometer array. The intensity of radiation at each cell produces a voltage that is ultimately transformed to a (grayscale) value.

On of the most prevalent applications of thermal imaging involves the detection of human thermal signatures. Perhaps the most import use of thermal imagery is in search and surveillance operations. Identifying human victims in cold environments, such as being lost in a forest, or victims in cold water drowning accidents, can be located using thermal imagery. Thermal imagery is also used in other areas such as intrusion detection, face and pattern recognition, object tracking, and other biometrically-related areas. Thermal images used for face recognition can guard secured points of access where entry and exit monitoring is important. Intelligent systems face recognition algorithms compare the features or characteristics of one thermal image to compare to other images features or characteristics that are in the database. The access point is restricted or permits access based on the results of the comparison.

Area surveillance is another important use of thermal imagery. Aerial thermal imagery can provide detection through smoke and dark areas using a vehicles thermal signature for traffic reports on a rainy or foggy day. Environmental applications include the detection of oil spills, or biological toxins like red algae (red tide). Automobile ergonomic and safety issues use thermal imagery in the visual detection of approaching motorized vessels. Defense applications of aerial thermal imagery include troop surveillance and detection, terrain mapping, plant coverage, and others.

Thermal imagery is also an important tool is the quality control applications in industrial settings. Manufacturing processes ranging from the production of food, glassware items, cast iron patterns, moulds, and other areas where quality assurance of the product and surveillance on the production line is a necessity ordinarily use infrared sensors as an important element of the sensors array.

3.4.6.2 Acquisition, Pre-Processing & Communication

The thermal sensor brick is equipped with an infrared sensor (the Omega infrared camera) to capture infrared images. The Omega infrared camera is one of the world's smallest and lightest commercially available infrared cameras. This camera was developed in 2002 as part of a joint venture program between Night Vision and Electronic Sensors Directorate NVESD (US Army – Communications, Command, Research and Development Center), and Indigo Systems Corporation of Santa Barbara, California. The Omega belongs to the UL3 family of infrared cameras manufactured by Indigo systems.

The Omega's small size of (3.5 cubic inches), light weight (102 grams), and very low power consumption (< 1.3 W) make it an excellent choice to use in the thermal sensor brick. The infrared sensor is a 164 by 120, un-cooled microbolometer, focal plane array (FPA). It is well suited for security applications, search and remote surveillance

applications, unmanned aerial vehicles applications, weapon sighting and target acquisition, inspection applications for mine fields, and in unattended ground sensors.

The Omega is a long-wavelength thermal camera with sensitivity in the range of 750 nm to 1350 nm. Its small size, light weight and low power consumption are achieved by employing state-of-the-art readout integrated circuit design and innovative electronics packaging concepts

The Omega camera does not have a Thermoelectric Cooler (TEC). This device, which is ordinarily found in most un-cooled cameras, maintains the focal plane array (FPA) at a stable temperature. Otherwise, the FPA output varies non-uniformly, causing undesirable image artifacts. The Omega instead utilizes a completely new technique by combining on-focal-plane circuitry and non-uniformity correction (NUC) processing to eliminate the use of TEC. This helps the camera to operate over a wider temperature range while at the same time maintaining its dynamic range and image uniformity.

These design features are responsible for the low size, weight, and power consumption. The absence of a TEC helps to reduce the complexity of the camera, giving it a higher range of operation. The camera is able to take images immediately after powering the unit. Cameras employing FPA temperature stabilization with a TEC usually have a waiting time before operations commence

The Omega thermal camera *acquires* data and can output the data in either digital or analog format. It delivers the images in a wide dynamic range (14-bit) at real-time video rates (30 fps) for RS 170 – A or (25 fps) for CCIR. An auto-ranging function can find extremely hot scenes, switching into an extended temperature range mode. This allows the camera to acquire images in scenes up to 400° C. The internal shutter continuously recalibrates the camera automatically, or it can be manually overridden to facilitate process-monitoring applications. The analog video output utilizes a feature called 'Smart Scene' which helps to enhance picture quality for all the scenes. This feature uses a dynamic, non-linear conversion to process the 14-bit digital image data into 8-bit data for analog video. The conversion algorithm automatically adjusts, frame by frame, to maximize the contrast in darker (colder) parts of the frame, while trying to avoid "blanking" of brighter (hotter) objects in the image frame. The advantage of this feature is the automatic and continuous optimization of images that is independent of scene dynamics. The camera is also equipped with a proprietary image optimization system that pre-processes the image data, eliminating the need for temperature stabilization of the array. This ultimately allows operations over a wider temperature range

Further processing and analysis requires the raw data to be converted into digital form. A *frame grabber* capture card takes in the analog input and produces digital data that is then sent to the *preprocessing block* for further processing operations. The first version of the thermal sensor brick uses the *IMPERX VCE-PRO Fast Analog CardBus* video capture

card, manufactured by Imperx Inc. This video capture card is necessary for capturing analog video sequences directly from the Omega infrared camera. The video capture card greatly facilitates obtaining live streaming video captured by the camera. The IMPERX VCE-PRO is a PCMCIA based video capture card and is inserted in the PCMCIA slot on the CPU. An S-Video to RCA adapter cable is used to connect the Omega camera and the frame-grabber card. The card comes with its own graphical user interface (GUI) which helps to set and control its operations. The frame grabber card suits the modular design philosophy since it can be used like a plug-and-play device. The VCE-PRO is also capable of being inserted and removed in a "hot" state.

The Omega camera is equipped with an internal conversion algorithm that automatically adjusts, frame by frame, to maximize the contrast in darker (colder) parts of the frame, while trying to avoid 'blanking' of brighter (hotter) objects in the image frame. The advantage of this feature is that the images are produced independent of scene dynamics due to the automatic, continuous optimization procedures. The camera is also equipped with a proprietary image optimization system that pre-processes the image data, eliminating the need for temperature stabilization of the array which allows operations over a wider temperature range. The thermal sensor brick is a self-sufficient system, making possible the quick removal and exchange of sensor brick(s) from the mobility platform without affecting operational functionality. In this manner, the brick exists as a stand-alone system allowing the acquisition and transmission of infrared data [18].

For a more detailed description of the thermal sensor brick, see the IRIS Laboratory internal report, Thermal Infrared Sensor Brick for Modular Robotics, by Nikhil Naik.

3.4.7 Laser Range Sensor

3.4.7.1 Overview

The laser range sensor is a special type of sensor, which uses laser light for determining the shape and geometry of an object or the environment. The SICK laser range scanner is shown mounted in Safebot in Figure 30. The scanner is a non-contact measurement unit that scans the surroundings in two-dimensions. As a scanning system, the device requires neither reflectors nor position marks. Unlike normal cameras, which capture reflected light from an object, the laser range sensor measures the distance of the object being scanned from the reference point on the sensor. Each pixel in a range image is an intensity value that represents the range or distance to an object.

Range is defined as the distance from a known reference co-ordinate to a point in the scene being examined. A sensor that detects the reflected laser light provides a range image that gives direct three-dimensional information about a scene that is unattainable by normal imagery. The laser range sensor has many potential applications. Some areas

Figure 30: The laser range sensor brick placed on Safebot (Developed by Santosh Katwal).

of application include area monitoring, object detection and measurement, and determining positions. The laser range sensor brick is also well-suited for the under-vehicle inspection application. The problems associated with under-vehicle scanning (low illumination and lack of access to the under-vehicle's center region and wheel wells) are easily overcome by the laser range scanner. The system design, which utilizes intelligent software applications, allows the scanner to be placed (plug-and-play) in a low-level tracked robot like Safebot, and maneuvered under the vehicle to take scans of the scene. These scans can be displayed in real-time on the screen to provide additional information in the inspection for the presence of possible threat objects under the vehicle. The Laser Range Sensor Brick's role in the under-vehicle inspection scenario is explicitly detailed in a subsequent chapter.

3.4.7.2 Acquisition, Pre-Processing & Communication

The laser range sensor brick is comprised of a laser scanner (the LMS 200 manufactured by the SICK Inc.) for *acquisition*, a micro ATX board (ASUS P4GE-VM) for *pre-processing* the acquired range data, a wireless network card (Linksys wireless-G) for *communication* with the central control unit or remote server, and a *power supply* block that consists of a PW-200-V DC-to-DC converter for the motherboard, a VI-LJ03-CY DC-to-DC converter for the SICK laser range scanner, and a 12 V Panasonic LC-RA1212P battery.

The LMS 200 is a non-contact Laser Measurement System that works under the principle of pulsed time-of-flight [Sic02]. It is an active scanning system that scans the surroundings and objects two dimensionally without the need of passive components such as reflectors or position markers. The pulsed laser beam is deflected by an internal rotating mirror so that a fan shaped scan is made of the surrounding area and the shape of the object is determined by the sequence of impulses received.

The scanner provides a distance value every 0.25°, 0.5° or 1° per individual impulse, depending on angular resolution of the scanner. As a result of the beam geometry and the diameter of the individual spots, the spots overlap on the target object up to a certain distance.

The scanner has a maximum range of 80 meters, but is ideally suited for a 60 meter distance. It has a variable data transfer rate with four options: 9.6, 19.2, 38.4 and 500 K baud. It can be connected with the processing board via a RS 422 serial interface card. The real-time measurement data is scanned by the device and outputted in binary format via the serial interface.

The LMS 200 Sick sensor operates under the principle of pulsed time-of-flight. The time taken by a pulse of laser beam to travel from the scanner to the target point and back is measured to determine the range of the target point from the source. A coded laser beam pulse emanates from a semiconductor laser diode, which strikes the object's surface, reflects back, and registers on the scanner's receiver diode. The time-measurement unit measures the time taken by the pulsed beam to get from the transmitter diode to the receiver diode. The processor calculates the resulting distance or "range" values. The corresponding digital values of the range are provided as the output.

The Sick scanner data can be used for object measurement and determining position. The measurement data correspond to the surrounding field-of-view contour scanned by the device. Since the data output is ordered sequentially, the angular displacement is known for each range value. The GUI is able to provide the co-ordinates of distance and angular displacement for every sample-point in the field-of-view. Some of the applications where this sensor can be used are area monitoring, object measurement and detection, and position determination position. With this sensor, measurement data is available in real-time and can be further processed if needed. The device scans at a very fast rate allowing the measured object to move at relatively high speeds. Although the LMS scanner can be used outdoors, it is best-suited for indoor operations.

The laser range sensor brick acquires data and like the other bricks, sends the data in either raw format or preprocessed form through the communication block to a central server or remote host. The intelligent systems use the information provided by the laser range scanner to make appropriate decisions with respect to the heading and path of the

mobility platform. Other intelligent systems use the application of laser range scanners to visualize objects in three (3) dimensions. This type of visualization is a critical factor in the field of reverse engineering and three-dimensional (3D) computer vision.

After acquiring the data, the preprocessing block takes the data and starts its own operation. The range data acquired by the acquisition block is in binary format. The data must be manipulated to obtain the range profile and then the corresponding range image. The main purpose of this block is to carry out preliminary or low-level processing that is needed on the raw range image. Preliminary processing involves filtering for noise removal, rotation of the image, smoothing, and sharpening, zooming, cropping and edge detection. At times, the block might need to perform higher-level processing tasks such as matching profiles, registration and three-dimensional (3D) scene reconstruction. Most of the higher-level processing is done remotely through the central remote host, however, the sensor's preprocessing block, by design, must be capable of performing these operations. This ensures maintaining the modular aspect of the overall system.

