International Journal of Automotive Technology, Vol. 3, No. 2, pp. 79–88 (2002)

1229–9138/2002/008–06

# SOFTWARE ARCHITECTURE FOR ADAPTIVE COLLISION AVOIDANCE SYSTEMS

Jeremy Blum[1)*] and Azim Eskandarian[2)]

[1)]Research Associate, Center for Intelligent Systems Research, George Washington University, Virginia Campus
[2)]Director, Center for Intelligent Systems Research, George Washington University, Virginia Campus

*(Received 12 December 2001; Revised 6 February 2002)*

**ABSTRACT**–Emergent Collision Avoidance Systems (CAS's) are beginning to assist drivers by performing specific tasks and extending the limits of driver's perception. As CAS's evolve from simple systems handling discrete tasks to complex systems managing interrelated driving tasks, the risk of failure from hidden causes greatly increases. The successful implementation of such a complex system depends upon a robust software architecture. Most of the difficulty in implementing system arises from interconnections between the components. The CAS architecture presented in this paper focuses on these interconnections to mitigate this problem. Moreover, by constructing the CAS architecture through the composition of existing architectural styles, the resulting system will exhibit predictable qualities. Some of the qualities represent limitations that translate into constraints on the system. Others are beneficial aspects that satisfy stakeholder requirements.

**KEY WORDS :** CAS (Collision Avoidance System), ACC (Adaptive Cruise Control)

## 1. INTRODUCTION

Automobile manufacturers are beginning to deploy the first generation of CAS's. The systems handle relatively simple and independent scenarios. However, there is significant concern that even these simple CAS's may actually make the driving environment more dangerous. In order to mitigate this concern, future CAS's must include an adaptive capability that allows the CAS to learn the driving style and limitations of a particular driver. Moreover, as more driving situations are managed by CAS's, the subtle interactions between CAS components can lead to catastrophic failure of the system.

This paper presents a robust software architecture for future CAS's that manages the complexity of the resulting system by instantiating the CAS architecture from a proven architectural styles. These architectural styles result in predictable aspects of the system. The qualities include benefits that are essential for meeting the system requirements, and drawbacks that present constraints on the components in the architecture. By addressing these drawbacks in the design phase, the resulting system is more resilient to catastrophic failure.

The remainder of the introduction contains an overview of CAS's, followed by a section detailing the benefits of a Software Architecture. The architecture is described from various viewpoints. Section 2 presents a ballpark view of the architecture. The owner's view of the architecture is presented in Section 3. Section 4 contains a more detailed description of the components, interconnections, and constraints that comprise the designer's view of the architecture. Section 5 presents the conclusion.

### 1.1. CAS Overview and Requirements

CAS's are still in an incipient state, addressing relatively simple and isolated tasks. The areas of control can be divided into lateral control and longitudinal control. The output of the system includes control signals, driver warnings, and driver perception enhancement.

In order to limit the liability associated with autonomous vehicles, CAS's in the near future will keep the driver "in the loop" and seek to enhance the ability of the driver to avoid dangerous situations. In order to limit the investment needed for centralized CAS system, the locus of control of the CAS will be within an individual vehicle. Another approach to CAS a centralized planning layer that coordinates the actions of multiple vehicles. (Zhang *et al.*, 1994; Varaiya, 1993) However, manufacturers of control systems that automate driving tasks are even more likely to be defendants in a lawsuit. Therefore, completely automated highways with centra-

**Corresponding author.* e-mail: jblum@seas.gwu.edu

lized control are unlikely to be realized any time soon. Equally unlikely is an autonomous vehicle that relieves the driver of any responsibility.

Lateral control systems include curve preview systems, lane keeping systems, and lane-changing aids. Curve preview systems can inform drivers of upcoming road layout, offer a suggested safe speed, and perform automatic braking when a curve is being approached too fast. (BMW, 1998; Shields and Roser, 2000) Lane keeping systems aim to keep the vehicle from straying from its current lane. Simple lane changing aids that detect objects in a driver's blind spot are also becoming available. (McLoughlin *et al.*, 1993).

Longitudinal control includes Adaptive Cruise Control (ACC), Stop and Go Cruise Control, and Intersection Control. ACC attempts to maintain a constant following distance and is now found in many commercially available vehicles. (Shields and Roser, 2000) The next generation of ACC, Stop and Go Cruise Control, must manage stop and go traffic which is characterized by sudden lane changes, frequent changes in target dynamics, and multiple targets. (Venhovens *et al.*, 2000) Intersection Control is not yet available. It must handle the four most common collision scenarios in intersections – left turn across a vehicle's path, inadequate gap on a perpendicular path, violation of a Traffic Control Device (TCD), and premature intersection entry. (Jacoy and Knight, 1998) To accomplish this task, the controller will need the ability to sense the presence and type of TCD's, the dimensions of the intersection, and the dynamics and intentions of target vehicles.

The output of a CAS is presented via traditional Head-Down Displays, newer Head-Up Displays (HUD's), auditory and haptic warnings, and in a number of other innovative fashions. Haptic warnings are sensed through touch and bodily movement. They tend to produce faster responses than the other alarms. The driver must learn to associate the haptic warning with the nature of the hazard. For example, the force needed to press the accelerator could be increased to signal the driver to decelerate.

