DRIVER ADVOCATE^{TM} TOOL

Chip Wood, Robert Leivian, Noel Massey, Jack Bieker, John Summers
Human Interface Lab-Phoenix
Motorola Labs, Motorola
7700 S. River Parkway
Tempe, AZ 85284 USA
E-mail: Chip.Wood@motorola.com

Summary: Using scenario driven research, a Driver Advocate^{TM} (DA) [1] system has been designed to advise the driver about potentially unsafe situations based on information from environmental sensors [2]. DA is an intelligent dynamic system that monitors, senses, prioritizes, personalizes, and sends alerts to the driver appropriate to the moment. This has the potential to sharply decrease driver distraction and inattention. To support the realization of DA, a DA Tool (DAT) has been developed to coordinate with a KQ (previously Hyperion) virtual driving simulator and allow the merging of the simulated driving performance, the enviormental sensors, and the intelligent use of audio, visual, and tactile feedback to alert the driver to potential danger and unsafe driving behavior. DAT monitors the traffic, lane following, forward and side clearances, vehicle condition, cockpit distractions, Infotainment use, and the driver affective behavior. The DAT is designed to be highly configurable, flexible, and user friendly to facilitate creative freedom in designing usability and human factors experiments and rapid prototyping.

INTRODUCTION

The combination of new Telematics' services, the increase in sophistication and functionality of modern electronics in the automobile, the proliferation of "multi-tasking" while driving, the "maturing" driver demographics, and increased traffic congestion form a potential problem to driver focus and safety. It requires a new paradigm for the driver-car interface. Using concepts from affective computing, scenario research, cognitive psychology, system engineering, intelligent agents, and more traditional human machine interface and usability design, the Driver Advocate^{TM} (DA) processes the incoming environmental, mechanical, and traffic information, infotainment status, and driver behavior to alert and focus driver's attention on the most "important" task-safety. DA monitors all driver-related events: road and lane position, traffic situational awareness- immediate and global, automobile mechanical status, weather and road conditions, cockpit conditions, both local and incoming, and finally the driver's physiology, habits, and instantaneous attention. The driver always remains in the loop and in control at all times and is merely alerted and advised to potential problems as the DA senses, analyzes, and prioritizes them. Only through systematic research, testing and evaluation can we hope to mitigate driver error due to distraction and only through a complete system approach can the integration and fusion of information from many disparate sources and types of sensors be prioritized and presented as clear, personalized, and unambiguous information to the driver in a timely manner. The alert and advice formulation needs to be optimized and personalized so that the driver’s learned automatic reaction is paramount and reaction time is minimized.

The purpose of this paper is to outline the functions, flexibility, and architecture of the DA Tool (DAT) Motorola has built to facilitate the research needed to further refine the approaches.

The DAT is a software state machine and a set of virtual sensors that monitor the virtual world of a KQ driving simulator. The DAT constantly monitors the environment, car position, traffic situation and gives ‘advice’ to the driver in real-time about ‘problems’ in that environment. This is facilitated through custom software, DA Sensor Protocol, (DASP) which interfaces to the KQ simulator’s virtual world. DASP is an efficient means to gather only the relevant positional and traffic data feedback from the simulator necessary to determine the appropriate DA advice to the driver. This advice is output through another set of customized interface software using audio, visual and/or tactile feedback (DA A/V&T). The style and presentation of this feedback is based on the individual driver’s preferences and learned habits. It is designed to inform the driver of problem situations and to elicit a desired response. For example, if the driver is too close to the car in front, a ‘virtual brake light’ could be flashed in the virtual conformal Heads Up Display (DA HUD). DA HUD is more custom software that allows the visual alerts of DA to be presented on the KQ simulator screen in the correct visual perspective view of the driver to the virtual world in front. Experiments will be performed to determine if this particular alert manifestation will cause a learned automatic reaction in most drivers to start to apply their brakes.