Generally, this block is intended to be used for processing, storing and transmitting the acquired images. The motherboard selected for use in the brick is the ASUS P4GE-VM micro ATX board, manufactured by ASUSTeK Computer Inc. It supports a Pentium IV processor and has three (3) PCI expansion slots. One slot is used for placing the RS-422 interface card that connects the scanner with the board. Another slot is used for the communication interface with the remote host. It has two 184 pin DIMM sockets for additional RAM [19].

For a more detailed description of the laser range sensor brick, see the IRIS Laboratory internal report, Laser Range Sensor Brick for Modular Robotics, by Santosh Katwal.

3.4.8 Radiological Sensor

The chief component of the *radiological sensor brick* is the Nucsafe Gamma and Neutron Detection System, built by Nucsafe Inc., Oak Ridge, TN. The sensor detects the presence of both gamma radiation (at six different energy levels) as well as neutron deterioration. The sensor utilizes a state-of-the-art integrated electronic systems package that includes the acquisition unit (the scintillation system), the pre-processing and calibration mechanisms, the communications system, and the power supply system. The nature of the advanced electronics and scintillation systems require that the components included within the housing of the detection unit be fabricated and assembled by the unit's manufacturing company. In this sense, the radiological sensor acts as a self-contained unit and meets the definition of a modular block for the purposes of inclusion into the SBS.

The radiological sensor detects gamma or neutron deterioration and sends an alarm notifying the operator. When the detector becomes operational, there is a short period that the radiological sensor uses to calibrate its detection system by detecting the local

background radiation. An alarm is sent when a gamma source or deteriorating neutrons are detected above a level that is prescribed by the initial background radiation check. Figure 31 shows the radiological sensor brick and its control GUI. The radiological brick is used in two important ways. The present state of the sensor brick sends an auditory and visual alarm through the GUI when detection has occurred. A future development will utilize the intelligent systems to use data from the radiological sensor brick as a localization tool that will be used to identify the source and pinpoint the placement of radiologically contaminated areas.

Figure 31: The radiological sensor brick and its control GUI.

4 THE REALTIME CONTROL SYSTEM

By implementing the *Realtime Control System* (RCS) as the operational architecture, the system can evolve into a more autonomously functioning system by lessening the dependence on human interaction at current levels of operation. This does not imply that a human can not intervene; it can show that human oversight at some lower levels of operation can be lessened so that the human resource may be otherwise allocated.

4.1 Introduction to the Realtime Control System

The Realtime Control System (RCS) is a hybrid, hierarchical reference architecture developed at the United States Department of Commerce, National Institute of Standards and Technology (NIST), by James Albus et al. RCS provides a methodology that conceptualizes, designs, engineers, integrates, and tests intelligent software for vehicle systems with any degree of autonomy. It has been selected to be used as the *operational architecture* for the SBS.

The Realtime Control System (RCS) is a hybrid architecture that combines deliberative with reactive components. It provides a *reference model* for unmanned intelligent autonomous vehicle systems on how their software systems should be identified and organized. It defines ways of interacting that ensure missions can be analyzed, composed, distributed, planned and executed intelligently, effectively, efficiently, and in coordination. RCS is especially pertinent to those applications involving unknown and possibly hostile environments.

To achieve this, RCS provides or defines the sensory processing, world model, knowledge management, cost-benefit analysis, behavior generation, and messaging functions as well as the associated interfaces that are necessary for applying a *computational model of intelligence*. RCS is consistent with military hierarchical command structure and doctrine and provides a methodology for conceptualizing, designing, engineering, integrating, and testing intelligent systems software for vehicle systems with any degree of autonomy, from manually operated to fully autonomous.

4.2 Overview of RCS

RCS is a *reference model architecture*; it is characterized by a generic *control node* that is applied to all the *hierarchical control levels*. RCS integrates the function elements knowledge representations, and flows of information so that intelligent systems can

analyze the past, perceive the present and plan for the future. The functional elements are organized into a *RCS Node*. These functional elements are:

- Behavior Generation (BG),
- World Modeling (WM),
- Sensory Processing (SP),
- Value Judgment (VJ), and
- Knowledge Database (KD).

Each RCS node (Figure 32) looks upward to a higher-level node from which it takes commands, for which it processes sensory information, and to which it reports status. Each RCS Node also looks downward to one or more lower-level nodes to which it issues commands and from which it accepts sensory information and status. Each RCS Node may also communicate with peer nodes with which it exchanges information.

RCS Nodes contain behavior generating (BG), world modeling (WM), sensory processing (SP), value judgment (VJ) processes, and a knowledge database (KD). Any or all of the processes within a node may communicate with an operator interface. Within a RCS Node, interconnections between SP, WM, and BG close a reactive feedback control loop between sensory measurements and commanded actions. The interconnections between BG, WM and VJ enable deliberative planning and reasoning about the costs and benefits of future actions. The interconnection between SP, WM, and VJ enable knowledge acquisition, situation evaluation, and learning. Predicted input generated by the WM from information in the KD is compared in VJ with observed input from sensors or lower-level nodes. The difference between observations and prediction can be used to update the WM and implement a recursive estimation process such as Kalman filtering. Information in the world model KD of each node can be exchanged with peer nodes for purposes of synchronization and information sharing [1].

A RCS Node is similar to a *RCS Module* in that both have a set of input buffers, a set of output buffers, and a set of input and output message formats. This is illustrated in Figure 33. Message formats include command frames, status frames, sensory inputs and outputs, peer-to-peer communications, and an operator interface. At any point in time, the set of input buffers define an input vector and the set of output buffers define an output vector.

A RCS Node is different from a RCS Module in that a RCS Node may be constructed from one or more RCS Modules. The RCS Modules within RCS Nodes are independent and asynchronous except for coordination through flags or semaphores passed in NML (note: Neutral Message Language is the computer software system language developed at NIST by Albus et al. and is used to integrate and implement RCS into a robotic system. E.g., it acts as a shell that is placed into a system so that the hierarchical command structure may utilize intelligent systems in the application of the system.). NML moves information between asynchronous RCS Modules by providing mailboxes into which

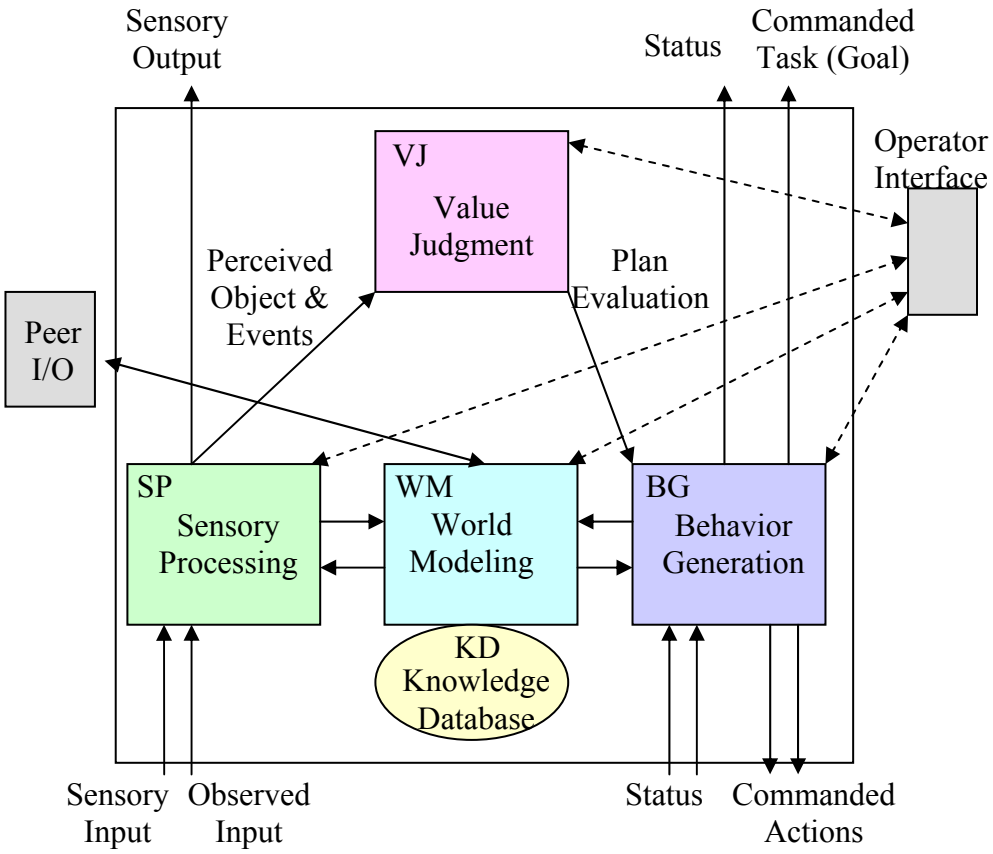

Figure 32: The basic building block, a RCS Node [1].

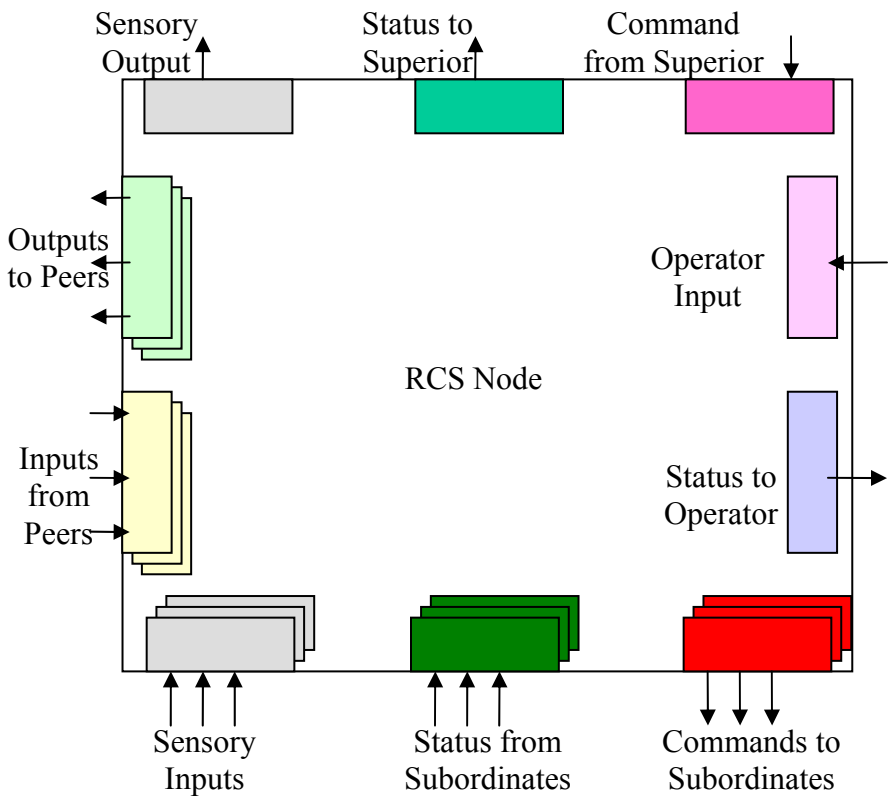

Figure 33: RCS Computation node showing input and output buffers [1].

messages are posted at times convenient to writers and read at times convenient to readers.

A RCS Module always runs on a single CPU and will often share that CPU with several other RCS Modules. In contrast, a RCS Node may be distributed over several CPUs. The number of RCS Modules required to implement a RCS Node varies depending on the application. In simple cases, it may be possible to implement an entire RCS Node within a single RCS Module. For example, simple sensory processing and world modeling functions might be embedded in the preprocess phase of a RCS Module. Behavior generation and value judgment might both be embedded in the decision process of a single RCS Module. In more complex applications, a number of RCS Modules will be required to implement the functionality of each RCS Node [1].