Other CAS output is quite varied. Night vision and intelligent headlights improve upon the limits of human vision. (GM, 2001; Kamiya, 1996) Another strategy is to measures the limits of human vision in low-visibility situations like heavy fog, and notify the driver of a maximum safe speed based on stopping distance. (BMW, 1998) Many vehicles utilize GPS/GIS systems for navigation either in production models or research vehicles. (Mercedes-Benz, 2001; Mimuro, 1996; GM, 2001) Drowsy driver detection systems have included a wide range of response modalities including issuing warnings, changing the cabin temperature, vibrating the driver's seat, and even pumping refreshing fragrances into the cabin. (Freund *et al.*, 1995)

As remarkable as these systems are, there is significant concern that the CAS may actually make the driving environment more dangerous. The user's requirements arise from the mitigation of this danger. Specifically, the CAS must be able to adapt to driver behavior. Non-adaptive timings and warnings have the following consequences:

- Driver vigilance may be reduced, if the CAS inspires either an unrealistically high level of driver trust in the system or misinterpretation of the CAS vehicle as an autonomous vehicle. (Moray, 1990)
- A driver may have to pay excessive attention to the CAS to prevent unwanted warnings or interventions. (Goodrich and Boer, 2000)
- Drivers may misunderstand the meaning of the alarms.
- The alarms issued by the CAS may cause driver embarrassment or irritability. (Groeger *et al.*, 1993)
- The CAS may provide information and recommendations that are not appropriate for a given driver's limitations. (Michon and Smiley, 1993)
- The CAS may overwhelm the cognitive capability of drivers, particularly older drivers. (Hendersen and Suen, 1999)

Furthermore, there is concern that other drivers may attempt to take advantage of the limitations of a CAS-equipped vehicle and trick it into a crash situation. This concern can be mitigated through a "black box" which records information about the vehicle and its surroundings for the purposes of reconstructing an accident. (Menig and Coverdill, 1999)

### 1.2. Need for a Software Architecture

The benefits of a software architecture are particularly applicable for CAS's. A systems architecture should not develop in an ad hoc way – emerging as components of the system are developed. Selecting the correct architecture is often crucial for successful system design, and moreover, selecting the wrong architecture may have disastrous results. (Garlan and Shaw, 1993)

This is particularly crucial in the development of complex safety-critical applications. Specifically, creating a software architecture for CAS, affords benefits in the ability to manage dangers that arise from inherent in system's complexity, to predict aspects of system performance, and to enhance possibilities for reuse.

Highly complex systems are prone to catastrophic failure. (Carlson and Doyle, 2000) The scale of the potential failure is a result of interactions between components. Current CAS systems operate essentially in isolation with limited opportunity for conflict. This situation will change rapidly as more functions are automated, and unexpected conflicts arise between modules. Growing a CAS through ad hoc composition of individual components will miss these interactions. The key to managing this complexity is develop the CAS based on a well-defined software architecture.

A seminal analysis of driving tasks identified 43 main driving tasks, which were further broken down into 1700 sub-tasks. (McKnight and Adams, 1970) As a CAS handles more of the tasks, conflicts will arise between tasks. For example, lane-keeping and collision-avoidance tasks may issue conflicting control signals. Consider a target car traveling the same direction in the next lane. If this car starts to veer into the subject car's lane, the collision avoidance function may want to steer the car away from the target. However, the lane-keeping function may resist this action. While this conflict is obvious, many more interactions will arise in subtle and unexpected ways. For example, researchers have determined that ACC will make it more dangerous to pass vehicles on a two-lane road.

In addition to managing complexity, software architectures also improve the predictability of the final system's properties. By using existing software architectures, the system designer can select architectural patterns whose advantages, as demonstrated in previous instantiations, are desirable for the current system. At the same time, the designer can address demonstrated weaknesses during the design process.

The third benefit realized through the implementation of software architectures is the increased capability for reuse. Reuse of hardware and software components provides a number of benefits. One benefit is that as components are reused, they are subject to more testing, and therefore more bugs can be identified and rectified. Reuse also reduces system development cost by minimizing the total amount of software that must be developed. Furthermore, reuse promotes the ability to consolidate system hardware. For example, if the CAS components reuse a module that reasons about target vehicle dynamics, they may also all be able to rely on a single set of sensors that detects target vehicles.

Reuse can be stressed at multiple points during the system development process. However, the potential scope of the reuse is the largest when specifying the system architecture, because at this point, the system has the fewest constraints. (Perry and Wolf, 1992) Often it is too late to consider reuse at the component level because components already are too constrained.

## 2. BALLPARK VIEW OF ARCHITECTURE

One architecture description is insufficient for the specification of a software system. Rather than one view of the architecture, there is a set of software architectures, each having a differing perspective. (Zachman, 1999) These perspectives include a scope description providing a ballpark view, an owner's view, and a designer's view.

**Collision Avoidance System**
- **Controlled System**
  - **Driver**
  - **Vehicle**
  - **Limitations**
- **Controlling System**
  - **Sensors**
  - **Control Logic**
  - **Actuators**

Figure 1. Entity hierarchy.

The ballpark view includes a list entities and processes. This viewpoint provides a birds-eye view of the entire system. Figure 1 shows the entity hierarchy for the CAS. The higher-level entities are composed of the lower level entities. At the top level are categories common typical of embedded systems. As in other embedded systems, the essential interaction is that of a controlling system regulating a controlled system.