PHILOSOPHY

The DAT is designed to be highly configurable to allow for creative freedom in designing user experiments. A primary research area in these experiments is to determine the appropriate action DA should take in order to direct the driver's attention to potential dangers. For example, in certain situations an audible alert may be most effective while in other circumstances visual and/or tactile alerts may be most effective. Since we need to understand how a person reacts to different alerts, we need to be able to configure Driver Advocate to respond identically at the same time under the same conditions but by presenting the driver with alerts that are easily changed for different experiments. DAT accomplishes this flexibility by separating the task of deciding when to alert the driver from which alert to present. This allows the researcher freedom to run experiments with different alert types while maintaining the conditions under which the driver will be alerted.

Another aspect of determining appropriate DA actions is understanding how multiple alerts interact. By integrating what might be separate components (such as lane tracking, forward collision warning, and infotainment messages) DAT has the advantage of deciding how messages from these various components will interact. This can improve driver safety by blocking or delaying low priority messages to prevent driver overload in dangerous situations. Experimenters are able to configure alert priorities by specifying five 'alert levels' for each component (so there are five levels of lane tracking alerts, five levels of forward collision alerts...). Effectively, these five alert levels specify how alerts escalate within each component. DA ensures that only one alert level is active at any instant in time. Furthermore, the alerts for any component can be based on the messages from other components so that the infotainment messages might not be presented to the driver when the forward collision-warning component is signaling a high priority alert. The DAT thereby allows rapid configuration for multiple experiments aimed at determining appropriate alert escalations and interactions.

DAT ARCHITECTURE

The DAT has three major parts: *info fusion*, *response selector,* and *action generator*.

Info Fusion
Monitors 100+ Conditions:
  Driver Controls
  Environment
  Vehicle
  Traffic
  Cockpit/ Infotainment Distractions
Provides to Response Selector:
  Condition's Current State
  Condition's State History

Response Selector
Determines 'Problems':
  Trigger: Recognizes particular States that require alert action(s)
  Agenda: Determines priority, type, and timing of alert action(s)
  Resolver: Recognizes particular States that terminate alert action(s)

Action Generator
Generates a specified set of audio, visual, and/or tactile alert action(s)

Table 1

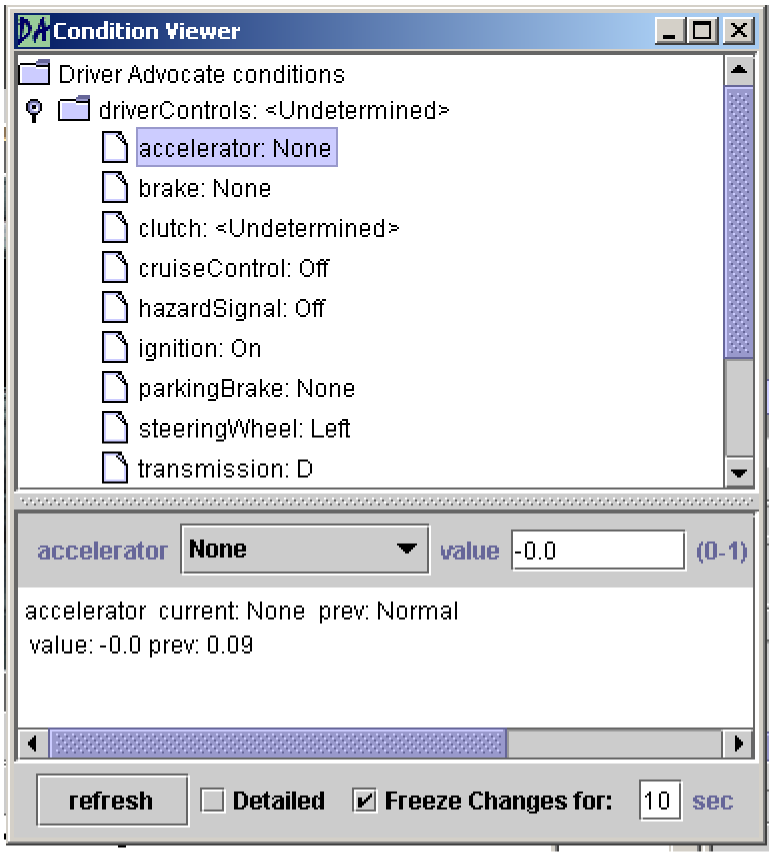
The *info fusion* has input from all attached sensors and fuses all information into a tree of 'conditions' and determines at any given point the current 'state' of each condition in the tree. The *response selector* monitors this 'condition tree' for a predefined set of 'problems' that are defined by pre-established combinations of condition states. The *action generator* then sorts any current 'problems' by pre-established priority scheme and executes a customized driver-specific 'agenda' of alerts to inform and correct the problem.