A collection of RCS computation nodes, illustrated in Figures 32 and 33, can be used to construct a distributed hierarchical reference model architecture shown in Figure 34.

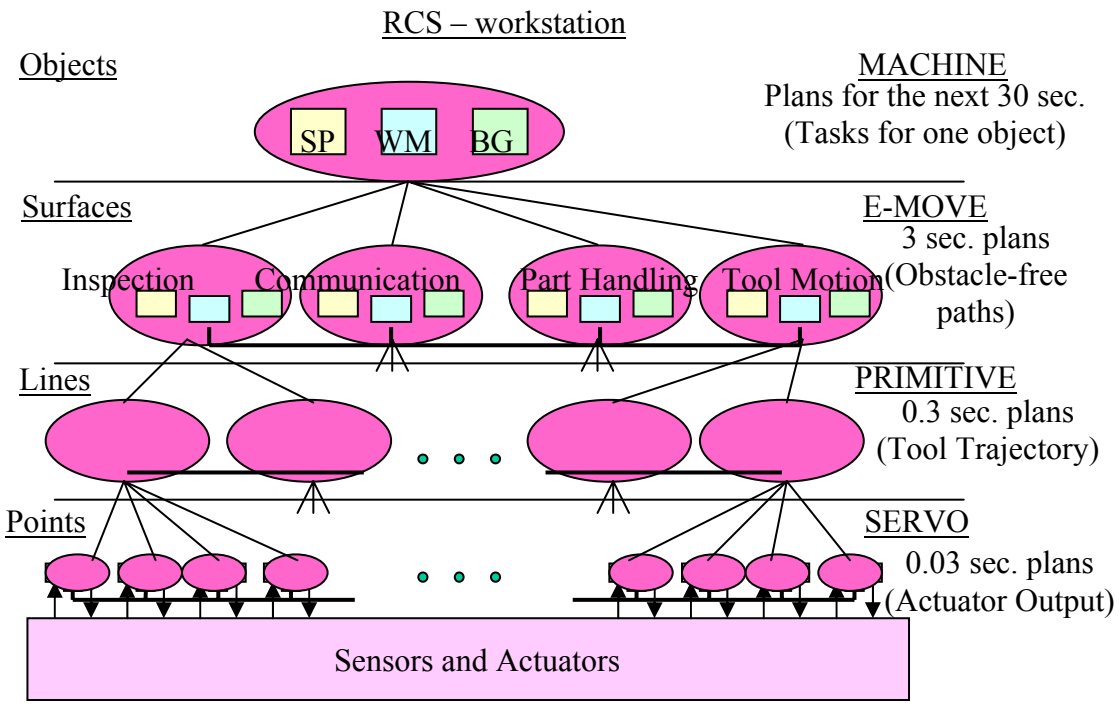

Figure 34: RCS Workstation Example [1].

Although this particular architecture is for a machine tool in a manufacturing environment, a similar architecture could be developed for an autonomous land vehicle, a construction machine on a construction site, an undersea vehicle carrying out cooperative operations with other undersea vehicle systems, and many other applications [1].

Each RCS Node in the architecture (in Figure 34, each RCS Node is represented as an oval, with the SP, WM and BG processes as sub-entities inside) acts as an operational unit in an intelligent system. Depending on where a particular RCS Node resides in the hierarchy, it might serve as a controller for one or more actuators, a subsystem, an individual machine, a group of machines comprising a manufacturing workstation, a group of workstations comprising a manufacturing cell, or a group of cells comprising a manufacturing shop. The functionality of each RCS Node (or RCS Module within a RCS Node) can be implemented by a set of software processes or by a person or group of persons [1].

4.3 RCS Functionality

Functionally, RCS is governed by a set of rules that applies the nodes as universal controllers to each *hierarchical level*. Controllers are assigned specific capability based on the mission, and activity analysis and modeling. A standard interfacing mechanism is used across the architecture. RCS also applies a task decomposition process that facilitates both the configuration and the execution of an operational architecture. The reference model nature makes RCS applicable to many intelligent mobile robotic systems [1].

4.3.1 Summary of Functionality

The RCS architecture provides a reference model for unmanned intelligent autonomous vehicle systems on how their software systems should be identified and organized. It defines ways of interacting that ensures missions, especially those involving unknown and possibly hostile environments, can be analyzed, composed, distributed and planned and executed intelligently, effectively, efficiently and in coordination. To achieve this, RCS provides the sensory processing, world model, knowledge management, cost-benefit analysis, behavior generation, and messaging functions as well as the associated interfaces. RCS is consistent with military hierarchical command structure and doctrine. RCS provides a methodology for conceptualizing, designing, engineering, integrating, and testing intelligent systems software for vehicle systems with any degree of autonomy, from manually operated to fully autonomous. The theoretical foundation for this methodology is the RCS reference model architecture [1].

The RCS reference model architecture is encapsulated in the following properties:

1. It defines the functional elements, subsystems, interfaces, entities, relationships, and information units involved in intelligent vehicle systems.
2. It supports the selection of goals, the establishment of priorities and rules of engagement, the generation of plans, the decomposition of tasks, and the scheduling of activities; and it provides for feedback to be incorporated into control processes so that both deliberative and reactive behaviors can be combined into a single integrated system.
3. It supports the processing of signals from sensors into knowledge of situations and relationships; and it provides for the storage of knowledge in representational forms that can support reasoning, decision-making, and intelligent control.
4. It provides both static (typically for long-term) and dynamic (typically for short-term) means for representing the richness and abundance of knowledge necessary to describe the environment and state of a battlefield and the intelligent vehicle systems operating within it.
5. It supports the transformation of information from sensor signals into symbolic and iconic representations of objects, events, and situations, including semantic,

pragmatic, and causal relationships; and it supports transformations from iconic (pictorial) to descriptive (symbolic) forms, and vice versa.

6. It supports the acquisition (or learning) of new information and the integration and consolidation of newly acquired knowledge into long-term memory.
7. It provides for the representation of values, the computation of costs and benefits, the assessment of uncertainty and risk, the evaluation of plans and behavioral results, and the optimization of control laws.

5 A COMPARISON OF SBS AND RCS

5.1 Overview

At the present, the SBS is composed of four (4) prototype sensor bricks (visual, thermal, laser range and radiological), the mobility platforms, and the intelligent systems applications that are entirely developed within the IRIS Laboratory. The use of the intelligent systems enables the intelligent mobile robot system to be applied to a task with some degree of autonomy. In the SBS, as these systems move from the initial stages of development and begin to become a *system of systems*, implementing the appropriate *operational* and *technical* control structures (architectures) are necessary to coordinate the continuing evolution and of development of the SBS.

From the viewpoint of the system architecture the modular design of the SBS provides unlimited possibilities with respect to the *types of mechanisms* that can be realized. However, the restriction on the *types of use* that a system can be applied to is strongly related to how the intelligence systems structure is *organized and arranged*. In the SBS (and other intelligent mobile robotic systems) the intelligent systems dictate the command, control and communication to all levels of the operational hierarchy, also known as the *command and control structure*. In essence, the intelligent systems act as the brain of the robotic unit [1].

The *hierarchical* command and control structure of the SBS in the *under vehicle inspection scenario* consists of intelligent agents (humans) at the highest level of the control structure. The human operators monitor the inspection process, compare the images or data from the various sensor modalities, and gather information to make future decisions with respect to the system's state (what sensors to use next, what should be the next sensor, the order of use of the sensors, when to stop the process, etc). The SBS can demonstrate periods of autonomous operation, but as the system begins to mature, it will need to incorporate more autonomy. The issue of organizational control must address the types of issues that will continue to arise through the process continuum; chiefly, this is a question of how to organize the various entities of the software system that provide the structure for control of the system.

5.2 Purpose and Methodology

Implementing the Realtime Control System into the SBS can be shown *comparatively* through a sequence of descriptions or snapshots of the SBS where successive iterations display a more fully developed system than the previous description. This method is used

to develop the concepts of RCS and to illustrate that even with small system modifications or additions, the issue of the system's operational control quickly becomes a contentious issue. Beginning with the present state of the SBS and concluding with a plausible future state, the example illustrates the importance of operational control structures in the continued development of a system as it moves toward autonomous operation.

5.3 Developmental Stages

In the initial stage of development, the SBS consists only of the Safebot, and the Laser Range Sensor Block. The only purpose of the system at this stage of development is to *look for and identify the perimeter* of a vehicle so that it can *plan* to sweep the vehicle underside. In this state, the SBS does not carry any additional sensors (such as the Visual Sensor Block).

The system at this state of development is realized by the RCS command hierarchy depicted in Figure 35. It compares the SBS in terms of four (4) levels to the RCS command hierarchy. At the fourth and highest level (Vehicle Level) the only mission objective requires the brick to successfully cycle through its sweep pattern of the vehicle underside (starting point, sweep, and return to starting point). At the next lowest level, the Subsystem Level, the intelligent systems that command and coordinate the various intelligent activities have only one (1) *node* to command and coordinate at a subordinate level (the Primitive Level Driver). Since the Primitive Level Driver Node has no peers (no other nodes at the Primitive Level under the Intelligent Systems subsystem), this effectively collapses the hierarchy into three (3) levels.

This is comparable with the depiction of the system shown in Figure 36. The right-hand side of Figure 37 shows the planning horizons for each respective level that might be expected by implementing RCS into the SBS (at this stage of development) for an under vehicle inspection scenario.

Each level has a planning horizon and tells how often that particular level re-plans. At higher levels of the command hierarchy, the planning horizon is longer, and the task concepts more abstract. Note there are no plans for any levels above the Vehicle Level for this version of the SBS. The re-planning horizon at the Vehicle Level extends about one (1) minute into the future. This means that, conceivably, the system could be re-tasked every minute. This re-tasking would merely consist of repetitions of the same start-sweep-return to finish exercise as before.