The controlling system is comprised of sensors, control logic, and actuators including servo-controllers, and warning and other driver information systems. It uses sensors to determine the current state of the controlled system. The controlling system uses this state to issue appropriate control signals and warnings.

There are three major entities in the controlled system. (Goodrich and Boer, 2000) The driver includes the intentions, conditions, and limitations. Driver intentions include whether the driver is stopping or changing lanes. Knowledge of driver intentions is critical to understanding when a CAS action is warranted, because a misinterpretation of driver intentions may cause false alarms.

Driver conditions include drowsiness or level of driver attention. Driver limitations include driver reaction times, familiarity with given driving tasks, and visual acuity. Knowledge of driver conditions and limitations are essential to meet the adaptability requirements of a CAS. The vehicle entity includes information about the vehicle's dynamics, position, and limitations. The data also includes information about current state of controls, such as the steering angle, throttle angle, and brake state. The environment includes more than the road geometry and the weather conditions. The environment actor also includes the position, velocity, and intentions of target vehicles.

In addition to an entity list, the scope description includes a list of processes and a description of the locations of the processing. As mentioned previously, the primary processes for the CAS include taking appropriate control actions, issuing appropriate control signals, and

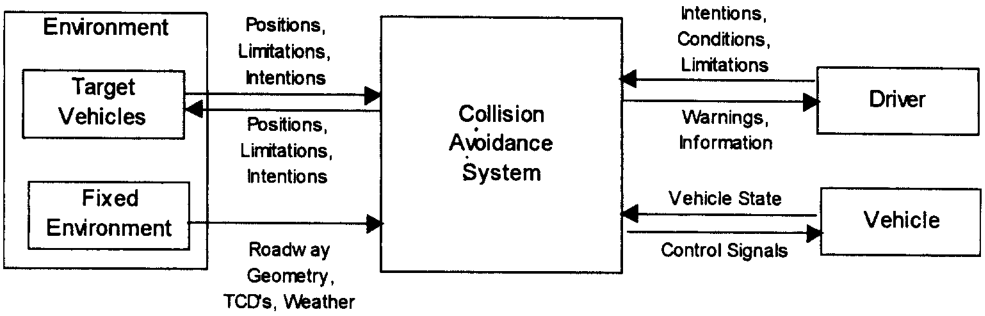


Figure 2. Entity relationship diagram.

extending the limits of driver perception. In addition, as mentioned in the introduction, this paper is considering CAS systems that keep the driver in the loop.

## 3. OWNER'S VIEW OF ARCHITECTURE

The owner's view begins to provide more information about the interaction of components in the system. It can be depicted in three diagrams – an entity relationship diagram, a functional flow as captured by a system state diagram, and a network diagram.

The entity-relationship diagram characterizes the interactions between the entities described in the previous section. One additional relationship is depicted here. That is that the CAS of the subject vehicle communicates information about driver intentions and vehicle status to the CAS of nearby vehicles.

The state diagram in Figure 3 captures the functional flow of the system. The CAS events come in a variety of forms. Examples include the detection of an intersection for intersection control; forward motion for lane-keeping; or the onset of traffic congestion for an intelligent routing system.

When an event is detected, the CAS determines what it

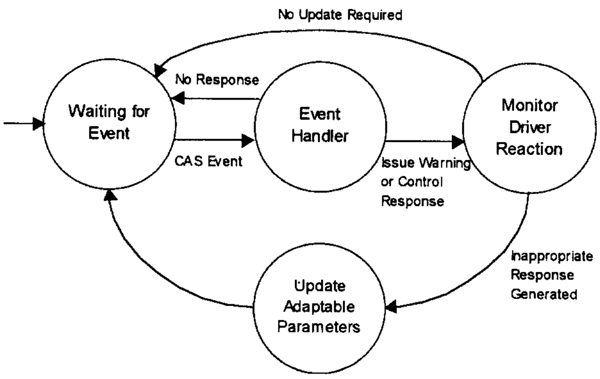


Figure 3. CAS state diagram.

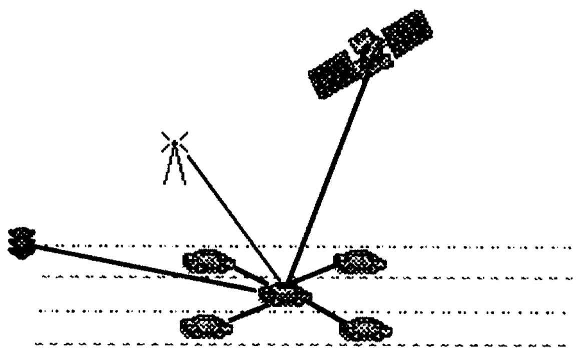

Figure 4. CAS network.

believes to be an appropriate response – a warning, driver information, or control response. Because of the likelihood of an inappropriate response, the system may evaluate the response by monitoring driver reaction. For example, if the driver ignored a warning without adverse consequences, or the driver overrode a CAS control signal, the system has evidence that its response was inappropriate. A suitable alteration of its adaptive response should then be made to attempt to avoid repetition of this error.