The *info fusion* uses traditional programming to 'fuse' information from one or more sensor inputs into a single state for each condition. The conditions are grouped into categories: *driver controls, environment, vehicle, traffic, and cockpit/infotainment distractions*. These major categories can be further subdivided within the whole set called the 'current condition tree'. Each condition in the tree is always in one of N possible fuzzy states. The program learns the correct threshold for each state over time based on the individual drivers. For example, the subcondition 'speed' of the condition 'environment' is set to one of "undetermined", "Backing", "Stopped", "Slow", "Normal", "Fast", or "Speeding". The condition has a numeric value in kilometers per hour but also is classified as either Normal or Fast, etc., based on the condition 'speed limit', the learned normal habits of the current driver, and the speed of surrounding traffic. Currently over

113 conditions are constantly monitored in real-time by the info fusion software. Each condition can be sampled at any pre-determined rate from 1 to 100 times per sec.

The *response selector* is informed by *info fusion* whenever a condition changes state (e.g. speed changing from Normal to Fast). The *response selector* then uses a predefined set of 'problem' definitions to determine if the new state of the condition either: generates a problem, solves a problem, or is innocuous. A 'problem' is defined by three components: a trigger, agenda, and a resolver. The "trigger" is used to define when a 'problem' has begun and the "resolver" is used to determine when the problem has been resolved. They are both Boolean binary expressions formed by a combination of statements which are OR'd or AND'd together. The statements themselves are values of conditions and states (for example, the following expression can be used to trigger a headway problem (trafficAhead = TooClose) AND (laneNumber != Opposing)). The "agenda" specifies the actions, order, and timing of all A/V&T actions and alerts to be taken to advise the driver of the problem and possible solutions

The *action generator* is a preprogrammed library of actions that can be executed by the system (e.g. making beeps or chirps, vibrating controls, visual cues in a HUD etc.) This runs agendas for each current problem and sorts out the order of presentation of each agenda by priority.