5.3.1 The Second Stage of Development

The next evolution of the SBS involves mounting at least one (1) additional sensor on the Safebot so that it can navigate autonomously as well as gather data from the vehicle underside (Safebot + Laser Range Block + Visual Block). This increase in the level of

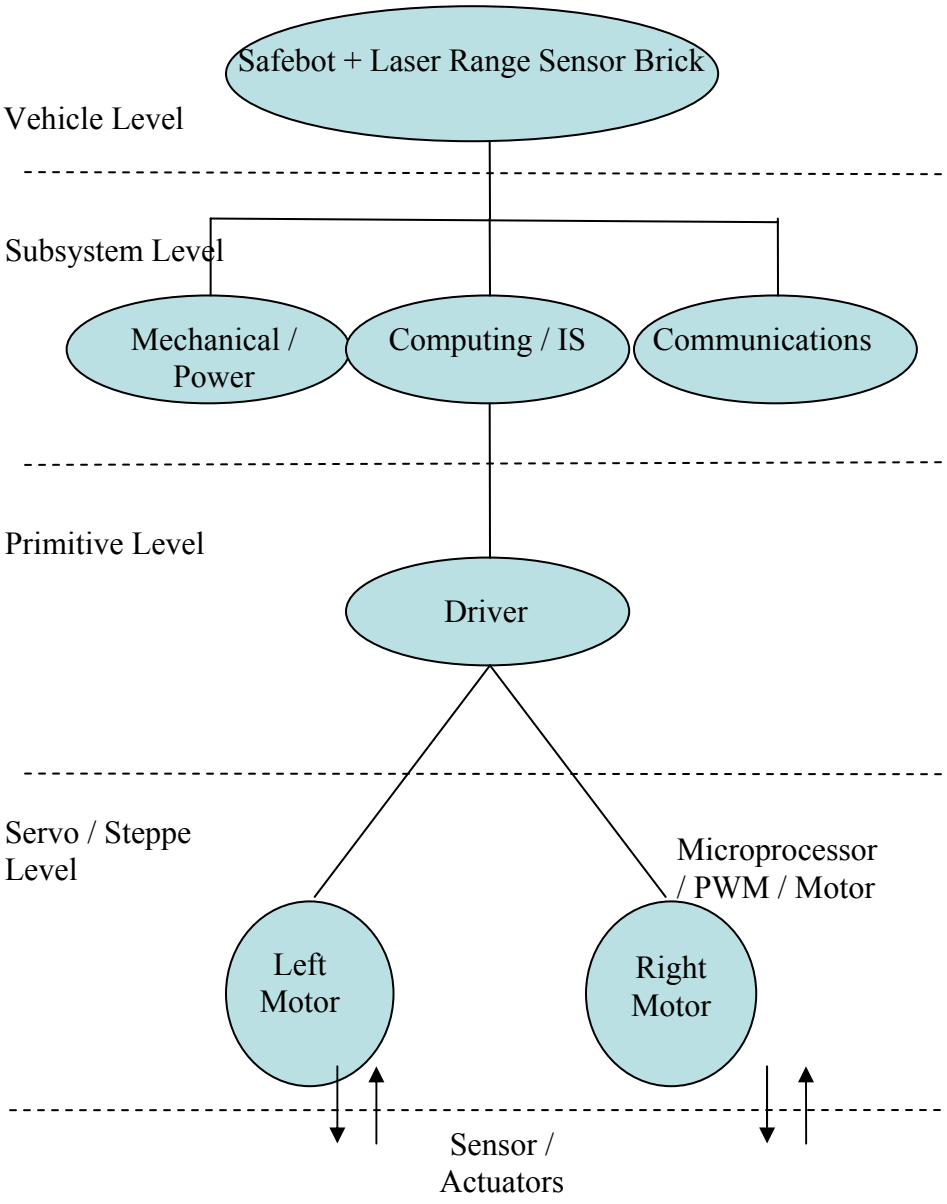

Figure 35: A comparison of the modular SBS with the initial implementation of RCS operational architecture.

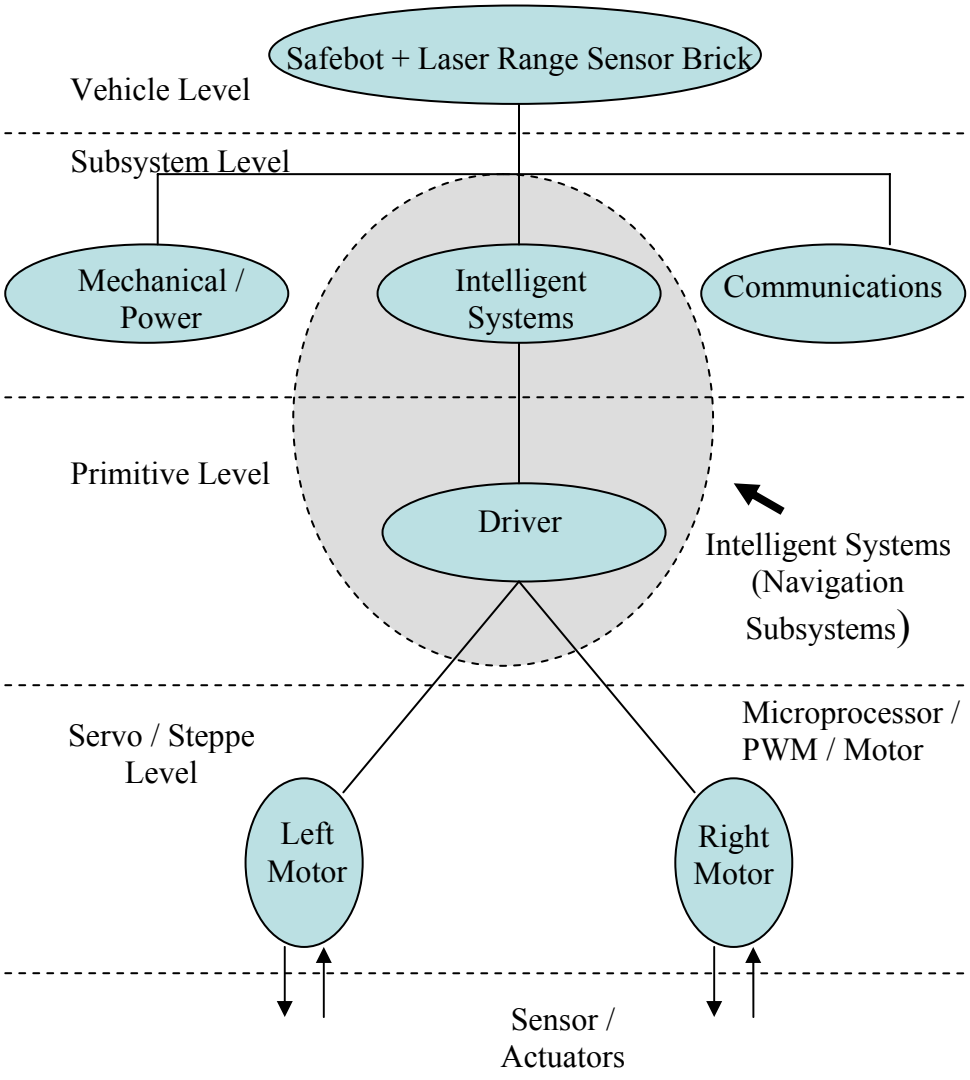

Figure 36: Ordering the SBS into three (3) hierarchical layers.

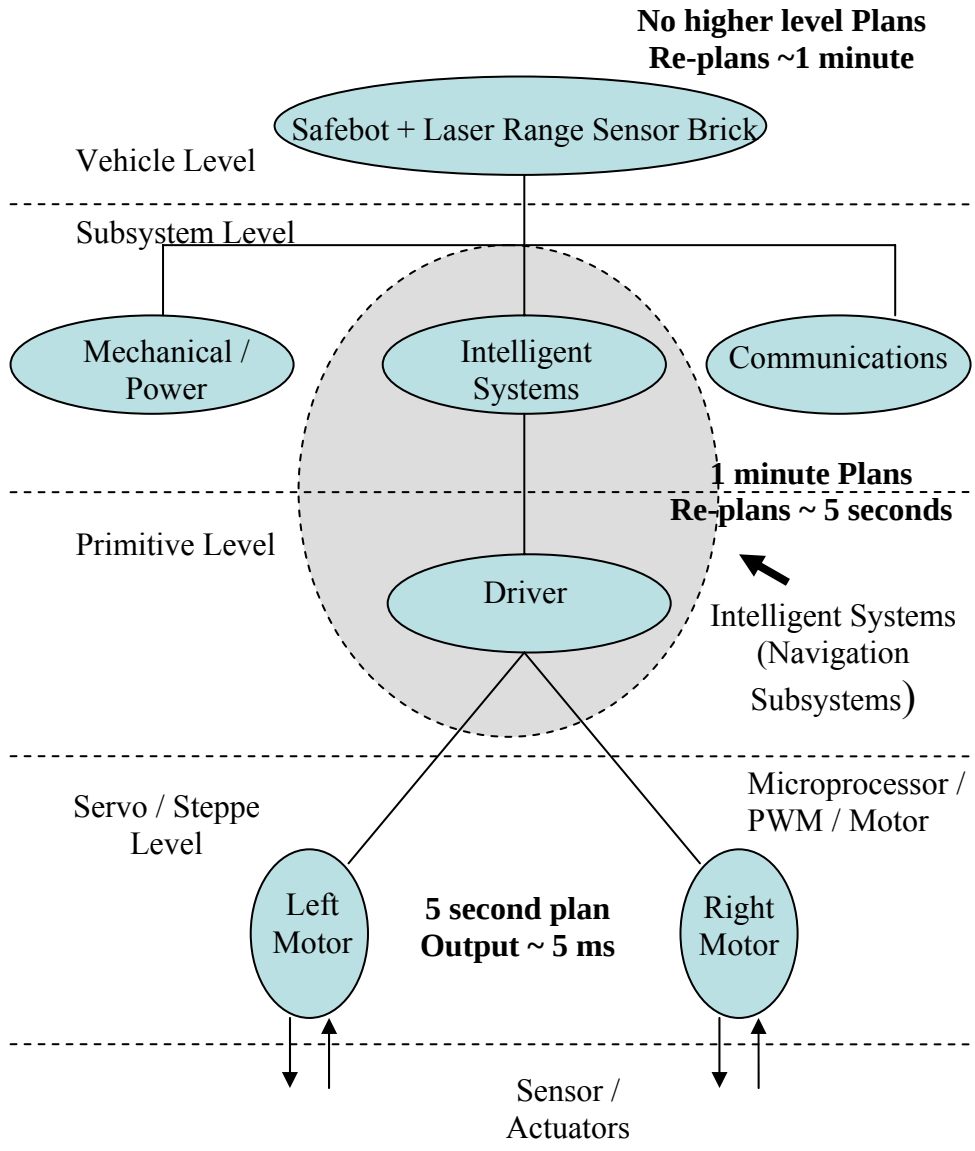

Figure 37: Planning horizons for the three (3) level hierarchy of the SBS.

sophistication, shown in Figure 38, forces the SBS's intelligent control applications to compare two forks of operation with the RCS architecture. One branch navigates the mobility platform while the other branch manages the acquisition sensor activities. The additional sensor provides acquired data that can be used at any level, and possibly at higher, more abstract levels. Since the data being gathered can infer information in the form of possible threat objects, this implies that the top level of the command hierarchy now resides above the vehicle level. In other words, data collected at the vehicle level can be used as information at a higher level (Section Level). This form of the SBS requires an expanded fit from the RCS framework. But the RCS and SBS are similarly designed (modularly) and organized so that this expansion is easy to achieve. At the top level of abstraction, an intelligent agent (at the Vehicle Level) can now receive data sets from an acquisition sensor (similar to the first stage of the under vehicle inspection sweep). This implies that higher, Section Level plans can be formulated.

At this second stage of the SBS's development, the system has gained a capability through the addition of hardware. This addition results in data that may be used to make *decisions* concerning not only the individual vehicle, but also other levels of the system. *Additional information has a wide reaching effect. It can be at used at either higher, lower, or the same levels of the command hierarchy, and may encompass any range of levels up to and including the entire system*. In summary, if a threat object is verified, the intelligent agent now is *aware* of its presence and can use that information for advantage.

5.3.2 The Third Stage of Development

After sweeping the vehicle underside (Safebot + Laser Range Block + Visual Block), the visual sensor acquires data that conveys information regarding possible threat objects. As recounted in the scenario above, once an anomalous object is recognized, the information can be used to begin planning with respect to verifying the alleged threat object as well as notifying superior levels of the command hierarchy of an impending event. If the system is expanded further (Figure 39) to include other sensor types and mobility platforms then more robust plans can be formulated. In the under vehicle inspection scenario, the next logical step requires including additional sensor units so that any possible threat object can be verified. By including other sensor blocks, the complete under vehicle sequence detailed in Chapter 6 can be executed. This level of sophistication does not affect the RCS organization in appearance. However, the underlying planning mechanisms that exist at the vehicle level must be restructured based on the conditional logic of the under vehicle inspection protocols.