The CAS Network Diagram depicts the data links that the CAS may rely on to glean information about the environment. Nearby cars communicate driver intentions and vehicle dynamics. For example, the European PROMETHEUS project used millimeter wave radio for inter-vehicle communication for vehicle platoons and other collision avoidance functions. (Sourour and Nakagawa, 1999) GIS/GPS plays a role in CAS in intelligent routing systems. It could also be used to gather information about upcoming road geometry, intersections, and traffic control devices (TCD's). Roadway to vehicle communication is possible at some point in the future, providing information about TCD's or road geometry. (Hayami *et al.* 1999; Choi, 2000) The wireless internet is also playing a role in CAS. (GM, 2001; Eaton, 1998)

## 4. DESIGNER'S VIEW OF ARCHITECTURE

The designer's view provides detailed information about the components that comprise the CAS. Complex architectures are often the instantiation of multiple architectural styles, combined through either through a hierarchy or through a mixture of architectural connectors. (Garlan and Shaw, 1993) This view of the CAS is described by a hierarchical heterogeneous architecture in a top-down approach through partition and concretization. (Abd-Allah, 1995)

### 4.1. High-Level View: Pipe and Filter Architecture

At the highest level, the CAS is a pipe and filter architecture. Pipe and filter architectures offer four major benefits that are relevant to the CAS, and two drawbacks that must be addressed during the design phase.

The first advantage is that the architecture offers easy maintenance and enhancement. (Garlan and Shaw, 1993) Maintaining or upgrading the system is simply a matter of upgrading an individual filter. As long as the input and output remain the same, new components can be freely switched with old ones. The second advantage is that the system's functionality can be understood as a composition of the functionality of the individual filters. (Garlan and Shaw, 1993) A third advantage of this architecture is its support for concurrency. As long as there is no bottleneck to the processing, each filter can run on a separate processor and in parallel with other filters.

The last advantage is that the systems support reuse. (Garlan and Shaw, 1993) The CAS supports reuse in a different fashion than pipe and filter systems that exist totally in software. Much of this reuse is enabled through using a canonical format for the data, which allows enormous freedom in mixing and matching filters. However, the data in a CAS is so different from filter to filter, that presenting a canonical data format is not desirable. Rather, the CAS generates reuse through the elimination of redundant hardware sensors and filters and with the possibility of transmission of data on a common system bus as opposed to wiring dedicated to individual CAS tasks.

The architecture, however, does have two significant disadvantages that must be addressed during system design. The first disadvantage concerns system performance. If a filter cannot begin processing before the previous filter has finished with the data, this architecture will degenerate into batch processing. (Hoffman *et al.*, 1996) Batch processing will have difficulty meeting the hard real-time processing requirements of the CAS. The second disadvantage of the pipe and filter architecture arises due to the use of multiple sensors. In this architecture, it is difficult to maintain correspondences between two separate but related data streams. (Garlan and Shaw, 1993)

The high level depiction of the architecture is shown in Figure 5. Sensors detect the current state of the environment. Filters refine the raw data from the sensors. A Sensor Synthesizer module reconciles conflicting information from different filters, and relays the state of the controlled environment to an event handler. The event handler dispatches notification of an event to appropriate CAS functions, which then issue information, warnings or control responses.

#### 4.1.1. Sensors

In order to provide adaptability to an individual driver, CAS will need to have access to the same information that a driver uses to make driving decisions. Since humans perceive the driving environment primary with sight, this additional data will be readily available from vision sensors. Due limitations in other sensor technologies, vision sensors must be included, so no additional sensors will be needed to provide adaptability. Furthermore, the sensors offer the first opportunity for reuse in the CAS architecture.

Sensor technologies include radar and laser and yaw rate gyroscopes. Adaptive Cruise Control is limited to high-speed, well-ordered roads because of the limitations of laser and radar sensors. These sensors have difficulty distinguishing between stopped vehicles and fixed objects, including overpasses, overhead signs, and roadside clutter. (Barber and Clarke, 1998) Consequently, all motionless objects are ignored, and so ACC is limited to well-ordered roads with minimum speeds on the order of 40 km/hr. (Venhovens *et al.*, 2000)

Roadway geometry can also confuse CASs that rely on radars and lasers. On curves, vehicles in nearby lanes

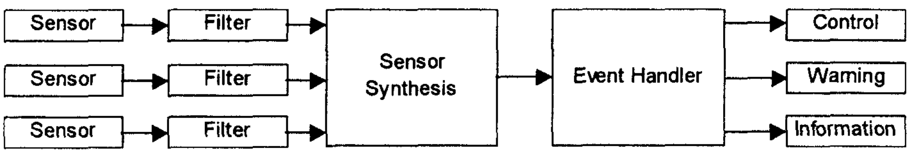


Figure 5. Pipe and filter architecture.

may be falsely identified as being in the same lane. (Barber and Clarke, 1998) Yaw rate gyroscopes, which can detect the curvature in a vehicles trajectory can mitigate this problem somewhat, but only if the vehicles trajectory matches the upcoming road curvature. (Behringer and Maurer, 1996) Vision sensors, though, do not suffer from the same drawbacks as laser and radar, and therefore will not only enable Stop And Go Cruise Control, but also provide the CAS with the means to extract of additional data that humans use for driving decisions.

Because the sensors offer data that are used by multiple functions of the CAS, they offer the first major area for reuse in the system. Rather than having a separate set of sensors for each CAS function, all of the functions will rely on the same set of sensors. There is actually quite a bit of redundancy that will be eliminated through the use of shared sensors. Some experimental vehicles use multiple cameras to provide distance information as well as 360° view of the exterior of the vehicle. The Mitsubishi Advanced Safety Vehicle, for example, uses two lane detecting cameras and three stereo cameras, for a total of eight cameras. (Mimuro *et al.*, 1996)

4.1.2. Sensor processing

The next set of filters in the top-level view of the architecture performs signal processing on the sensor input. These filters must address the two disadvantages of the architecture – the tendency for it to degenerate into batch processing and the need to maintain the correspondence between related data streams.