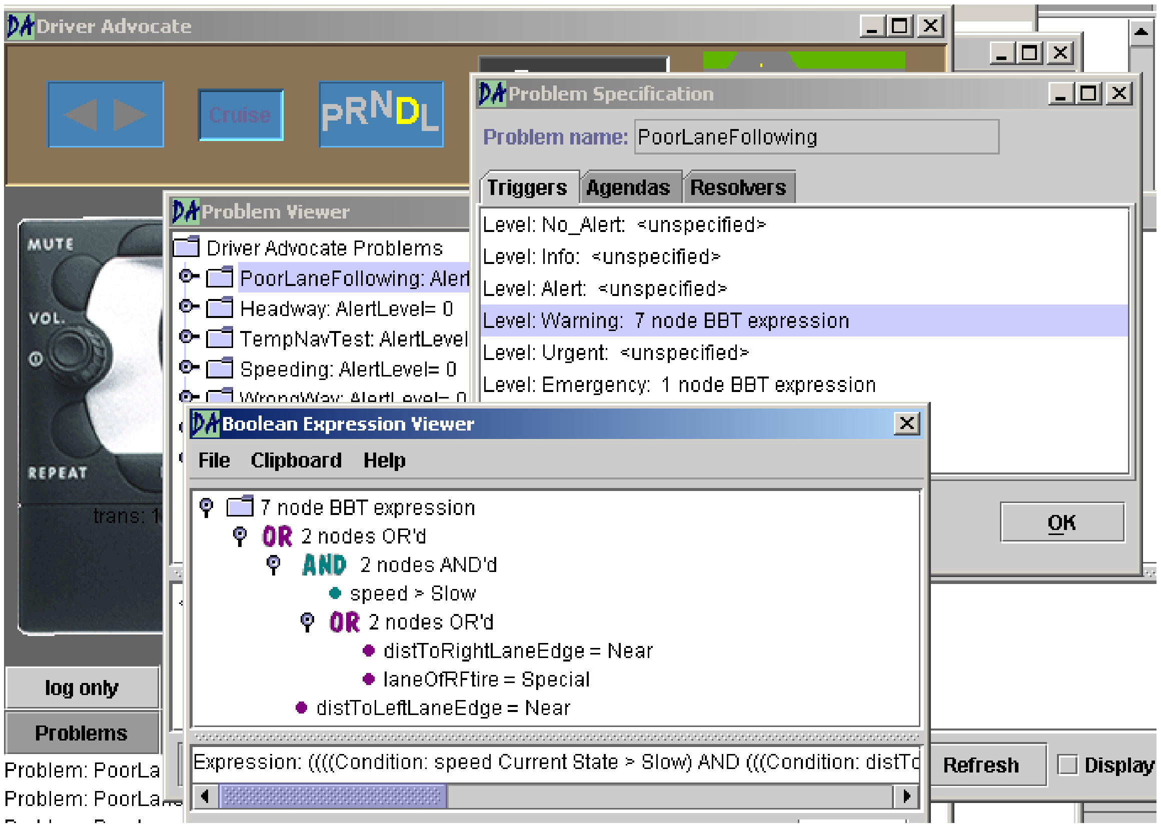
```
Condition Viewer
Driver Advocate conditions
  driverControls: <Undetermined>
      accelerator: None
      brake: None
      clutch: <Undetermined>
      cruiseControl: Off
      hazardSignal: Off
      ignition: On
      parkingBrake: None
      steeringWheel: Left
      transmission: D

accelerator  None  value -0.0  (0-1)
accelerator  current: None  prev: Normal
 value: -0.0 prev: 0.09

refresh   Detailed   ✓ Freeze Changes for: 10 sec
```

Figure 1 Condition viewer.

The Condition Viewer [Fig#1] is used to monitor and patch all current values derived by the current state of the sensors. In this case, the condition of the accelerator pedal is shown, as 'none', that is, the driver's foot is off the pedal. It was 9% depressed the previous sampling time. The current driver advocate system monitors over 100 such conditions and the changes in state for all these conditions is used to determine any response selector actions in monitoring and correcting problems that occur in real time.



Problem name: PoorLaneFollowing

Level: No_Alert: <unspecified>
Level: Info: <unspecified>
Level: Alert: <unspecified>
Level: Warning: 7 node BBT expression
Level: Urgent: <unspecified>
Level: Emergency: 1 node BBT expression

Driver Advocate Problems
PoorLaneFollowing: Alert
Headway: AlertLevel= 0
TempNavTest: AlertLevel
Speeding: AlertLevel= 0
WrongWay: AlertLevel= 0

```
7 node BBT expression
  OR 2 nodes OR'd
    AND 2 nodes AND'd
      speed > Slow
      OR 2 nodes OR'd
        distToRightLaneEdge = Near
        laneOfRFtire = Special
    distToLeftLaneEdge = Near
```

Expression: ((((Condition: speed Current State > Slow) AND (((Condition: distTo

Problem: PoorLa
Problem: PoorLa

Figure 2 Problem viewer and current definitions programmed into the system

The Problem Viewer window [Fig#2] is displaying the trigger for the warning level of the problem: PoorLaneFollowing. The trigger is defined as speed greater then slow and begin near the right edge or having a right front tire in a special (e.g. turn or passing lane) lane. Using these windows, selector buttons, and popup menus the researcher can easily configure the behavior of the DA system for controlled experiments in various driving scenarios.

REFERENCES

[1] Driver Advocate^{TM} is a trademark of Motorola, Inc.

[2] Remboski, Gardner, Wheatly, Hurwitz, MacTavish, Gardner, "Driver Performance Improvement through the Driver Advocate: A Research Initiative toward Automotive Safety", Proc of the 2000 International Congress on Transportation Electronics, SAE P-360, pp509-518