The Vehicle Level must now be provided with plans that detail the sequence of sensor blocks, the mounting and dismounting of the sensor blocks, the power management of the sensor blocks, and the intelligent systems that are associated with the sequential application of the sensors. At this level of sophistication, the intelligent systems now control two Primitive Level entities; the Driver Primitive for navigation, and the Gaze Primitive for the acquisition sensor. Since the Primitive Level has two peers, the

Future RCS implementation (under-vehicle inspection)

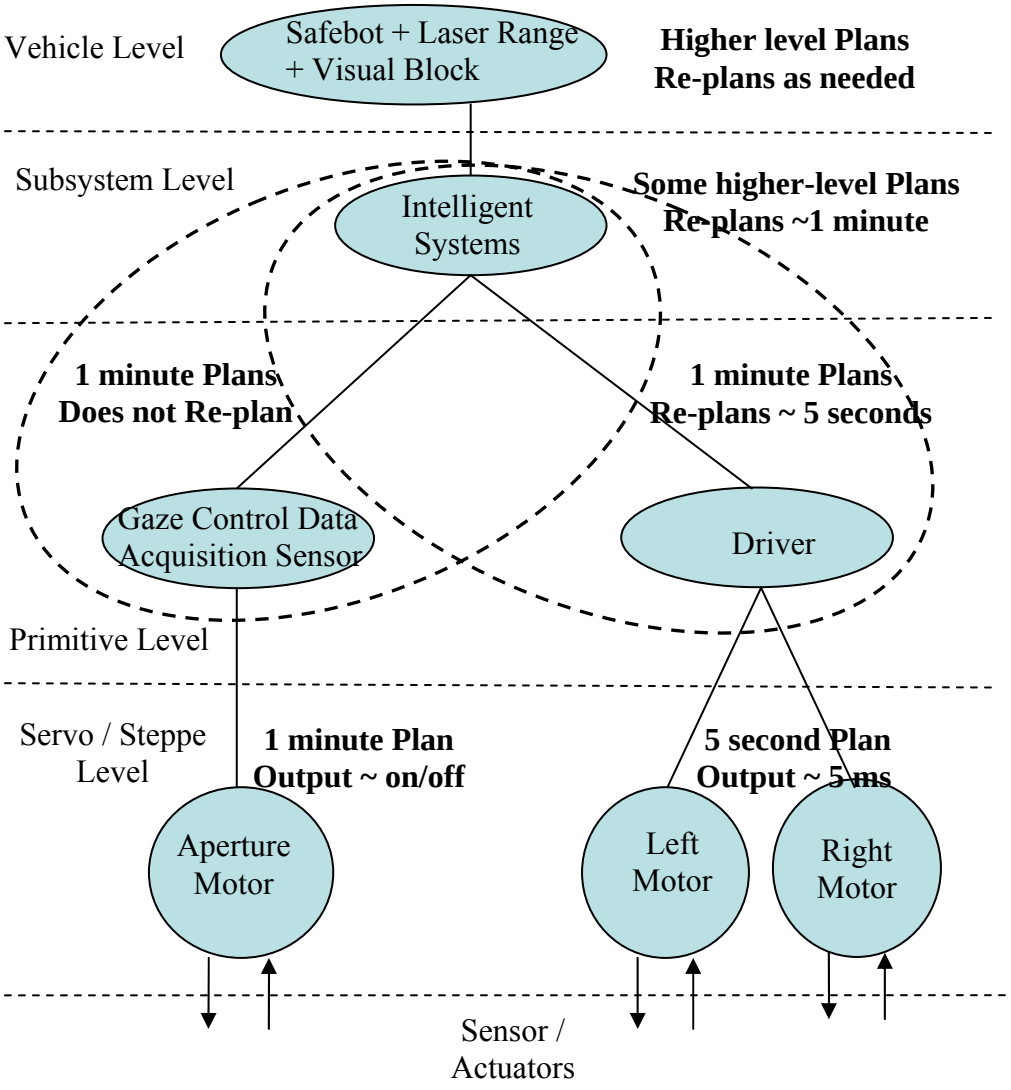

Figure 38: Comparing the SBS to four (4) levels of abstraction in RCS.

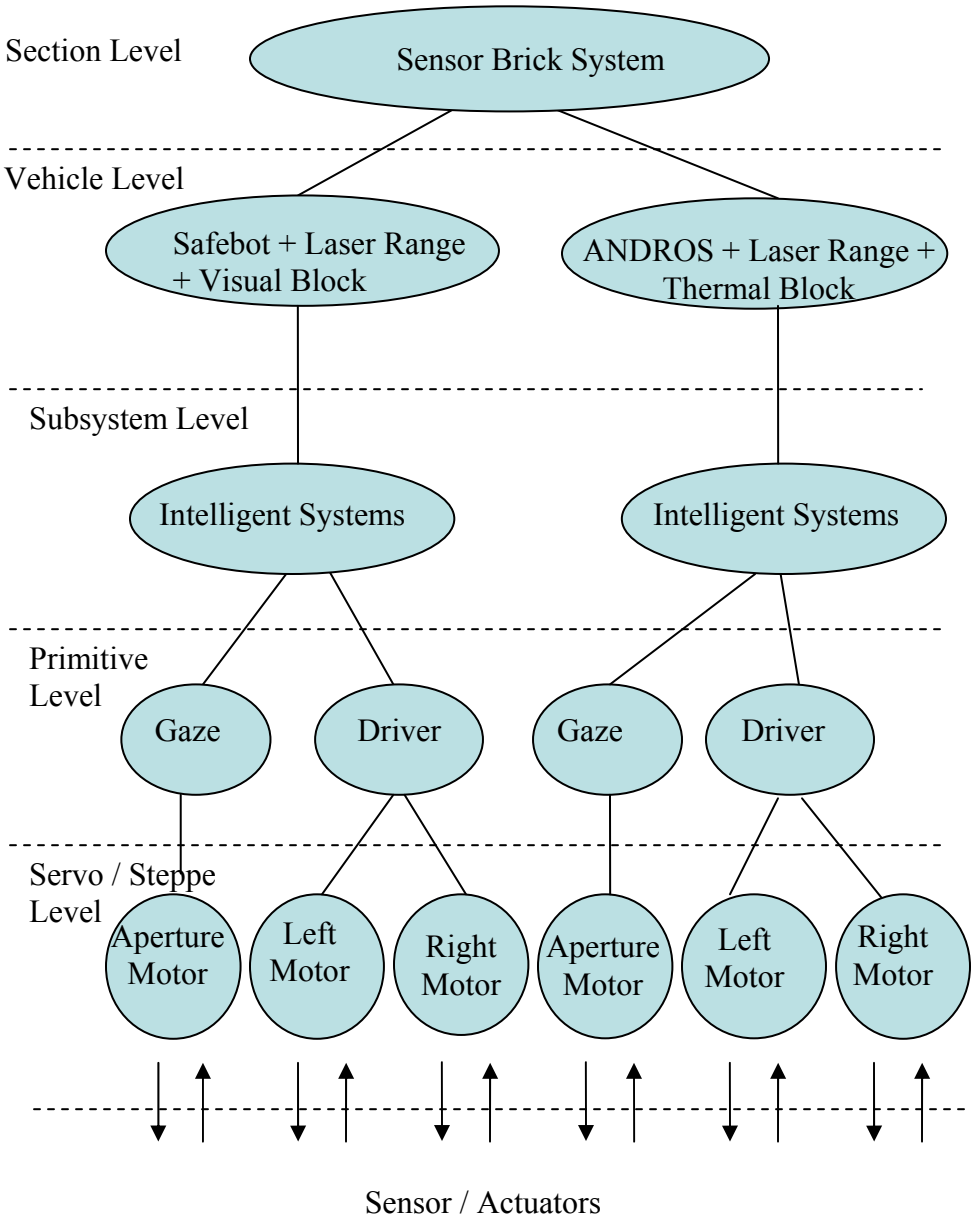

Figure 39: The third stage of development.

coordination effort has necessarily increased for the intelligent systems subsystem and the resulting peer-to-peer communications that will be necessary. This re-organization is readily identified due to the strong consistency that exists between the compatible SBS and its control structure, RCS.

6 EXPERIMENTATION WITH THE SENSOR BRICK SYSTEM

6.1 Introduction

Mobile robotic systems with multiple sensors have many general uses including security monitoring and hazard detection. In this respect, robotic systems are especially useful for inspection of areas where humans cannot or should not enter. This chapter provides a detailed description of one type of application in which the SBS is used to perform an *under-vehicle inspection*. The sensor bricks serve primarily as data acquisition systems. As each sensor brick acquires data from a real-time environment, it either preprocesses and transmits the data wirelessly to an intelligent agent (a human interface, or intelligent software) which acts as one of the command and control points for the robotic system, or it uses the information directly, as in the case of intelligent navigation systems that use laser range data.

To demonstrate an application of the SBS, a single mobility brick (Safebot) transports a progression of sensor bricks as they sweep through a vehicle underside. This is the second level of development cited in Chapter 5. The SBS uses a progression of sensor bricks to collect, fuse, and upload data on the underside of the vehicle to accomplish the inspection.

6.2 System and Deployment Issues

6.2.1 Environmental Control

The SBS can be used in the application of the under-vehicle inspection scenario. The scenario is a vehicular checkpoint or roadblock where the detection tasks are done by the sensor brick units of the SBS. The environmental requirements for the system to be operational are not restrictive; all that is required is a suitable, flat, planar surface. The surface must be hard or compacted, relatively smooth and devoid of significant debris, gravel, or other obstacles that would impede or noticeably perturb Safebot, the dual-tracked mobility platform. As long as the operational surface is flat, and well-compacted, the system has the sufficiency to be operational.

As an alternative to having the sensor bricks move about the vehicle, the sensor brick-based system could be placed in a pre-cut trough. However, there are two principle objections to this type of use. Placing the sensor brick-based system in a ditch requires an

excavation hole to be constructed and the sensors to be set permanently into the trough. This limits the portability of the system and requires the drivers of the vehicles-to-be-inspected to behave cooperatively. Not only must the drivers comply with instructions from the checkpoint operators with respect to the proper orientation and position, they must then navigate the vehicle over the sensors in a smooth and controlled manner.

6.2.2 Advantages over Other Methods

The SBS provides an excellent alternative to the traditionally manned, mirror-on-a-stick under-vehicle inspection (Figure 40) by utilizing low-profile mobility platforms that conduct inspections with multiple types of sensors (Figure 41). Using the SBS instead of the traditional method increases the chance of detection and abates a prohibitively dangerous situation to the human operator. The main advantages of the Modular Robotic System's under-vehicle inspection method over the traditional manual-inspection method with mirrors are:

- The mirror stick must be manually held under the vehicle and gives as little as 25% coverage;
- Human operators are close to the vehicle, gravely exposed to the threat;
- Poor lighting under the vehicle effects the percentage of the under-side that can be viewed;
- The robotic system does not require the physical presence of an operator and can cover 100% of the vehicle under-side.

If no threat is perceived or anomalous situation is detected, the initial sensor acquisitions (usually the visual camera) may be all that is required. However, if anomalies are present during the first sensor sweep or if other factors or situations warrant it, additional protocols can be enacted in order to characterize the nature of the anomaly [20].