Object recognition filters, particularly for vision sensors, have a high likelihood to degrade the architecture into batch processing. A conventional algorithm may take too long to analyze a complex scene, and for simple scenes, it may waste computing resources. One solution to this problem is an anytime algorithm. (Sobottka and Burke, 1999) These algorithms have the property that a solution is available anytime the algorithm is terminated and the solution improves with additional computing time. The algorithm starts with a restricted heuristic to obtain an initial and possibly sub-optimal solution. As time remains, more computationally intensive methods iteratively improve the solution.

Additional processing must be done to synthesize the output from the various filters. Because of filter limitations, different filters may present conflicting information. The Sensor Synthesis filter must rectify these conflicts. Furthermore it must mitigate the other drawback inherent in this architecture. It must maintain correspondence between outputs of various filters. Therefore, this drawback of the pipe and filter architecture creates a constraint on the sensor synthesis module.

4.1.3. Event handler

The details of the event handler filter are left abstract at this level. Only its interfaces are defined. At this level, the Event Handler filter includes both the event announcers and the event handlers.

Event-handler architectures come in a wide variety of forms. The common invariant of these architectures is that the announcer of events does not know which components will be affected by an event. (Garlan and Shaw, 1993) Consequently, the announcers cannot make assumptions about order of processing or even about what processing will occur as a result of their events. The order of processing is particularly important with a CAS, as certain functions are more safety critical than others.

Like the pipe and filter architecture, an event-driven architecture offers advantages that benefit the CAS and drawbacks that must be addressed through constraints as the specific components of the architecture are described. (Garlan and Shaw, 1993) The advantages include easy expansion and easy implementation of concurrency. Disadvantages include difficulty in debugging, difficulty in reasoning about correctness, and global resource management issues.

Adding new tasks to an event-driven architecture is easy because by default components do nothing in response to an event. Concurrency is also easy to achieve in this architectural style. The event announcers and event handlers can run simultaneously on their own processors. The ability to easily move complex CAS functions to their own processor provides the ability of the system to scale to handle more scenarios.

However, one of the major disadvantages of event-handler architecture is that it is difficult to debug and guarantee the correctness of the program. One ramification of this style is that by default announcers relinquish control over the performance of the computation. As such, announcers cannot guarantee order of processing or even if an announced event will be handled by a component. Moreover, the uncertain order of events combined with persistent state information makes reasoning about correctness nearly impossible.

The other pertinent disadvantage of event handlers is that resource management can become an issue. If there is a large amount of state data, it may be impossible to transmit state information with an event. In such a case, a globally accessible memory is needed for components to access state information. The global memory then presents resource management issues. Furthermore, since there is no explicit invocation inherent in the architecture, processor resource management can become a problem as multiple event handlers vie for this resource.

The drawbacks of event handlers will be addressed in the next section, as we concretize and constrain the implementation. These problems would likely be present in any CAS. However, without recognizing the use of this established architectural pattern, it would be difficult to identify and address these types of characteristic drawbacks in the design process.

4.2. Concretization of Event Handler – Blackboard

In order to address the drawbacks raised in the previous section, the event handler is instantiated as a blackboard system. The blackboard system is typically made up of three components: a repository, knowledge sources, and a scheduler. (Craig, 1995)

The repository stores current state information and a proposed solution to a problem. Each knowledge source encapsulates an algorithm that uses the state information to update the proposed solution. In one type of blackboard, based on the current state, a controller activates appropriate knowledge sources. Blackboards are particularly appropriate for situations like collision avoidance where there is no direct solution available, and multiple CAS functions must be utilized to obtain a valid solution.

The blackboard metaphor embodies a high degree of reuse. The centralized storage of state information allows for multiple CAS functions to share significant hardware resources. Furthermore, the reasoning and logic in CAS functions can also be reused. For example, many CAS functions are enhanced through with kinematics model for the subject vehicle and surrounding vehicles. This kinematics model can be encapsulated in one knowledge source and then shared by multiple CAS functions.

Another advantage is the partitioning of the system functions into knowledge sources with well-defined interfaces facilitates development by a team of programmers. Each programmer can focus on an individual knowledge source without worrying about duplication of effort.

Blackboards do have a disadvantage concerning the quality of the input data. Error in the input can lead the blackboard to produce erroneous results. This drawback creates an additional constraint that is handled by the Sensor Synthesis filter. Due to the limitations in sensor technologies, different sensors may produce conflicting data. The Sensor Synthesis module will rectify these problems with knowledge of sensor limitations.

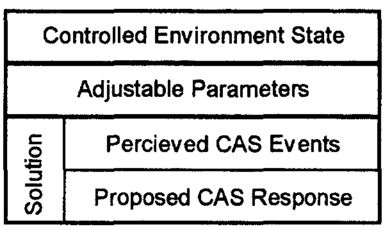


Figure 6. Segmentation of the blackboard repository.