6.3 An Inspection Scenario

In following example, the SBS is engaged in an under-vehicle inspection scenario that demonstrates how the various sensor bricks are implemented as a series of anomalous events unfold. The under-vehicle inspection scenario describes a situation where vehicles are inspected to assure that they contain nothing that could be perceived or conceived as a threat. The inspection of the vehicles under-carriage is accomplished using the Safebot mobility platform with a series of mounted sensors. The system uses data gathered by the various sensors allowing human operators to see, understand, and act before harm can be done by any potential threat action. Although other parts of the vehicle must be inspected by other sensor brick combinations (for example, the trunk and passenger compartments of the vehicle can be inspected with Andibot), the focus is streamlined here to the under-vehicle scene. Passenger identification could be accomplished in conjunction with Segbot, where the sensor bricks might be advantageously mounted to perform visual identification of the passengers.

Figure 40: Traditional under vehicle inspection using a mirror-on-stick.

Figure 41: Under-vehicle inspection using the SBS [18].

6.3.1 Intelligent Navigation Systems

The *intelligent navigation systems* form one branch of the hierarchical command structure. The other branch concerns the *intelligent acquisition and analysis systems* that are used to produce information concerning the under-vehicle. The second operational branch is concerned with the acquisition of sensor data of all modalities including the visual, thermal, laser range, and radiological sensor data. The data streams can then be analyzed through human or automated inspection procedures to increase the probability of threat detection.

The inspection procedure begins as a vehicle encounters a checkpoint. After the vehicle of interest is within the appropriate inspection frame, the autonomous operation begins the inspection process by acquiring the appropriate scene range information using the laser range sensor brick.

The mobility platform uses the range data to move to its initial inspection point by receiving commands through the intelligent systems software for navigation. During transit to the first waypoint, the laser range sensor updates its location by acquiring new data and checking its apparent trajectory with the trajectory it has calculated by the encoded feedback.

At the first waypoint, the laser range sensor finds the relative positions corresponding to the location of the vehicle's wheels. At this initial waypoint, the scan takes place from a point that is slightly underneath the suspect vehicle. The wheel locations (corner-points data) are passed to the *intelligence navigation system* where it computes a *sweep* path that is passed to the mobility and drive actuators.

During the sweep, the laser range sensor provides updated corner-points that are periodically updated to re-localize the brick so that it apparent trajectory may be compared with the actual trajectory.

Ideally, all of the sensors are mounted on the under-vehicle mobility platform so that no sensor changes would be necessary. However, the simultaneous acquisition of all four modalities is not possible at the present stage of development of the SBS, so multiple passes must be performed to gather all the data types. Each sensor unit must be deployed individually after the pass of the previous sensor unit by either replacing the sensor unit in the cargo bay of the mobility platform, or by using multiple sensor bricks (the sensor plus its own mobility platform) in a progression, one after the other.

Whether or not the sensor data extractions occur iteratively or simultaneously is not an important matter; the relevant issue is the maximum extraction of information from the analysis of all four types of the sensory data.

6.3.2 The Visual Sensor Brick

Each sensor's data stream imparts a unique source of information that either a human operator or automated comparison algorithms may use to analyze and detect threats. Even though each sensor provides a critical source of information, using multiple sensors provides different types of data that can improve the amount of information extracted from a scene.

In the first step of the operation, intelligent agents (human operators, intelligent systems, or some decision engine) use the visual sensor brick data to look for obvious anomalies like the extra (false) muffler, denoted by the red star in Figure 42. It is evident from the image that the visual sensor brick is inherently superior to the manual, mirror-on-a-stick inspection method.

At this point, an abnormal event has been detected during the process of under-vehicle inspection. Even though the extra muffler is appended to the vehicle underside (the visual information says that something is not ordinary), it does not determine whether the muffler is a threat potential or if it is merely appended to the underside for some other purpose.

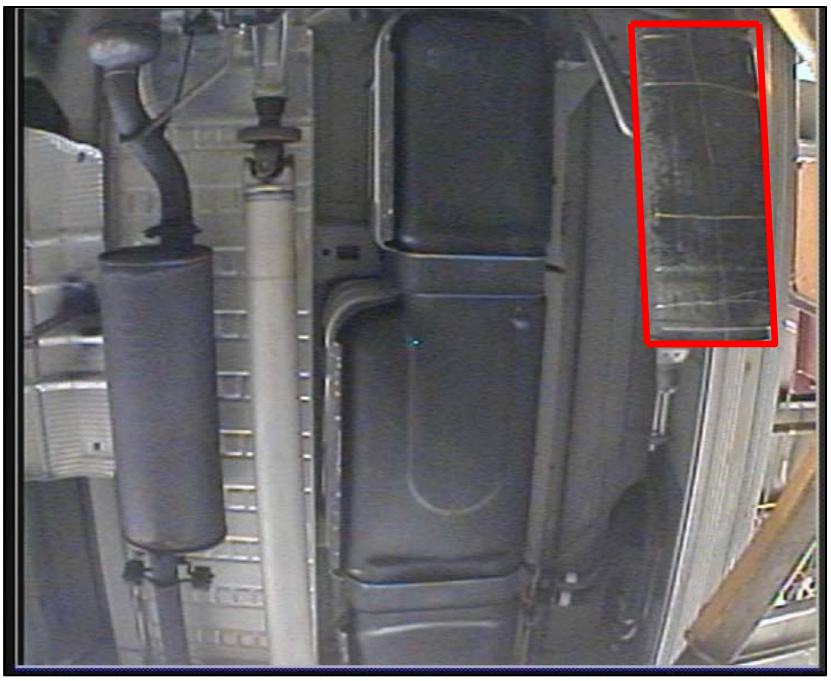

Figure 42: An unusual object detected with the visual sensor brick [18].

6.3.3 The Thermal Sensor Brick

Another data source, provided by the thermal camera, can help to eliminate this type of ambiguity. By using the thermal sensor, the muffler's thermal signature can be read, as well as verifying areas of expected thermal sources. As the sections of the under-vehicle that would normally be hot or cold are verified, they reveal hidden or possible threat objects which may or may not be visible with the visible-spectrum sensor conditions.

These data sets highlight the advantages of using the thermal camera information in conjunction with the visual images. Figure 41 depicts a sequence of images taken from the visual and thermal sensor bricks. The first image sequence shows the vehicle underside in a clear manner. However, unless you are familiar with the structures of a vehicle underside, or using intelligent systems to make a comparison, the foreign object (the dark rectangle in the middle of the images) is not obvious. But in the middle image (Figure 43) the thermal image clearly shows an area with a thermal signature that is out of place. The pseudo-colored thermal image enhances the square object to a greater degree.

6.3.4 The Laser Range Sensor Brick

Thermal cameras can detect threats that may remain invisible if viewed only through a visual-spectrum camera. Figure 43 depicts the visual and thermal images that show a threat object detected by an unexpected thermal pattern. An anomalous image detected by the visual sensor uses the thermal sensor brick to deliberate the status of the object. The laser range brick can be used in a similar capacity to help identify and expose potential threat objects. Under-vehicle scanning performed by the range sensor brick is useful in the same sense as the thermal sensor. The laser range sensor brick can provide additional information that can assist in the detection of a threat object. Consider the image of a visual scan of a muffler (Figure 44). Even though nothing is detected by the visual inspection of the image, the image may be a false placard that merely looks like a muffler. Sweeping the under-vehicle (the muffler) with the laser range sensor enables it to be quickly and effectively identified as having three-dimensional information. However, Figure 45 shows the case of a false muffler that has tried to fool the intelligent systems by using only a flat, poster of the muffler image. The laser range sensor brick shows that this muffler is an anomaly because the range scan returns a *flat* (two dimensional) range image. This alerts the human operator of the threat presence and location of the detected threat object. These results imply that the visual, thermal, and range sensor bricks can all be used to effectively scan the under-vehicle surface to check for threats and other external objects.

6.3.5 Other Enhancement Methods

Other methods that enhance the multiple sensor brick approach include but are not limited to:

- Grayscale Imagery

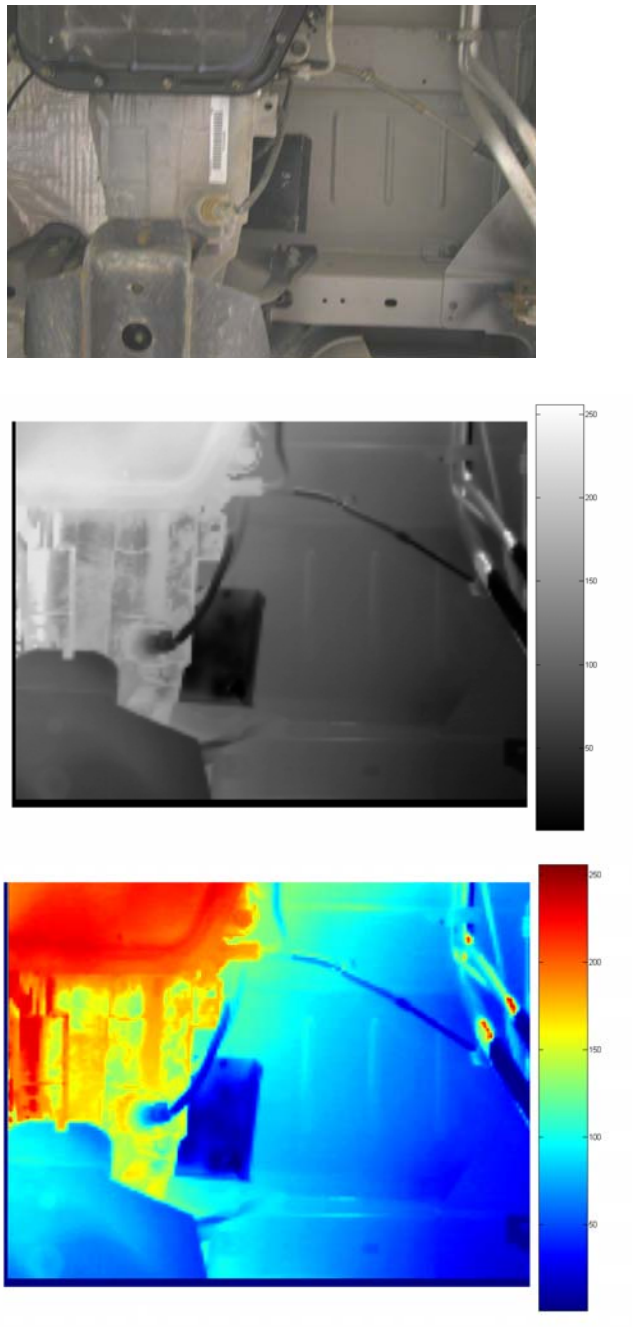

Figure 43: A visual, thermal, and pseudo-colored thermal image sequence of the vehicle underside inspection scenario [18].