As shown in Figure 6, the repository used by the CAS is segmented. Part of the blackboard is used to reflect the current state of the controlled system, including state of environment, vehicle, and driver. The blackboard also contains a proposed solution in two areas. The first area contains the transformation made by the knowledge sources from the state information to the identification of CAS situations, such as "straying from lane at rate of x ft/s" or "ignoring an upcoming stop sign." The last segment of the repository contains the adaptability data that models the behavior of a particular driver. Since multiple drivers may use a given CAS, some means must be provided to store multiple driver records and identify the current driver. Details of the format of this adaptability data and the constraints of the knowledge sources are provided in section 4.3.

The scheduler must guarantee that a CAS knowledge source is activated when appropriate. To accomplish this, the scheduler needs to know the controlled environment states associated with a given knowledge source. For example, lane keeping should not be activated when the vehicle is stationary or when vehicle is changing lanes. Adaptive cruise control should not be used for speeds less than 40 km/hr. The scheduler also guarantees the worst-case time to activation of a CAS function. The scheduler can invoke the most time-critical knowledge sources first. This could be accomplished assigning priorities to the knowledge sources.

The blackboard scheduler addresses the disadvantages of an event-handling architecture. First, the scheduler provides for explicit invocation of knowledge sources. The central repository and the scheduler also mitigate the event-handler problem with global resource management. The repository provides a pattern for global memory management. The scheduler is responsible for processing resource management by controlling when knowledge sources are scheduled.

The scheduler embodies the major difference between this blackboards and an autonomous agent architecture, which have also been suggested as possible choice for some CAS features. (Burmeister *et al.*, 1997) Asynchronous Teams of Autonomous Agents, A-Teams, are one type of agent architecture. (Talukdar, 1998)

A-Teams have many similarities to blackboard. Both produce a distributed search of a solution space. Both contain a common solution or solutions that are modified by agents or knowledge sources. The essential difference is that in an A-Team, agents retain autonomy by maintaining control over when they run. In this blackboard, the scheduler controls when knowledge sources run.

A-Teams have been used collision avoidance in robotics. (Kao *et al.*, 1996) However, A-Teams do not mitigate the problem inherent in this software architecture. The scheduler's inherent ability to manage processor resources and guarantee the order of processing is critical for the CAS. These qualities are not inherent to autonomous agent architectures.

4.3. Knowledge Sources

The specification of the components that comprise the knowledge sources provides additional constraints on the blackboard. These knowledge sources include Kinematics, Black Box, Adaptive, Learning, Baseline, Synthesis and Notification components. The functions of some of these knowledge sources have been specified earlier. The Kinematics component provides a model of the dynamics of host vehicle and target vehicles. The "black box" component can discourage other drivers from attempting to track a CAS-equipped vehicle into a crash.

The Synthesis component resolves potentially conflicting control signals issued by CAS functions. This module is critical for managing the complexity that arises from component interactions. As mentioned previously, these interactions may be quite subtle. However, by having a central component responsible for these conflicts, these interactions can be managed.

The Notification knowledge source must contain an adaptive controller to ensure that signals sent to the driver are appropriate for their cognitive ability. The logic of this component will be quite complex, as it must gather information about the limitations and experiences of a driver. A learning module accompanies this knowledge source. This module must be able to distinguish the cause of unexpected reaction to alarms. Depending on whether the cause is a false alarm, a misunderstood alarm, or an alarm that overwhelmed a driver, appropriate corrections to preferred warning modality should be made.

The static CAS knowledge sources play a role similar to current CAS controllers. However, the controllers are no longer optimized for median behavior, but rather to ensure that the adaptive CAS neither reproduces faulty behavior nor issues excessive false alarms. The static controller will now raise alarms at the latest safe time, a time at which a high level of crashes can be avoided. For example, for forward following distance control, a one-second warning would prevent 90% of accidents. (Woll, 1997)

Each adaptive CAS function is managed by two knowledge sources. One provides the adaptive response and the other monitors driver reaction to tune the response for a particular driver. The learning technologies that could be used to implement this adaptable control are varied.

A simple scenario occurs if driver behavior can be modeled accurately by a simple function with a few adjustable parameters. For example, suppose, as shown in Figure 7, following distance in heavy traffic can be modeled by the spring damper diagram, which has been

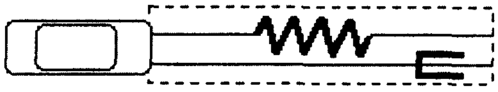

Figure 7. Virtual bumper.

used as a control strategy for ACC. (Gorjecstani, 2000) The CAS would produce a braking force or acceleration force that is proportional force exerted by the system. Then the spring and damping coefficients for a given driver could be found using regression analysis.

Fuzzy logic controllers are another way to provide adaptability. The controllers are appropriate if one understands the rules that govern driver performance. For example, fuzzy logic controllers are useful for longitudinal control. (Araki *et al.*, 1996)

For many CAS tasks, it will be impossible to divine the structure of a function that models human driving behavior. Even if the structure of a function can be identified, the parameters to a function may vary based on other environmental conditions. For example, since the incidence of emergency braking increases in dense traffic, a driver may naturally extend the length of a virtual bumper under these conditions. (Hu *et al.*, 1999)

For these situations, a lattice-like datafield or an associative datafield may provide a simple and effective control strategy. (Schmitt *et al.*, 1994) With these storage devices, the value of the control parameter is found by interpolating between a number of previously observed values.