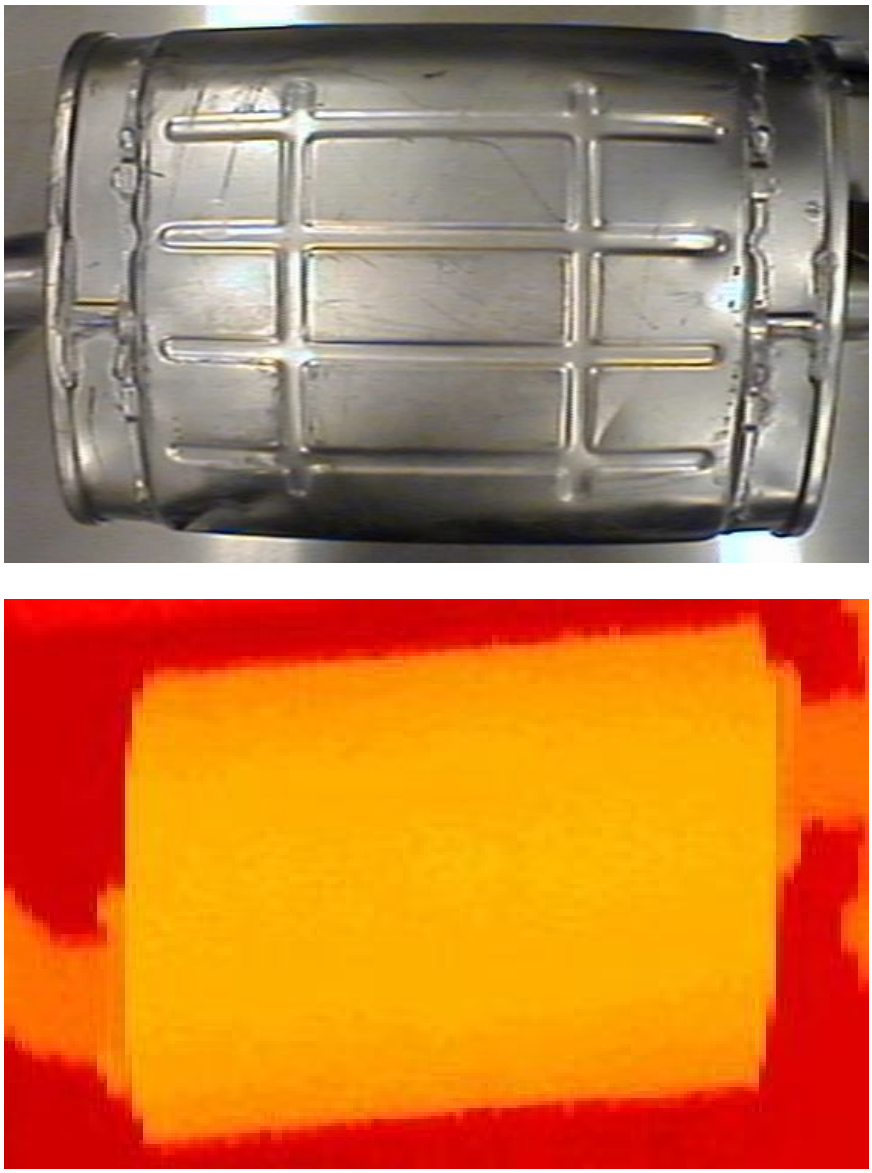

Figure 44: A visual image of a real, three-dimensional muffler and the pseudo-colored laser range image [19].

Figure 45: A visual image of a false two dimensional muffler (a poster) and the pseudo-colored laser range image [19].

- Mosaicing
- Image Pseudo-Coloring (Thermal, Laser Range Sensor Bricks)
- Three-Dimensional Scene Reconstruction (Visual and Laser Range Sensor Bricks)
- Source Localization (Radiological Sensor Brick)

6.3.5.1 Grayscale Imagery

Cameras of all types acquire images based on the relative intensity of radiation they receive. For the visual camera, these are known as black and white, or *grayscale* images. The visual and thermal camera, as well as the laser range scanner, produces grayscale images. The visual camera bases the intensity reading with respect to the amount of light it absorbs. The thermal camera bases the intensity reading with respect to the amount of infrared radiation it absorbs. The laser range scanner uses the range or distance data based on the time of flight of the emitted laser pulse that is received by the scanner. After the visual, thermal or range data (in the form of images) is acquired, three intelligent system techniques, developed in the IRIS Laboratory by graduate students, are used to enhance the data in order to maximize the gain of information from an inspection or application.

6.3.5.2 Mosaicing

As the camera sweeps through the vehicle underside, it acquires images that can be processed into a *mosaic*. A Mosaic is a large image complied from many smaller images. In the case of the visual, thermal, and laser range sensor bricks, a mosaic is a composite image produced from selected images taken during the sweep of the object of interest (the vehicle underside).

Figure 46 is an example of a mosaiced image. The figures display mosaiced and pseudo-colored images produced by intelligent systems developed by graduate students in the IRIS Laboratory. The mosaiced image allows the intelligent agent to make comparisons and evaluations based on one composite image. In the under-vehicle inspection scenario, mosaicing techniques produce an image of the entire vehicle underside, substantially facilitating the detection of threat objects in two ways. A direct examination of the mosaic of an object (an under-vehicle mosaic) assists human operator in the detection of threat objects by providing a prospective that enables better decision making. Indirectly, the mosaiced image can be compared to a standardized image (database) of how the object should appear. The inspection process described in section 8.3 can be effectively applied using mosaiced images of entire vehicle undersides.

6.3.5.3 Pseudo-Coloring

Pseudo-coloring is a technique that assigns portions of the visual spectrum (bands of frequencies or wavelengths) to ranges of values that are normally 8 bits in depth (from 0 to 263). Pseudo-coloring provides certain esthetic advantages when displaying or analyzing images. Portions of Figures 43, 44, and 45 are pseudo-colored images.

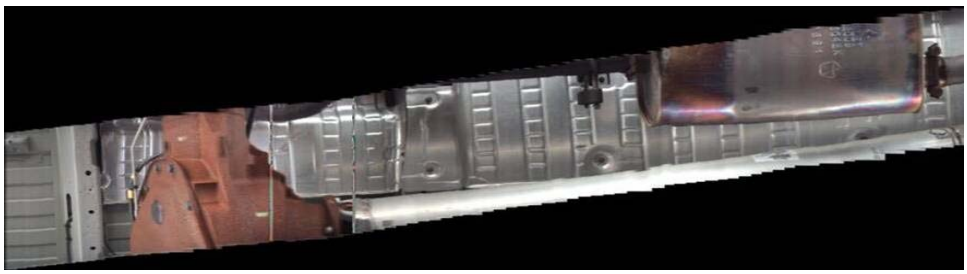

Figure 46: An image composed by the technique of Mosaicing [41].

6.3.5.4 Three Dimensional Scene Reconstruction

The three key elements of three-dimensional scene reconstruction are data collection, surface reconstruction, and object segmentation. Three dimensional scene reconstruction techniques can use various sensory data, typically visual and a laser range sensors are used. One challenge in this area is data collection. In the under-vehicle inspection scenario, the ambient illumination is a problem for the visual camera. Spectral reflectance off of metallic parts is a problem for the laser range sensor as is other types of measurement noise. When scanning the vehicle underside, millions of points of data are stored, some of which is redundant due to the extra scanning needed for occlusions. The intelligent systems developed in the IRIS Laboratory employs an efficient technique using *superquadric representation* that eases the computational burden in the surface segmentation phase of the reconstruction methodology. Figure 47 shows an under vehicle scene that has been reconstructed using superquadric representation.

6.3.5.5 Source Localization

Source localization techniques use intensity data produced by the radiological sensor brick to locate and identify the radiological source(s) that may exist in an area of interest. The radiological sensor brick intelligent radiological source localization systems can be tasked to locate radiological sources in a variety of environments and can be used in an under vehicle inspection paradigm. The localization methodology uses a form of *triangulation* to precisely locate the radiological source.

6.4 Sensor Brick System Demonstration

6.4.1 Overview

The modular SBS applied to an under-vehicle inspection scenario is described in this section. It describes the *concept* of using the four (4) sensor bricks of the system to inspect and detect threat objects (in conjunction with the Safebot mobility brick). At the present, the SBS can be used as an actual working, physically demonstrable system, to

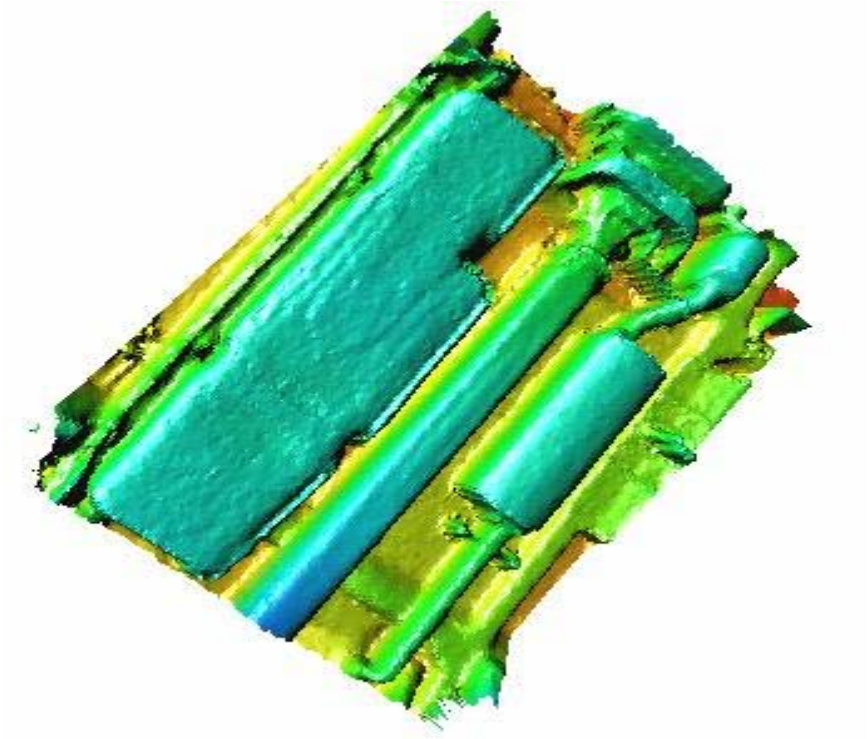

Figure 47: Three Dimensional Reconstruct of an Under Vehicle Scene [42].

perform the tasks necessary in the under vehicle inspection scenario. The initial phase of the design process of the SBS involves developing an understanding for the modular system concept and developing theoretical designs for the sensor brick prototypes.

Additionally, the initial phase consisted of surveys on the identification of the sensor brick components (the blocks). Many possible components that could be a part of the sensor brick (the sensor block, processing block, communications block and power block) were identified and compared in terms of functionality and cost. This phase of the system development concluded with the selection, assembly, and initial integration and testing of the sensor brick prototypes.

The first version of the SBS included the creation of an exterior chassis individually constructed for each sensor brick. The blocks were required to fit inside a single box fabricated from an aluminum sheet. Each brick was designed to have a single ON/OFF switch and was powered from one power supply battery. The chassis was designed so that the maximum height of the brick did not exceed 2.75 inches. The sensor bricks were

fabricated so that modular design principles were maintained (for example, to allow the quick interchange of sensor bricks on the mobility platforms).

At the present, the second version of the SBS is focusing on making all three bricks exactly the same, with each sensor brick using the same block components if possible. By making the package smaller, at least 2 bricks should be able to fit on Safebot (or any mobility platform). The electronic component layout has been modified so that the bricks are stable and reliable. Additionally , the sensor bricks, have been implemented with the JAUS specification for compliance to that important standard related to interoperable control. The next generation of sensor bricks will be smaller and more compact, so that in the future, all the bricks may fit into one physical package that can be mounted on a Safebot- type of mobility brick.

6.4.2 System Demonstration

The present version of the SBS can be physically demonstrated utilizing the Safebot mobility brick. Using the laser range sensor brick, it localizes the corner vertices of a vehicle, allowing the Safebot to sweep the vehicle underside. The laser range sensor locates the tires locations and then uses an intelligent systems application to sweep the vehicle underside. The data acquisition process can then be easily obtained; as shown in Figure 48: it shows the Safebot mobility brick using the laser range brick to autonomously sweep the IRIS Van.

Figure 48: Safebot with the visual sensor brick sweeping the underside of the IRIS Van.