Unfortunately, many CAS functions will have too many inputs for datafields to be effective. Drivers use many more inputs than static CAS functions. For example, a recent study found that like in a static CAS, driver's perception of safe speed not only varied by road geometry and type, but also based on whether the roadside environment was open farming, heavily treed, or walled. (Fildes *et al.*, 1990)

In these cases, the control strategy might utilize neural networks. Feed-forward neural networks have been shown to be universal approximators of non-linear functions. (Hornik *et al.*, 1989) Consequently, they are a powerful tool if the structure of the underlying control function is unknown. Drowsy driver detection is an area where neural networks are being employed. Sayed and Eskandarian proposed a neural network approach for distinguishing drowsy drivers' steering activity and generating a warning signal. (Sayed and Eskandarian, 2000) They report a high success rate based on driving simulator data. Other systems include monitoring the duration of time that the drivers' eyelids stay closed during blinks. (Freund *et al.*, 1995) Eskandarian and Thiriez used a neural network model of human sensory motor responses for a collision avoidance system. (Eskandarian and Thiriez, 1998) Hybrid neural network/ fuzzy controller strategies have been demonstrated for learning obstacle avoidance behavior. (Sidani and Gonzalez, 2000; Fangqin and Feng, 1999)

## 5. CONCLUSION

The development of a software architectures is crucial for complex systems like CAS's. In a complex system such as a CAS, the interconnections between the components are as important as the components. In fact, these interactions, left unexamined, can have disastrous effects. The software architecture presented focused on managing dangers manifested in these interactions.

By constructing the architecture through the composition of existing architectural styles, the resulting system exhibits predictable qualities. Some of the qualities represent limitations of the system. Other qualities represent beneficial aspects that can satisfy stakeholder requirements. The predicted limitations of the resulting CAS translated into constraints imposed on components or constraints that led to a particular choice of components. The result is a robust design for an effective and comprehensive Collision Avoidance System.

## REFERENCES

Abd-Allah, A. (1995). *Composing Heterogeneous Software Architectures*. PhD Dissertation: University of Southern California.

Araki, H., Yamada, K., Hiroshima, Y. and Ito, T. (1996). *Development of a Rear-End Collision Avoidance System*. Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 224-229.

Barber, P. and Clarke, N. (1998). Advanced Collision Warning Systems. IEE Colloquium on Industrial Automation and Control: Applications in the Automotive Industry, Digest No. 1998/234, 2/1–9.

Behringer, R. and Maurer, M. (1996). Results on Visual Road Recognition for Road Vehicle Guidance. Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 415-420.

BMW. (1998). Driver Assistance at BMW. http://www.telematik.de/bmw/english/fe_fah.htm.

Burmeister, B., Haddadi, A. and Matylis, G. (1997). Application of multi-agent systems in traffic and transportation. *IEE Proceedings of Software Engineering* **144**, **1**, 51–60.

Carlson, J. and Doyle, J. (2000). Highly optimized tolerance: Robustness and design in complex systems. *Physical Review Letters* **84**, **11**, 2529–2532.

Choi, S. B. (2000). The design of a look-down feedback adaptive controller for the lateral control of front-wheel-steering autonomous highway vehicles. *IEEE Transactions on Vehicular Technology*, **49**, **6**.

Craig, I. (1995). *Blackboard Systems*. Ablex Publishing Corporation, New Jersey.

Eaton. (1998). Eaton VORAD Introduces SmartCruise: World's First Look at Radar-based Adaptive Cruise Control.http://truck.eaton.com/na/news_about_us/news_releases/1998/releases/641998_smart.htm.

Eskandarian, A. and Thiriez, S. (1998). Collision avoidance using a cerebellar model arithmetic computer neural network. *Computer-Aided Civil and Infrastructure Engineering* **13**, 303–314.

Fangqin, L. and Feng, L. (1999). Fuzzy Control of Automatic Automobile Obstacle Avoiding. Proceedings of the IEEE International Vehicle Electronics Conference, 282–285.

Fildes, A., Leening, A. and Corrigan, J. (1990). Speed Perception 2: Drivers' Judgements of Safety and Speed on Rural Straight and Curved Roads and for Different Following Distances. Federal Office of Road Safety, Australia, Contract Report 60.

Freund, D. M., Knipling, R. R., Landsburg, A. C., Simmons, R. R. and Thoma, G. R. (1995). A Holistic Approach to Operator Alertness Research. Paper Presented at the Transportation Research Board, 74th Annual Meeting, Washington, DC.

Garlan, C. and Shaw, M. (1993). An introduction to software architecture. [Book Chapter] *Advances in Software Engineering and Knowledge Engineering*. Singapore: World Scientific, 1–39.

Gish, K. W., Staplin, L., Stewart, J. and Perel, M. (1999). Sensory and cognitive factors affecting automotive head-up display effectiveness, *Transportation Research Record* **1694**, 10–19.

GM. (2001). Cadillac Vizon Offers Unprecedented Safety Technology. http://media.gm.com/division/cadillac/concept_vehicles/vizon_technology.html.

Goodrich, M. A. and Boer, E. R. (2000). Designing human-centered automation: Tradeoffs in collision avoidance system design. *IEEE Transactions on Intelligent Transportation Systems* **1**, **1**, 40–54.

Gorjecstani, A., Shankwitz, C. and Donath, M. (2000). Impedence Control for Truck Collision Avoidance, Proceedings of the American Control Conference. Chicago, IL.

Groeger, J. A., Alm, H., Haller, R. and Michon, J. A. (1993). Impact and acceptance. [Book Chapter] *Generic Intelligent Driver Support*, J.A. Michon, ed., Washington DC: Taylor & Francis, 217–227.

Hayami, M., Ohta, T. and Tajima, T. (1999). The Development of Dangerous Zone Avoidance Control System in UTMS. 1999 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems, 49–54.

Hendersen, S. and Suen, S. L. (1999). Intelligent Transportation Systems: A Two-Edged Sword for Older Drivers?, *Transportation Research Record* **1679**, 58–63.

Hoffman, A., Horn, E., Keller, W., Renzel, K. and Schmidt, M. (1996). The Field of Software Architecture. Technical University of Munich, Technical Report TUM-I9641.

Hornik, K., Stinchombe, M. and White, H. (1989). Multilayered feedforward networks are universal approximators. *Neural Networks* **2**, **5**, 359–366.

Hu, J., Lygeros, J., Prandini, M. and Sastry, S. (1999) A Probalistic Framework for Highway Safety Analysis. Proceedings of the 38th Conference on Decision & Control, Phoenix, AZ, 3734–3739.

Jacoy, E. H. and Knight, J. R. (1998). Adapting Radar and Tracking Technology to an On-Board Automotive Collision Warning System. The AIAA/IEEE/SAE IEEE Digital Avionics Systems Conference **2**, **I24**, 1–8.

Kamiya, H., Fujita, Y., Tsuruga, T., Nakamura, Y., Matsuda, S. and Enomoto, K. (1996) Intelligent Technologies of Honda ASV, Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 236–241.

Kao, J. H., Hemmerle, J. S. and Prinz, F. B. (1996). Collision avoidance using asynchronous teams. *1996 IEEE International Conference on Robotics and Automation* **2**, 1093–1100.

McKnight, A. J. and Adams, B. B. (1970). Driver Education Task Analysis Alexandria, VA: Human Resources Research Organization; Springfield, VA: Reproduced by the National Technical Information Service.

McLoughlin, H. B., Michon, J. A., van Winsum, W. and Webster, E. (1993). GIDS intelligence. [Book Chapter] *Generic Intelligent Driver Support*. J.A. Michon, ed., Washington DC: Taylor & Francis, 89–112.

Menig, P. and Coverdill, C. (1999). Transportation Recorders on Commercial Vehicles, 1999 NTSB International Symposium on Transportation Recorders.

Mercedes-Benz. (2001). Mercedes-Benz CL 500, http://www.m-benz.com/models/cl500-01.html.

Michon, J. A. and Smiley, A. (1993). Introduction: A guide to GIDS. [Book Chapter] *Generic Intelligent Driver Support*, J.A. Michon, ed., Washington DC: Taylor & Francis, 3–18.

Mimuro, T., Miichi, Y., Maemura, T. and Hayafune, K. (1996). Functions and Devices of Mitsubishi Active Safety ASV. Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 248–253.

Moray, N. (1990). Designing for transportation safety in the light of perception, attention, and mental models, *Ergonomics* **33**, **10/11**, 1201–1213.

Perry, C. and Wolf, A. (1992). Foundations for the Study of Software Architecture. *Software Engineering Notes* **17**, **4**, 40–52.

Sayed, R. and Eskandarian, A. (2001). Monitoring Drowsy Drivers with Artificial Neural Network. Miami Beach, FL: Intelligent Transportation Society of America 2001, 1–12.

Schmitt, M., Ullrich, T., Tolle, H. (1994). Associative Datafields in Automotive Control, Proceedings of the Third IEEE Conference in Control Applications **2**, 1239–1244.

Shladover, S. E. (1999). Intellectual challenges to deployability of advanced vehicle control and safety systems. *Transportation Research Record* **1679**, 119–125

Shields, T. R. and Roser, M. (2000). Trends in Automotive Use of ITS Technologies for Safety. Seoul 2000 FISITA World Automotive Congress, Seoul, Korea, 1–4.

Sidani, T. A. and Gonzalez, A. J. (2000). A framework for learning implicit expert knowledge through observation. *Transactions of the Society for Computer Simulation International* **17**, **2**, 54–72.

Sobottka, K. and Bunke, H. (1999). Anytime Behavior for Obstacle Tracking. 1999 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems, 368–373.

Sourour, E. and Nakagawa, M. (1999). Mutual decentralized synchronization for intervehicle communications. *IEEE Transactions on Vehicular Technology* **48**, **6**, 2015–2027.

Swaroop, D. and Huandra, R. (1998). Intelligent cruise control design based on a traffic flow specification. *Vehicle Systems Dynamics* **30**, 319–344.

Talukdar, S. N. (1998). Collaboration rules for autonomous software agents. *Decision Support Systems* **24**, **3-4**, 269–278.

Varaiya, P. (1993) Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control* **38**, 195–207.

Venhovens, P., Naab, K. and Adiprasito, B. (2000). Stop and go cruise control. *International Journal of Automotive Technology* **1**, **2**, 61–69.

Woll, J. (1994). Radar based collision warning system. *SAE Paper No.* 94CO36.

Zachman, J. (1999). A Framework for Systems Architecture. *IBM Systems Journal* **38**, **2-3**, 454–470.

Zhang, W., Shladover, S., Hall, R. and Plocher, T. (1994). A functional definition of automated highway systems. *Transportation Research Board* **73**.