7 CONCLUSION

Modular design is one way to address the design, fabrication, testing and application of cost-effective, intelligent mobile robotic systems. In this thesis, the application of modular design as the system architecture of an *intelligent mobile robotic system* has been shown to be the ideal system architecture with respect to constraints of the operational and technical architectures. The advantages of modular systems are numerous; modular systems, like the SBS, can be used in a large number of ways to solve a *variety* of real-world problems.

As more commercial mobile robotic systems are designed using modular principles, the systems will continue to undergo changes. However, with the fundamental architectures in place, the design and construction of these systems will no longer have to be closely associated with building and testing to suit specific or expected applications.

The system architecture is based on the concept of modular *sensor bricks*. Each modularly designed sensor brick utilizes four primary subsystems which are encapsulated in modules of commercially available technology. The *SBS* is also characterized by the essential composite of its sub-systems, each sensor brick's operational independence, and the ability of the system to support the seamless interchange of the sensor bricks. The modular design of the system facilitates the integration, maintenance and upgrade of the system components, ultimately improving operation reliability and effectiveness.

The modular system architecture of the SBS is ideally suited for the Realtime Control System operational architecture. Implementing the RCS framework provides a structure for control of the system as it matures into a more fully autonomous system. By implementing a working architecture, the SBS can lever the essential concepts concerning the control of intelligent systems that are embodied in the RCS software system called Neutral Message Language (NML).

Future work on the SBS should consider the incorporation of the NML system developed by Albus and his colleagues at NIST. NML will facilitate the further development of the SBS by reducing the burden associated with the development of a parallel system that NML would replace. By focusing on the implementation of the operational architecture (RCS) and its sister communications system (NML) which support the specifications set in the Joint Architecture for Unmanned Systems (JAUS), will allow the future SBS to incorporate more autonomous capabilities and be applied in many other tactical scenarios.

REFERENCES

[1] Albus, James S., Meystel, Alexander M. (2001). *Engineering of mind: an introduction to the science of intelligent systems*. NewYork: John Wiley and Sons.

[2] Murphy, Robin R. (2000). *Introduction to AI Robotic Systems*. Cambridge: MIT Press.

[3] Arkin, R. (1998). *Behavior-Based Robots*. Cambridge: MIT Press.

[4] Antsaklis, Panos J., Passino, Kevin M. (1993). Introduction to Intelligent Control Systems with High Degrees of Autonomy, In Panos J. Antsaklis and Kevin M. Passino (Eds.), *An Introduction to Intelligent and Autonomous Control*. (pp. 1 – 26). Boston: Kluwer Academic Publishers.

[5] Albus, James S. (1993). A Reference Model Architecture for Intelligent Systems Design, In Panos J. Antsaklis and Kevin M. Passino (Eds.), *An Introduction to Intelligent and Autonomous Control*. (pp. 27 – 56). Boston: Kluwer Academic Publishers.

[6] Meystel, Alex. (1993). Nested Hierarchical Control, In Panos J. Antsaklis and Kevin M. Passino (Eds.), *An Introduction to Intelligent and Autonomous Control*. (pp. 129 – 162). Boston: Kluwer Academic Publishers.

[7] Albus, J. S. (1991 May/June). Outline for a Theory of Intelligence. IEEE Transcripts on Systems, Man, and Cybernetics, Vol. 21, No. 3.

[8] Huang, Hui-Min, Albus, J., Kotora, J., Liu, R. (2003 October). Robotic Architecture Standards Framework in the Defense Domain with Illustrations Using the NIST 4D/RCS Reference Architecture. Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems. Las Vegas, NV.

[9] Smuda, B., Schoenherr, E., Andrusz, H., Gerhart, G. (2005). Deploying the ODIS robot in Iraq and Afghanistan. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 119-129), March. Orlando, FL.

[10] Albus, James (2002). 4D/RCS: A Reference Model Architecture For Unmanned Vehicle Systems, Version 2.0, NISTIR 6910, Gaithersburg, MD.

[11] Chatfield, J., et al (1998 January). New Architecture Directions. The Edge (The Mitre Corporation Advanced Technology Newsletter), Vol. 01, No. 03.

[12] Page, D., Fougerolle, Y., Koschan, A., Gribok, A., Abidi, M., Gorsich, D., Gerhart, G. (2004). SAFER Vehicle Inspection: A Multimodal Robotic Sensing Platform. Proceedings of SPIE (International Society for Optical Engineering), Vol. 5422, (Unmanned Ground Vehicle Technology VI), (pp. 549-560), April. Orlando, FL.

[13] Albus, James, Barbera, A., Scott, H., Balakirsky, S. (2005). Collaborative Tactical Behaviors for Autonomous Ground and Air Vehicles. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 244-254), March. Orlando, FL.

[14] Sukumar, S., Page, D., Gribok, A., Koschan, A., Abidi, M. (2005). Intelligent 3D Sensing for Robotic Inspection of Hazardous Facilities. Transactions of the American Nuclear Society, Vol. 92, San Diego, CA, (pp.54-55).

[15] Touchton R., Kent, D., Galluzo, T., Crane III, C.D., Armstrong II, D.G., (2005). Planning and modeling extensions to the Joint Architecture for Unmanned Systems (JAUS) for application to unmanned ground vehicles. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 146-245), March. Orlando, FL.

[16] Beronius, Alexander (2004). Case study of modularization in the industry. Development of Modular Products, (pp. 1-6).

[17] Office of the Undersecretary of Defense (AT&L) (2004). *The Department of Defense Joint Robotics Program Master Plan*.

[18] Naik, Nikhil A. (2004). Infrared Imaging Sensor Brick For Modular Robotic Systems. Project In-Lieu Of Thesis (PILOT) Report, IRIS Laboratory Internal Report, The University of Tennessee.

[19] Katwal, Santosh (2004). Range Sensor Bricks for Modular Robotics Systems. Master's Thesis, IRIS Laboratory, The University of Tennessee.

[20] Poduri, Anjana (2004). Video Sensor Bricks for Modular Robotic Systems. Project In-Lieu Of Thesis (PILOT) Report, IRIS Laboratory Internal Report, The University of Tennessee.

[21] Foster-Miller, Incorporated, (2005). http://www.foster-miller.com.

[22] Remotec, Incorporated, (2005). http://www.remotec.com.

[23] Society of Automotive Engineers technical standard 4893 (1996). Generic Open Architecture (GOA) Framework, SAE AS-5 GOA Task Group.

[24] O'Grady, P (1999). *The age of modularity – Using the new world of modular products to revolutionize your corporation*. Adams and Steele Publishing.

[25] Grinstead, B., Koschan, A., Abidi, M. (2005). Hybrid Self Localization for a Mobile Robotic Platform in Indoor and Outdoor Environments. Transactions of the American Nuclear Society, Vol. 92, San Diego, CA, (pp.52-53).

[26] Sukumar, S., Page, D., Gribok, A., Koschan, A., Abidi, M., Gorsich, D., Gerhart, G. (2005). Surface Shape Description of 3D Data from Under-Vehicle Inspection Robot. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 621-629), March. Orlando, FL.

[27] Chen, Q., Page, D., Koschan, A., Abidi, M (2005). A 'Brick'-Architecture-Based Mobile Under-Vehicle Inspection System. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 182-190), March. Orlando, FL.

[28] Boughorbel, F., A. Koschan, Abidi, M. (2005). A New Multi-Sensor Registration Technique for Three Dimensional Scene Modeling with Application to Unmanned Vehicle Mobility Enhancement. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 174-181), March. Orlando, FL.

[29] Grinstead, B., A. Koschan, Abidi, M. A. (2005). A Comparison of Pose Estimation Techniques. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage (Eds.), Proceedings of SPIE (International Society for Optical Engineering, Vol. 5804, (Unmanned Ground Vehicle Technology VII), (pp. 166-173), March. Orlando, FL.

[30] Kong, S., Heo, J., Abidi, B., Paik, J., Abidi, M. (2005). Recent Advances in Visual and Infrared Face Recognition – A Review. Computer Vision and Image Understanding, Vol. 97, No. 1, (pp. 103-135), January.

[31] Koschan, A., Page, D., Ng, J.-C., Abidi, M., Gorsich, D., Gerhart, G. (2004). Integration of Color and Shape for Detecting and Tracking Security Breaches in Airports. International Journal of Industrial Robot, Vol. 31, No. 5, (pp. 435-442), September.

[32] Zhang, Y., A. Koschan, Abidi, M.A. (2004). Superquadric Representation of Automotive Parts Applying Part Decomposition. Journal of Electronic Imaging, Special Issue on Quality Control by Artificial Vision, Vol. 13, No. 3, (pp. 411-417).

[33] Heo, J., Kong, S., Abidi, B., Abidi M. (2004). Fusion of Visual and Thermal Signatures with Eyeglass Removal for Robust Face Recognition. IEEE Workshop on Object Tracking and Classification Beyond the Visible Spectrum in conjunction with CVPR 2004, (pp. 94-99), July.

[34] Grinstead, B., Koschan, A., Abidi, M. (2004) "Developing a priori 3D Models of Large Environments to Aid in Robotic Navigation Tasks," Proceedings of SPIE Unmanned Ground Vehicle Technology VI, Orlando, FL Vol. 5422, (pp. 561-568).

[35] Koschan, A., Ng, J-C., Abidi, M. (2004). Multi-perspective Mosaics for Under Vehicle Inspection. Proceedings of SPIE Unmanned Ground Vehicle Technology VI, Vol. 5422, (pp. 1-10), April. Orlando, FL.

[36] Meystel, A. (1990). Knowledge Based Nested Hierarchical Control. In G Saridis (Ed.), *Advances in Automation and Robotics*, Vol. 2, (pp. 63-152). Greenwich, CT: JAI Press.

[37] Saridis, G. N. (1977). *Self-Organizing Control of Stochastic Systems*. New York: Marcel-Dekker.

[38] Brooks, R. (19??). Challenges for Complete Creature Architectures. Proceedings of First International Conference on Simulation of Adaptive Behavior, (pp. 434-443).

[39] Brooks, R. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation (now IEEE Transactions on Robotics and Automation), Vol. 1, No. 1, (pp. 1-10).

[40] The Joint Architecture for Unmanned Systems (JAUS) Domain Model, V3.1 (2004).

[41] Koschan, A., Page, D. Ng, J-C., Abidi, M., Gorsich, D., and Gerhard, G. (2004). SAFER Under Vehicle Inspection Through Video Mosaic Building. International Journal of Industrial Robots, Vol. 31, No. 5. (pp. 435-442).

[42] Sukumar, S., Page, D., Gribok, A., Koschan, A., Abidi, M., Gorsich, D, Gerhard, G. (2005). Surface Shape Description of 3D Data from Under-vehicle Inspection Robot. Proceedings SPIE Unmanned Ground Vehicle Technology VII, Vol. 5804, Orlando, Fl, (pp. 621-629)

Vita

James Robert Wilson III received the Bachelor of Science in Industrial Engineering in December 1987, from The University of Tennessee, Knoxville. From 1988 until 1990, he worked as a research assistant in the Resource Modeling and Technology Economics Group, in the Energy Division at The Oak Ridge National Laboratory. From 1995 until 2001 he taught mathematics at the secondary level in both parochial and public school systems. In 2002, returning to The University of Tennessee's College of Engineering, he earned a second undergraduate degree, the Bachelor of Science in Electrical Engineering, in May 2004. Immediately after graduation, he accepted a graduate research assistantship in the Imaging Robotics and Intelligent Systems (IRIS) Laboratory. The Master of Science Degree in Electrical Engineering is expected to be completed in December 2005.